МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет водного господарства та природокористування

Л. В.Зубик
І. М. Карпович
О. М. Степанченко

ОСНОВИ СУЧАСНИХ WEB – ТЕХНОЛОГІЙ. ЧАСТИНА 1

Навчальний посібник

Для студентів спеціальностей
113 “Прикладна математика та механіка”
122 “Комп’ютерні науки та інформаційні технології”
123 “Комп’ютерна інженерія”

Рівне – 2016

УДК 004(075.8)
ББК 32.973Я73
З91

Рекомендовано до видання вченою радою Національного університету водного господарства та природокористування. Протокол № 13 від 27 листопада 2015 р.

Рецензенти:
Мартинюк П. М., доктор технічних наук, завідувач кафедри прикладної математики Національного університету водного господарства та природокористування, м. Рівне;
Сафоник А. П., кандидат технічних наук, доцент Національного університету водного господарства та природокористування, м. Рівне.

Л. В. Зубик, І. М. Карпович, О. М. Степанченко
391 Основи сучасних web-технологій. Частина 1. Навчальний посібник. – Рівне: НУВГП, 2016. – 290 с.

Розглянуто основні напрямки проектування, розробки та супроводу інтерактивних web-додатків з використанням новітніх технологій.

Для студентів, які здобувають освіту в галузях знань “Інформаційні технології” та ”Математика та статистика”, а також спеціалістів-практиків, які займаються проблемами автоматизації та інформатизації діяльності підприємств.

УДК 004(075.8)
ББК 32.973Я73

Зміст

Передмова

Навчальний посібник **"**Основи сучасних web-технологій**"** призначено для підготовки студентів спеціальностей "Прикладна математика та механіка", "Комп'ютерні науки та інформаційні технології", "Комп'ютерна інженерія" відповідно до вимог державного стандарту освіти.

Створення національної Grid-інфраструктури для розвитку інформаційного суспільства в Україні передбачає широке використання інформаційних і комунікаційних технологій в усіх галузях господарства. Ефективне управління виробництвом, формування оптимальних рішень у різноманітних сферах економіки та бізнесу, сталий розвиток підприємства значною мірою залежать від рівня впровадження та використання інформаційних систем і технологій. Характерною рисою адаптації інформаційної системи до використання в глобальній мережі Internet є наявність у ній web-сайту, доступного користувачам завдяки сервісу WWW. Все це сприяло появі нового напрямку в галузі програмування – web-дизайну.

Фахівцям з інформаційних технологій доводиться здійснювати розробку серверного та клієнтського програмного забезпечення, адміністрування web-сайту, реалізацію заходів із підвищення ефективності його функціонування та формування дизайну web-сайту. Як свідчить практика, важливим завданням web-дизайнера є розробка та підтримка клієнтського програмного забезпечення web-сайту для забезпечення високої ефективності його функціонування. Всі ці завдання потребують достатньо глибоких професійних знань та навиків. Тому основна увага в навчальному посібнику спрямована на принципи та прийоми створення клієнтської частини web-сайту засобами HTML, CSS та JavaScript.

Досвід впровадження сучасних складних web-орієнтованих систем свідчить про необхідність їх розробки шляхом активної взаємодії між компанією-розробником програмного забезпечення та представниками замовника. Для цього необхідно знати принципи проектування та розробки web-сайтів, мати чітке уявлення про їх структуру, принципи функціонування, місце і роль в організації роботи підприємства,

мати уявлення про перспективи подальшого розвитку web-систем, а також вміти їх використовувати на високому рівні.

Широкого розповсюдження набули різноманітні Internet-додатки, зокрема, пошукові інформаційні системи, засоби електронної комерції, соціальні мережі, інформаційні, наукові та освітні портали, web-сайти органів державного управління тощо. Крім того, велика кількість програмного забезпечення, для функціонування якого використовується так званий віконний інтерфейс користувача, поступово переводиться у web-орієнтований спосіб взаємодії із користувачем для підвищення ефективності розробки та супроводу.

Вимоги до web-програмістів постійно ростуть. Час виготовлення, якість і невисока вартість продукції були і залишаються головними критеріями при виборі доброго розробника. Оволодіння технологіями серверного і клієнтського програмування дозволить кваліфіковано розв'язувати поставлені завдання в найкоротший термін.

Основним завданням посібника є огляд відомих підходів та технологій, мов програмування і загальноприйнятих стандартів проектування та створення web-документів. Комплекс лабораторних робіт, викладених у посібнику, допоможе студентам здобути практичні навички створення web-додатків з використанням сучасного інструментарію.

Навчальний посібник містить також матеріали для самостійної роботи студентів, спрямованої на закріплення знань, що одержані під час проведення аудиторних занять. Для оцінки та практичної апробації отриманих знань в кінці кожного розділу вміщено контрольні запитання та завдання. Поглибленому вивченню матеріалу сприятимуть літературні та електронні джерела, перелік яких наведено наприкінці посібника.

Навчальний посібник – результат спільної праці з рівною часткою творчого внеску кожного з авторів. Автори сподіваються, що посібник буде корисним як web-програмістам-початківцям, так і досвідченим спеціалістам-практикам.

1. HTML5

Окремий інформаційний ресурс мережі WWW (World Wide Web), що має власну адресу, називають web-сторінкою або web-сайтом. Для передачі даних у мережі WWW використовується протокол передачі даних прикладного рівня HTTP (англ. Hyper Text Transfer Protocol).

Програмне забезпечення, що надає користувачам інтерфейс для доступу до інформації web-сторінок, називають **браузером**. Призначення будь-якого web-браузера - прочитати документ і відобразити його як web-сторінку. *Приклади* популярних браузерів: Google Chrome, Mozilla Firefox, Opera, Internet Explorer, Safari тощо (браузери у списку перелічено у порядку спадання їх популярності в Україні за з даними StatCounter станом на початок січня 2015 р.).

Для звертання до сайту використовують доменне ім'я, яке складається з кількох доменів, відокремлених крапкою. *Наприклад*, `maps.google.com` означає, що домен третього рівня `maps` входить в домен другого рівня `google`, який, відповідно, входить в домен першого (верхнього) рівня `com`. Домени верхнього рівня поділяють на загальні і географічні. Загальні домени призначені для певного класу організацій, *наприклад*: .com – для коммерційних сайтів; .org – для некомерційних організацій; .net – для сайтів, діяльність яких пов'язана з Інтернетом. Географічні домени призначені для конкретної країни, *наприклад*: .ua – Україна, .us – США, .eu – Європейський союз, .de – Німеччина.

1.1. Правила побудови HTML–документів

Основи web-дизайну та удосконалення проекту

Будь-який ресурс, опублікований у Всесвітній мережі, від глобального інформаційного порталу з десятками тисяч відвідувачів в день до скромної домашньої сторінки, куди заходять усього дві людини протягом місяця, - це насамперед художній твір, складний комплекс інженерно-дизайнерських рішень. Процес створення такого твору називається web-дизайном.

При плануванні і створенні будь-якого web-ресурсу слід пам'ятати, що головний критерій розробки сторінок - зручність користувача, тобто майбутнього відвідувача сайту. Складність реалізації цього правила обумовлена широким спектром апаратних і програмних засобів як розробників, так і потенціальних відвідувачів. Звідси виникає необхідність певної стандартизації підходів до web-дизайну, вироблення алгоритмів, які могли б задовольнити всю потенційну аудиторію.

Причини для створення web-ресурсу можуть бути різними: інформація, яка допоможе знайти роботу, публікація літературних творів, рисунків, музики, наукових досліджень автора. Некомерційний сайт, зазвичай, виростає з добре продуманої домашньої сторінки.

Комерційні Інтернет-проекти організуються з метою отримання фінансового прибутку, який може бути як прямим, так і опосередкованим (практично безкоштовна реклама в мережі Internet, швидкий доступ за ключовими словами до цільової аудиторії, пошук клієнтів, партнерів та ін.).

Отримання прямого прибутку передбачає створення проекту, розрахованого на залучення фінансових засобів безпосередньо з Internet. Компанії, що живуть за рахунок створення web-ресурсів, розробляють пошукові машини, інформаційні портали, Internet-магазини, інтерактивні аукціони, віртуальні рекламні агенства тощо.

HTML-документ – файл текстової або нетекстової природи (звук, відео, зображення), створений за допомогою **мови гіпертекстової розмітки HTML** (англ. Hyper Text Markup Language, сучасні версії HTML5, XHTML), **каскадних таблиць стилів CSS** (англ. Cascading Style Sheets) та програм (сценаріїв), написаних мовою JavaScript, PHP тощо.

HTML5 - це остання версія HTML. В ній враховані зміни, які відбулися у всесвітній павутині з часу створення попередньої версії у 1999 році. В HTML5 було додано багато нового, тому деякі можливості, які були доступні в HTML4 тільки з використанням зовнішніх плагінів (зокрема, Adobe Flash) або клієнтських скриптів, тепер стали доступні для звичайних тегів. HTML5 підтримують практично всі сучасні браузери, що

дозволяє використовувати нові можливості без зайвих побоювань.

Найважливіші нововведення HTML5:

- підтримка відео і аудіо (елементи video і audio);
- реалізація можливості малювання на web-сторінках довільних об'єктів (елемент `canvas`);
- підтримка форм більш складної структури (нові значення `type` в елементі `<input>` та чимало нових елементів і атрибутів);
- додавання семантичних тегів, що дозволяють зробити web-сторінки більш зрозумілими для пошукових систем, браузерів і інших програм чи пристроїв, які читають web-сторінки (елементи `footer, header, nav, article`);
- організація DOM (Document Object Model) сховищ - більш функціональної альтернативи `cookie`.

Якщо HTML-документ відкрити у редакторі текстів, він буде відображений як звичайний текстовий файл. Якщо ж HTML-документ відкрити у браузері, він буде відображений відповідно до тегів розмітки і називатиметься **web--сторінкою**.

В документі може міститись інформація двох типів: звичайний текст та команди (**теги**) мови HTML. Всі теги мови HTML обмежуються символами ”**<**” та “**>**”, між якими записується **ідентифікатор** (**ім’я**) **тега**. Розрізняють **одинарні** та **парні теги**.

Наведемо *приклад* одинарного тега для відображення у вікні браузера горизонтальної лінії `<hr>`. На відміну від одинарного, для визначення парного тега (**тега-контейнера**) використовуються **відкриваючий** та **закриваючий тег**. Закриваючий тег відрізняється наявністю символа “*/*” перед ім’ям тега. Деяким тегам (наприклад, **
) притаманна скорочена форма запису (
**). *Наприклад*:

```
<td> Зміст тега </td>
```

Для розробки сучасного повноцінного сайту недостатньо знати тільки HTML. Щоб створити гарний і функціональний сайт, потрібно володіти хоча б такими технологіями, як CSS, JavaScript, PHP. Крім того, бажано вміти працювати принаймні з

одним графічним пакетом, наприклад, Adobe Photoshop і мати навички створення банерів і flash-об'єктів. Необхідно також знати основні прийоми роботи та розуміти структуру файлів баз даних, наприклад, MySQL або PostgreSQL. Для досягнення запланованих результатів у терміни, визначені замовником продукту, слід бути ознайомленим із можливостями та принципами організації роботи у системах управління вмістом сайту - **CMS** (англ. Content Management System). Ознайомлення варто розпочинати із простіших варіантів: Joomla, WordPress, Drupal. Після цього можна опановувати і більш складні за структурою та можливостями продукти: Magenta, 1С Бітрікс тощо.

Створення HTML-документів

Для створення HTML-документа можна:

- відкрити текстовий редактор;
- набрати довільний текст і розмітити його HTML-тегами;
- зберегти файл з розширенням *.htm* або *.html*.

Після відкриття створеного файлу за допомогою web-браузера, він буде відображений як web-сторінка. Крім того, можна скачати готовий HTML-документ і відредагувати його для своїх потреб за допомогою текстового редактора. Як і будь-який інший файл, html-файл має ім'я і вміст, який повинен відповідати певним вимогам.

Ім'я HTML-документа. Нехай користувач придбав домен, наприклад, example.com і розмістив у ньому набір HTML-документів. Якщо відвідувач введе в адресному рядку браузера URL-адресу http://example.com, то web--сервер спробує відкрити HTML-документ з ім'ям *index.html* (або *index.htm* чи *index.php*) - домашню сторінку.

Вміст HTML-документа має чітко визначену структуру.

Першим у HTML -документі буде вказаний **DOCTYPE** або **DTD** (англ. Document Type Definition - оголошення типу документа) – рядок, що повідомляє браузеру, яку версію планується використовувати при створенні HTML-сторінки. DOCTYPE вказується перед тегом <html>.

Наприклад, для HTML5 DOCTYPE має такий вигляд:

<!DOCTYPE HTML>

Визначення загальних розділів HTML -документа

Після оголошення DOCTYPE першим тегом HTML-документа є парний тег <html> </html>. Тег <html> є тегом-контейнером для всієї HTML-сторінки. Він містить заголовок документа і тіло документа.

Розділ заголовка відкривається тегом **<**head**>** і закривається тегом **</**head**>**. Елементи, розташовані в секції **<**head>, не відображаються на сторінці, а використовуються для службових цілей. У секції <head> можуть розташовуватися скрипти, інструкції про оформлення сторінки та інша мета-інформація про HTML-документ.

Наступні елементи повинні обов'язково розташовуватися в секції <head>: <link>, <title>, <base>,<style>, <script> і <meta>.

Змістова частина документа визначається тегом-контейнером **<body> </body>**. Отже, структура найпростішої HTML-сторінки буде мати вигляд:

```
<html>
<head>
<title> Заголовок </title>
</head>
<body>
</body>
</html>
```

Розглянемо детальніше групу елементів заголовка.

Елемент **<title>** обов'язково повинен бути присутнім у всіх HTML-документах. Елемент <**title**>:

- визначає заголовок вікна браузера;
- використовується як заголовок сторінки в результатах видачі пошукових систем;
- використовується як заголовок сторінки при додаванні сайту в обране.

Приклад запису:

```
<title> Назва HTML-сторінки </title>
```

Завданням заголовка є надання браузеру інформації, необхідної для коректного відображення HTML-сторінки. Теги,

що розміщені всередині розділу заголовка (крім назви документа), на екрані не відображаються.

Елемент <meta>

В розділі заголовка часто використовують тег **<meta>**. В ньому розміщується інформація про дані, які знаходяться в HTML-документі. *Приклад* метаданих: кодування сторінки, короткий опис вмісту, ключові слова, підпис автора, дата останньої модифікації. Метадані зазвичай не відображаються на сторінці, але використовуються браузерами і пошуковими системами.

Призначення тега **<meta>**:

1. Визначення кодування тексту HTML-сторінки. Для повідомлення браузеру користувача про кодування, яке використане на цій сторінці, необхідно застосувати атрибут тега <meta> **charset**. *Наприклад*, для кодування у форматі windows-1251 цей тег можна записати у вигляді:

```
<meta http-equiv = “Content-Type” content = "text/html; charset = windows-1251"/>
```

Досвідчені web–розробники рекомендують використовувати кодування UTF-8, яке дає можливість використовувати символи будь-якої мови світу. В HTML5 таке кодування може бути задано так:

```
<meta charset="UTF-8"/>
```

Якщо кодування явно не вказано, браузер при відображенні сторінки буде визначати його автоматично. Якщо кодування при цьому буде визначено неправильно, користувач побачить сторінку, яка містить безглузді символи. Тому кодування обов'язково повинно вказуватися до кожного HTML-документа.

2. Визначення ключових слів (англ. keywords), що використовуються пошуковими системами. Деякі пошукові системи під час індексації сторінки звертаються до мета-елементів. Наступний мета-елемент задає ключові слова для документа:

```
<meta name="keywords" content="HTML, CSS, JavaScript, jQuery, AJAX" />
```

Зауваження: у зв'язку з тим, що багато web-розробників для підняття рейтингу видачі сайту вказують велику кількість ключових слів, які не стосуються їх сайту, але користуються популярністю, пошукові системи були змушені припинити використання ключових слів тега `meta в` якості анотацій сторінок.

3. Визначення опису HTML-сторінки в пошуковій системі.

Наприклад:

```
<meta name="description"
content="Безкоштовні онлайн підручники з HTML,
CSS, JavaScript, jQuery, HDOM і AJAX" />
```

4. Відображення у браузері визначеної HTML-сторінки.

Наприклад, для переходу на HTML-сторінку *untitled1.html* через 20 секунд після відображення поточної HTML-сторінки необхідно записати :

```
<meta http-equiv="refresh" content="20;
URL=untitled1.html" />
```

Зауваження. В пунктах 1-4 використана скорочена форма запису парного тега **<meta>**, а саме **<meta />**.

Решту елементів, що відносяться до розділу `<head> </head>`, а саме `<link>,<base>,<script>` та `<style>` ми розглянемо детальніше в наступних розділах.

Тег **<body>** може мати низку необов'язкових параметрів (табл. 1.1), які визначають форматування всієї HTML-сторінки.

Наприклад, для визначення документа з текстом зеленого кольору та відсутністю смуг прокрутки слід написати:

```
<body text="green" scroll="0">
```

Приклад шаблону HTML-документа*:*

```
<!DOCTYPE HTML>
<html>
<head>
<title> Заголовок </title>
<meta  charset=”UTF-8" />
</head>
<body>
</body>
</html>
```

Збереження і перегляд документа

Якщо передбачається відкривання сторінки в кодуванні UTF-8, то HTML-документ краще зберігати в цьому кодуванні.

Таблиця 1.1.

Параметри тега <body>

Параметр	Призначення
alink	Визначає колір активного гіперпосилання
background	Вказує на адресу фонового зображення
bottommargin	Визначає межу нижнього поля документу
bgcolor	Визначає колір фону документу
bgproperties	Дозволяє або забороняє прокрутку фонового зображення
leftmargin	Визначає межу лівого поля документу
link	Визначає колір не переглянутого гіперпосилання
rightmargin	Визначає межу правого поля документу
scroll	Встановлює або забороняє смуги прокрутки вікна браузера
text	Визначає колір тексту
topmargin	Визначає межу верхнього поля документу
vlink	Визначає колір переглянутого гіперпосилання

Збережений документ можна переглянути в будь-якому доступному браузері. Коли верстка здійснюється на замовлення, відображення web-сторінки слід перевіряти у кількох найбільш популярних браузерах. Скачати версії популярних браузерів можна за такими посиланнями:

- Google Chrome — http://www.google.ru/chrome;
- Mozilla Firefox — http://www.mozilla.org/ru/firefox/new/;
- Opera — http://www.opera.com/download/;
- Internet Explorer — http://windows.microsoft.com/ru-RU/internet-explorer/downloads/ie;

- Safari — http://www.apple.com/ru/safari/download/.

Редактори для верстки

Багато спеціалізованих редакторів підсвічують синтаксис HTML, CSS, PHP і полегшують пошук помилок, містять інтерактивну довідку про аргументи функцій, інструменти вибору кольору, призначення того чи іншого тега/атрибуту/параметра (на відміну від редактора *Блокнот*). Використання для верстки, зокрема, MS Word *категорично заборонено*.

Notepad++. Популярна безкоштовна заміна *Блокнота*. Програму можна скачати із сайту http://notepad-plus-plus.org/. Одна із функцій програми — підсвічування синтаксису різних мов програмування, зокрема HTML, CSS, JavaScript, PHP. Схожі можливості у редактора *SciTE* (http://code.google.com/p/scite-ru/).

phpDesignerPro. Спеціалізована програма для професійної верстки і програмування, більш функціональна, ніж *Notepad++*. Її можна скачати із сайту http://www.mpsoftware.com. Ознайомлювальна версія цієї платної програми працює 21 день. Деякі можливості програми:

- містить шаблони різноманітних документів, в тому числі, HTML-документа;
- передбачене підсвічування синтаксису: PHP, XHTML, HTML5, CSS3, JavaScript, XML, Perl, VBScript, SQL, Java та ін. Крім підсвічування синтаксису, програма вміє автоматично доповнювати незавершені теги, атрибути, властивості і параметри CSS, функції PHP і jQuery. Функція автоматичного доповнення зв'язана із функцією підказки;
- завдяки вбудованому візуальному режиму можна перевіряти вигляд HTML-документа в різних популярних браузерах (для цього браузери мають бути встановлені на комп'ютері і в налаштуваннях програми *phpDesignerPro* повинні бути вказані правильні шляхи до цих браузерів на ПК).

Інші корисні програми для розробників сайтів:

- Adobe ***Dreamweaver*** (http://www.adobe.com/cfusion/tdrc/index.cfm?product=dreamweav

er) — візуальний редактор для створення web-сайтів і додатків;

- ***Hotdog*** (http://www.sausage.com/hotdog-professional.html) — програма допоможе не тільки створити HTML-документ, але і програмувати на Flash, PHP, ASP, SQL; вміє оптимізувати і анімувати GIF-зображення (банери);
- ***HTMLPad*** (http://www.soft.necromancers.ru/prog_4764.html) — підтримує HTML, JavaScript, VBScript, SSI, ASP и Perl, вміє створювати макроси;
- ***PSPad*** (http://www.pspad.com/ru/download.php) — ще одна безкоштовна заміна Блокнота, яка "знає" синтаксис HTML і CSS;
- ***EditPlus*** (http://www.editplus.com) — ще один редактор, який підсвічує синтаксис HTML, PHP, Java, C/C++, CSS, ASP, Perl, JavaScript, VBScript.

Валідатори належать до класу програм, які допомагають у розробці сайтів. Це сервіси Інтернету, які перевіряють HTML-документ на наявність помилок і відповідність правилам обраного DOCTYPE. Найпопулярнішим валідатором вважається сервіс http://validator.w3.org . Якщо потрібно перевірити код CSS, то валідатор для цього доступний за адресою http://jigsaw.w3.org/css-validator/.

Використання тегів

У HTML все, що вводиться в документі, буде виводитися на екран браузера суцільним текстом, тобто переходи на новий рядок ігноруються; декілька пропусків підряд замінюються одним. Для форматування тексту використовують **теги**. Теги - це команди, які вказують браузеру, як саме потрібно вивести на екран розміщений всередині тега текст. Браузер ігнорує невідомий йому тег. Валідатор повідомляє про помилку, якщо виявить тег, що не належить обраній специфікації HTML.

Структура тега:

```
<ім'я тега атрибут1="знач1" атрибут2
                                    ="знач2"...>
```

Тег складається з імені тега, за яким може слідувати список атрибутів, розміщених між кутовими дужками (*наприклад*, <i>).

Атрибути керують поведінкою тега. Вони можуть мати конкретні значення, що задаються після знака рівності. Значення атрибутів беруться в лапки. Атрибути відокремлюються пробілом, порядок слідування атрибутів довільний. Імена тегів і атрибутів не чутливі до регістру, проте рекомендується використовувати нижній регістр. *Приклади*:

```
<font color="red" face="Arial">текст <b>
                жирний текст </b> текст
```

Типи тегів

- **теги верхнього рівня,** з яких складається базова структура документа – його каркас, тобто ті, що визначають розділ заголовка і тіла документа: <html>, <head>, <body>;
- **теги заголовка документа** — до цієї групи належать всі теги, розміщені всередині тега <head>: <title>, <meta>;
- **блокові елементи** інформують про структуру документа, їх використання призводить до появи нового рядка (будь-який тег, що слідує за блочним тегом, буде відображатися з нового рядка): <blockquote>, , <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <hr>, <p>, <pre>;
- **рядкові** (вбудовані) елементи містяться всередині блокових, вони охоплюють лише ними визначену частину тексту документа і не вимагають створення нового рядка: <a>, <b>, <big>, , <i>, <img>, <small>, , , <sub>, <sup>;
- **універсальні елементи** — теги цього типу можуть бути як блочними, так і рядковими: <del>, <ins>;
- **списки** — в цю групу тегів входять всі теги, які застосовуються для створення нумерованих або ненумерованих списків: <ul>, <ol>, <li>, <dd>, <dt>, <dl>;
- **таблиці** — теги, призначені для створення таблиць: <table>, <thead>, <tbody>, <td>, <th>, <tr>.
- **форми**

Найчастіше будемо використовувати блочні і рядкові теги.

Атрибути і їх значення

Всередині одинарного або відкриваючого тега можуть бути розміщені атрибути і їх значення:

```
<тег атрибут1="знач1" атрибут2="знач2">текст
                                        </тег>
```

Атрибути і їх значення змінюють функціональність того чи іншого тега. Як уже відмічалось, порядок розміщення атрибутів у тегу довільний, декілька атрибутів всередині одного тега розділяються пробілами. Атрибути можна переносити в новий рядок. Кожен атрибут повинен мати значення. Значенням атрибута можуть бути або певні ключові слова, або будь-який користувацький текст.

Лапки. Значення атрибута вказується в лапках після знака =. Лапки бувають одинарні ' або подвійні ". Якщо всередині одних лапок треба розмістити інші, слід використовувати лапки різного типу. *Наприклад*:

`атрибут="значе'нн'я"` або `атрибут='значе"нн"я'`.

Значеннями атрибутів можуть бути:

- **ключові слова** (*наприклад*, колір): `black` (чорний), `blue` (синій) тощо;
- **код кольору** `#ff0000` (червоний), `#000000` (чорний) і т. д.;
- **розмір** – дійсне число або ціле число від 1 до 100 у відсотках;
- **адреса** в інтернеті чи адреса ресурсу.

Приклади використання атрибутів:

```
<a href="http://www.wisdomweb.ru/">
wisdomweb.ru</a> <img src="mountimg.jpg"/>
```

За допомогою атрибута `href` елемента `<a>` вказується адреса документа, на який буде здійснюватися перехід після клацання мишею. За допомогою атрибута `src` елемента `<img>` можна вказати місце розташування у файловій системі малюнка, який буде вставлено в HTML-документ.

Список стандартних HTML-атрибутів

Перелічимо стандартні атрибути, притаманні практично всім HTML-елементам:

Атрибут	Опис
accesskey	Визначає сукупність клавіш для доступу до елемента
class	Визначає ім'я класу для елемента
id	Визначає унікальний ідентифікатор для елемента
style	Визначає стиль елемента
title	Містить додаткову інформацію про елемент (значен- ня цього атрибута відображається на екрані у вигляді підказки при наведенні курсора миші на елемент)

Макет web-сторінки

Як правило, кожна web-сторінка складається із стандартного набору областей, для яких існують спеціальні назви. Верхня частина сторінки називається **шапкою** або хедером (англ. header - заголовок). Зазвичай у ній розміщуються пункти головного меню, контактні дані, логотип, певний малюнок або слайд-шоу. Нижня частина сторінки називається **підвалом** або футером (англ. footer - нижній колонтитул). У ній розміщуються різноманітні лічильники, рейтинги, а також копірайт, посилання на розробників, контактні дані. Між шапкою і підвалом розміщується основна частина сторінки. Іноді зліва від основної частини сторінки розташовується окрема інформаційна смуга - ліва бічна панель. Так само справа від основної частини сторінки може розміщуватися права бічна панель.

Основну структуру HTML-документа можна створити з допомогою тегів `div`, як це показано в наступному HTML-коді:

```
<!DOCTYPE HTML>
<html>
<head>
<title> Заголовок </title>
<meta charset="UTF-8" />
```

```
</head>
<body>
<div>
<header>Шапка </header>
<div>Основна частина </div>
<footer>Підвал </footer>
</div>
</body>
</html>
```

Ми створили окремі контейнери для шапки, підвалу і основного вмісту. Окремі контейнери розміщені всередині батьківського контейнера `div` для зручності. Насамперед, батьківський контейнер потрібен для розташування вмісту посередині сторінки. Така верстка називається **фіксованою**. Детальніше про **фіксовану** і "**гумову"** верстку буде йти мова в розділі, присвяченому CSS.

Перегляд коду сторінки в інтернеті

Вміст будь-яких сторінок в інтернеті доступний для перегляду. Дослідження коду чужих сторінок допоможе початківцю швидше засвоїти HTML. Для того, щоб переглянути код, необхідно при перегляді сторінки натиснути праву кнопку миші і вибрати пункт "Переглянути HTML- код сторінки" (текст може незначно відрізнятися в різних браузерах).

Питання для самоконтролю

1. Основні можливості мови HTML.
2. Визначення HTML – документа.
3. Правила запису тегів в HTML – документі.
4. Яка різниця між парними та непарними тегами?
5. В чому полягає різниця між тегами рівня блоку та послідовними тегами?
6. Технологія створення HTML – документів.
7. Макет web –сторінки.

1.2. Робота з текстом. Заголовки. Списки

Форматування основного тексту HTML-сторінки

Безпосередньо всередині тега <body> не повинно бути ні тексту, ні зображень. За правилами будь-який текст чи зображення повинні розміщуватися всередині блокових чи рядкових тегів, списків або таблиць.

Розглянемо основні теги форматування.

Тег **** – призначений для відображення тексту напівжирним шрифтом:

```
<strong> Це напівжирний шрифт </strong>
```

Тег **** – призначений для відображення тексту курсивом.

Наприклад: `<em> Це курсив </em>`

Тег **<u>** – призначений для відображення тексту підкресленим:

```
<u> Це підкреслений текст </u>
```

Тег **<s>** – відображення тексту перекресленими символами.

Наприклад: `<s> Це перекреслені символи </s>`

Тег **<sub>** – відображення тексту у вигляді верхнього індекса.

Наприклад: `<sub> Верхній індекс </sub>`

Тег **<sup>** – відображення тексту у вигляді нижнього індекса.

Наприклад: `<sup> Нижній індекс </sup>`

Тег **<pre>** визначає блок попередньо відформатованого тексту. Все, що записано всередині тега <pre>, буде виводитися на екран браузера з усіма пробілами, табуляціями і перенесенням рядків. Цей тег зручно використовувати для вставки у HTML-документ рядків коду, написаного будь-якою мовою програмування.

Всі теги форматування тексту можуть бути вкладені один в інший. Це дозволяє реалізувати комбіноване виведення тексту. *Наприклад*, для виведеннння тексту курсивом і підкресленим, записуємо:

```
<i><u>  Приклад підкресленого курсиву
</u></i>
```

Порядок вкладення тегів один в інший значення не має. Попередній *приклад* можна записати і так:

```
<u><i> Приклад підкресленого курсиву </i></u>
```

Вигляд тексту у вікні браузера при використанні тегів форматування показано на рис. 1.1.

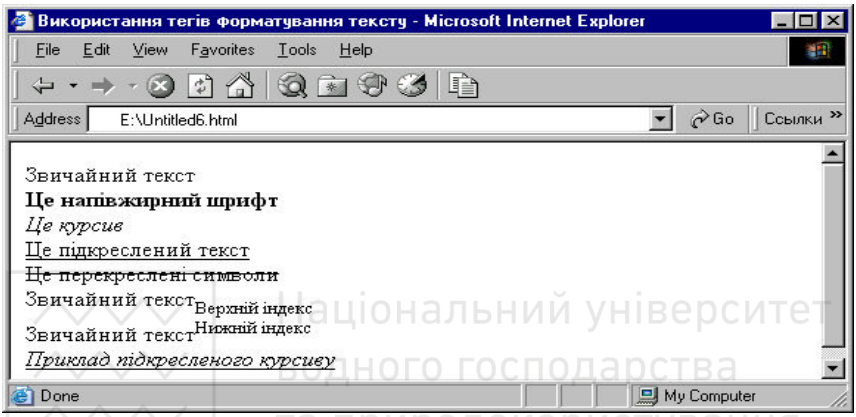


Рис. 1.1. Використання тегів форматування тексту

Тег **<font>** призначений для визначення параметрів шрифту: колір, розмір та гарнітура. Зміна шрифту встановлюється значеннями параметрів цього тега `color`, `size` та `face`. Параметр `color` дозволяє змінювати колір символів.

Наприклад: `<font color="green"> Цей текст зеленого кольору </font>`

Параметр `face` дозволяє задати гарнітуру шрифту, яким браузер буде виводити текст. Для цього значенням параметра слід вказати назву шрифту.

Наприклад:

```
<font face="Courier New"> Використовуємо шрифт CourierNew</font>
```

Параметр `size` дозволяє визначити розмір символів, який задається в умовних одиницях від 1 до 7 (результати на рис. 1.2).

Розмір символів при стандартних установках 3. Розмір символів можна вказувати як в абсолютних, так і у відносних величинах:

```
<font size="x"> Визначення символів в
    абсолютних величинах </font>
<font size="+x"> Визначення символів у
    відносних величинах </font>
<font size="-x"> Визначення символів у
    відносних величинах </font>
```

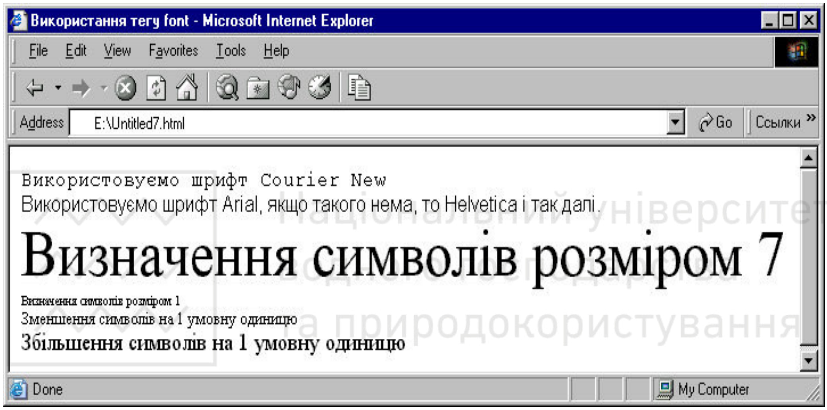


Рис. 1.2. Приклади використання параметрів `face` та `size` тега `<font>`

Тег **`<basefont>`** використовується для призначення параметрів шрифту, які будуть використовуватися як стандартні при перегляді HTML-сторінки. Параметри тега `<basefont>` ті ж самі, що і тега `<font>`. Зміна параметрів `<basefont>` поширюється на ту частину HTML-сторінки, яка розміщена нижче тега `<basefont>`. При сумісному використанні тегів `<basefont>` і `<font>` пріоритет має тег `<font>`. На відміну від `<font>` тег `<basefont>` не має закриваючого тега. Його дія поширюється до кінця HTML-сторінки або до ще одного тега `<basefont>`.

Наприклад:

```
<basefont size=4> Використання тега basefont
```

Відзначимо, що теги <font> та <basefont> належать до послідовних тегів, тому не можуть включати в себе теги рівня блоку, зокрема, <table> або <tr>.

Одинарний рядковий тег **<wbr>** дає можливість вказати місце, де дозволено робити перенесення рядка в тексті.

Тег **
** використовується для примусового перенесення рядка. Необхідність використання цього тега пояснюється тим, що при відображенні текстових документів перенесення тексту із одного рядка в інший відбувається автоматично та залежить в основному від розмірів шрифту та розмірів вікна браузера. При цьому перенесення рядка може здійснюватися тільки за символами, що розділяють окремі слова.

Розглянемо *приклад* запису виразів "Перший рядок" та "Другий рядок" в двох різних рядках. Відповідний HTML-код такий:

```
<body>
Перший рядок <br> Другий рядок
</body>
```

Парний тег **<nobr>** на відміну від тега
 використовується для заборони переведення рядка. *Наприклад*, для гарантованого розміщення в одному рядку речення "Цей текст повинен відображатись в одному рядку" необхідно записати:

```
<nobr> Цей текст повинен відображатись в
одному рядку </nobr>
```

Якщо цей текст не буде вміщуватись в одному рядку, то у вікні браузера з'явиться горизонтальна смуга прокрутки.

Тег **<p>** призначений для розділення тексту HTML-сторінки на окремі абзаци. Він є тегом–контейнером, може використовуватися з необов'язковим параметром `align`. Значення параметра `align` та наслідки його використання для абзаців ті ж самі, що і для заголовків (див. таблицю 1.3). Візуально у вікні браузера абзац буде відділятися від іншого тексту за допомогою відступу. Величина відступу встановлюється засобами CSS.

Приклад використання тега <p>:

```
<body> Звичайний текст
<p align="center"> Абзац з горизонтальним
вирівнюванням "до центру" </p>
</body>
```

Тег **<hr>** призначений для виведення у вікні браузера горизонтальної лінії, до і після якої вставляються порожні рядки. Необов'язкові параметри тега <hr> та їх призначення наведені в таблиці 1.2.

Таблиця 1.2.

Параметри тега <hr>

Параметр	Призначення
align	Визначення горизонтального вирівнювання лінії. Можливі значення: left ("до лівого краю"), center ("до центру"), right ("до правого краю"). Якщо параметр align не використаний, то застосовується вирівнювання "до центру"
color	Встановлює колір лінії
noshed	Відміняє рельєфність лінії
size	Визначає товщину лінії
width	Встановлює довжину лінії

Наприклад, для визначення лінії червоного кольору товщиною 2 пікселі довжиною 400 пікселів без рельєфу можна записати:

```
<hr color="red" size=2 width=400 noshade>
```

Парний блочний тег <**dialog**> використовується для створення діалогу між співрозмовниками. Окремі репліки створюються за допомогою парних тегів <dt> і <dd>, розміщених всередині цього тега. Тег <dt> містить ім'я співрозмовника, а тег <dd> — його репліку:

```
<dialog>
<dt> Співрозмовник 1 </dt>
<dd> Доброго дня.</dd>
<dt> Співрозмовник 2 </dt>
<dd> І вам доброго дня. </dd>
</dialog>
```

Парний рядковий тег **<mark>** призначений для виділення важливої частини тексту: фрази, ідеї або думки. В браузерах *Google Chrome і Mozilla Firefox* текст, розташований всередині цього тега, відображається на жовтому тлі. В інших браузерах

текст не виділяється.

Парний рядковий тег **<time>** використовується в тексті для виділення дати та/або часу. Він може мати декілька атрибутів:

- `datetime` - містить дату і/або час для виділеного тексту;
- `pubdate` - включає в себе дату публікації документа.

Парний рядковий тег **<meter>** дозволяє виділяти в тексті числа разом з одиницями вимірювання. При цьому атрибути тега `<meter>` дозволяють описати дане число:

- `value` - містить саме число;
- `min` - визначає можливий мінімум для даного числа, якщо число може перебувати тільки в певному діапазоні;
- `low` - граничне значення, при досягненні якого число вважається малим;
- `hight` - граничне значення, при досягненні якого число вважається великим;
- `max` - визначає максимум для даного показника;
- `optimum` - визначає оптимальне значення для цього показника.

Зауваження: Тег `<meter>` не підтримується навіть 9-ою версією браузера Internet Explorer, але підтримується іншими популярними браузерами.

Парний рядковий тег **<progress>** показує динаміку виконання якогось завдання, тобто відображає число або фразу, яка описує, на якому етапі перебуває виконання тієї чи іншої задачі.

В подальшому динаміку можна змінювати за допомогою JavaScript. Атрибути цього тега дозволяють краще зрозуміти, наскільки близько до поставленої мети вдалося підійти:

- `value` - число, яке визначає поточний етап виконання завдання;
- `max` - число, при досягненні якого завдання буде виконано.

Зауваження: Тег `<progress>` підтримується 10-ю версією браузера *Internet Explorer*, а також іншими браузерами.

Заголовки

Перш ніж залишитися на сторінці і приступити до її прочитання, користувачі зазвичай швидко пробігають очима по її вмісту, перевіряючи, чи містить вона цікаву для них інформацію. Часто заголовки - це перше, на що вони звертають увагу, тому неправильне використання заголовків може призвести до втрати відвідувачів. Насамперед, заголовки повинні коротко і точно описувати вміст, який вони презентують. Найбільш важлива інформація сторінки повинна розташовуватися під заголовками більшого розміру, а найменш важлива під заголовками меншого розміру. Пошукові системи також використовують заголовки для індексації структури та вмісту web-сторінок. Тому для того, щоб сторінка виводилася в пошуковій системі за відповідним запитом, слід використовувати змістовні заголовки.

Теги заголовків (**<h1>, <h2>, <h3>, <h4>, <h5>, <h6>**) призначені для виділення на екрані користувача певних фрагментів тексту HTML-сторінки. Виділення тексту реалізується за рахунок зміни розмірів та "жирності" тексту. Тегові <h1> відповідає найбільший розмір тексту, а <h6> – найменший розмір. Особливістю цих тегів є вставка відступу до і після виділеного фрагмента тексту.

Парний тег **<hgroup>** дозволяє групувати теги заголовків (від <h1> до <h6>) web-сторінки або розділу. *Наприклад*, для

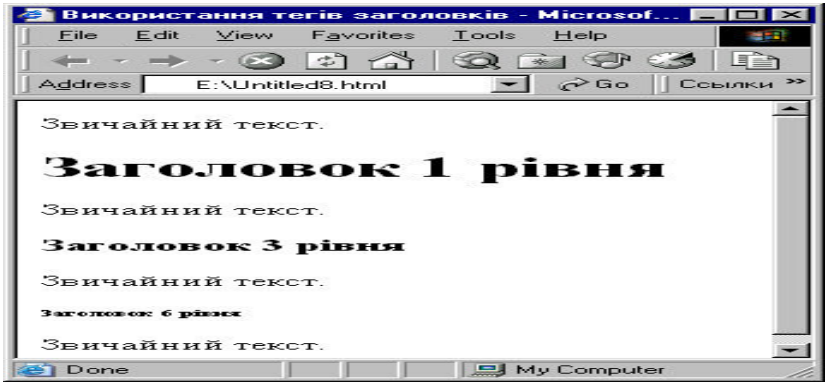


Рис. 1.3. Використання тегів заголовків

відображення, показаного на рис. 1.3, можна записати:

```
<body>
Звичайний текст. <h1> Заголовок 1 рівня </h1>
Звичайний текст. <h3> Заголовок 3 рівня </h3>
Звичайний текст. <h6> Заголовок 6 рівня </h6>
Звичайний текст.
</body>
```

Вказані теги заголовків є парними і можуть використовуватися з необов'язковим параметром `align`, який служить для горизонтального вирівнювання тексту. Можливі значення параметра `align` показані в таблиці 1.3.

Таблиця 1.3

Значення параметра `align`

Значення параметра	**Наслідки використання**
`left`	Текст вирівнюється до лівого краю вікна браузера
`center`	Текст вирівнюється до центру вікна браузера
`right`	Текст вирівнюється до правого краю вікна браузера
`justify`	Текст вирівнюється за шириною вікна браузера

Якщо параметр `align` не заданий, то використовується вирівнювання заголовка до центру вікна браузера. Розглянемо *приклад* визначення на HTML-сторінці чотирьох заголовків третього рівня з різними параметрами вирівнювання (результат подано на рис. 1.4).

```
<body> Звичайний текст.
<h3 align="left"> Заголовок вирівнюється до лівого краю вікна браузера</h3>
Звичайний текст.
<h3 align="center"> Заголовок вирівнюється до центру вікна браузера</h3>
Звичайний текст.
<h3 align="right"> Заголовок вирівнюється до правого краю вікна браузера</h3>
Звичайний текст.
```

```
<h3 align="justify"> Заголовок вирівнюється по
ширині вікна.
   Візуально відрізняється від left тільки в тому
випадку, коли текст заголовка містить більш ніж
два рядки тексту. </h3>
   Звичайний текст. </body>
```

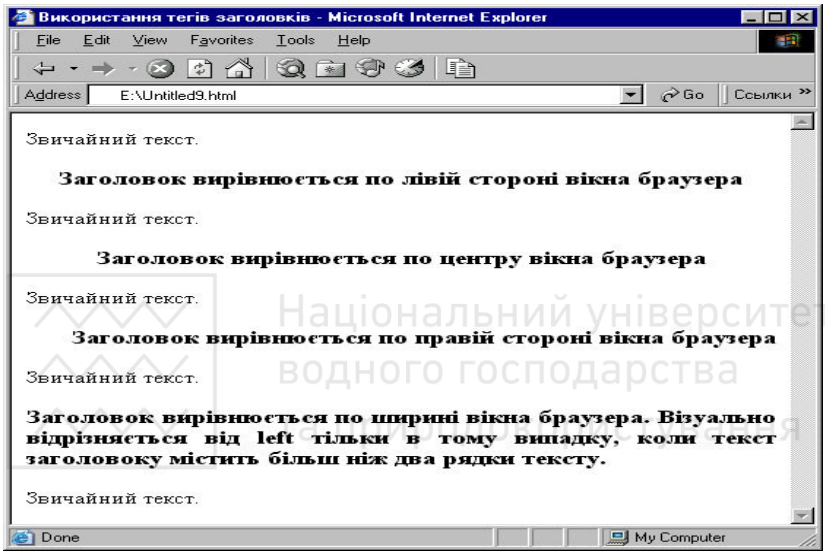


Рис. 1.4. Використання тега заголовка 3-го рівня з різними параметрами вирівнювання

Оформлення HTML-документів за допомогою CSS

Пакет **CSS** (англ. Cascading Style Sheets - каскадні таблиці стилів) був запропонований майже одночасно з HTML4. Він розширює можливості оформлення HTML-документів за рахунок того, що **дозволяє встановлювати певні характеристики відразу для великих масивів даних**, наприклад, для всіх сторінок сайту:

- розмір, накреслення та колір шрифту;
- розташування елементів;
- фон для елементів;
- вирівнювання текстів;
- оформення таблиць, списків тощо.

Детальніше моживості CSS будуть розглянуті в розділі 2. Тут лише відмітимо, що після введення CSS деякі теги і атрибути HTML стали вважатися застарілими, адже за допомогою CSS оформлення документів здійснюється більш ефективно. Оскільки вони не будуть підтримуватися у майбутніх версіях HTML і XHTML, то не варто використовувати такі **застарілі HTML-теги і атрибути:**

Теги	Опис
`<strike>`	закреслений текст
`<font> і <basefont>`	шрифт для тексту
`<center>`	вирівнювання вмісту до середини
`<menu>`	список меню

Атрибути	Опис
`color`	колір тексту
`bgcolor`	колір фону
`align`	вирівнювання тексту

Використання коментарів та спеціальних символів

В деяких випадках необхідно не показувати у вікні браузера частину HTML-коду сторінки, тобто "закоментувати" цей код. Для цього використовуються теги коментарів `<!-- та -->`

Наприклад:

```
<!-- Цей текст не буде відображатись у вікні браузера -->
```

Символи, які неможливо відобразити в HTML-документах звичайним чином, називаються **спецсимволами**. Спецсимвол можна відобразити тільки використовуючи відповідну йому мнемоніку або спеціальний код. Із спецсимволів найчастіше використовується “нерозривний пробіл” (його мнемоніка ` `). Цей символ використовується для контролю перенесення рядка (після цього символа автоматичне перенесення рядка неможливе) і для вставлення в текст пробілів, що йдуть поспіль (за замовчуванням, якщо вставити у текст послідовність з 5-ти пробілів, браузер виріже 4 і відобразить лише один). Також у HTML-коді часто

використовуються спецсимволи < (мнемоніка `<`) і > (мнемоніка `>`). Самі ці символи неможливо безпосередньо використовувати в коді, оскільки браузер "плутає" їх з кутовими дужками.

Як відомо, при створенні HTML-сторінки повинні застосовуватися тільки ті символи, що входять до базової частини таблиці кодів ASCII. Проте деякі спеціальні символи не входять до неї. Наведемо мнемоніки спецсимволів, які застосовуються часто:

Спецсимвол	Мнемоніка
&	&amp;
© (знак copyright)	&copy;
€	&euro;
>	&gt;
←	&larr;
<	&lt;
пробіл	
→	&rarr;
®	&reg;
§	&sect;
™	&trade;

Списки

Списки – поширена форма оформлення даних як в електронних, так і в друкованих документах. Мовою HTML передбачено використання трьох стандартних видів списків: маркованого, нумерованого та списку визначень. Відзначимо, що від інших елементів на HTML-сторінці стандартні списки відділяються відступами.

Марковані списки

Для визначення маркованого списку використовується тег-контейнер **<ul>**, в якому розміщуються всі елементи списку. При цьому кожен пункт списку повинен починатися тегом **<li>.**

Наведемо *приклад* HTML-коду маркованого списку, відображення якого показане на рис. 1.5:

```
<body> Звичайний текст
```

```
<ul>
<li> Перший пункт маркованого списку
<li> Другий пункт маркованого списку
<li> Третій пункт маркованого списку
</ul>
Звичайний текст
</body>
```

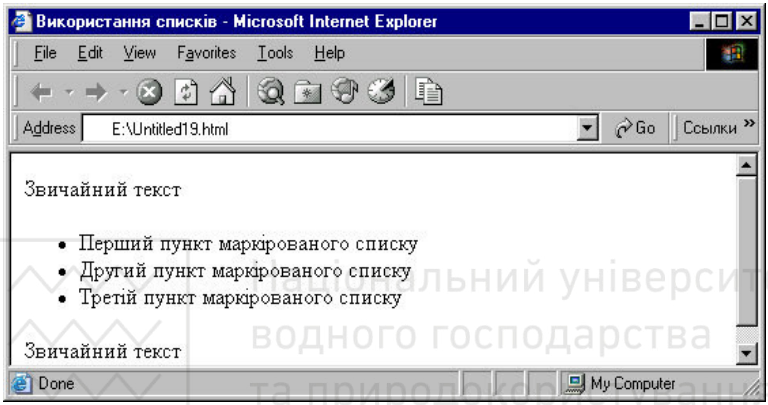


Рис. 1.5. Відображення маркованого списку

Як в тегу `<ul>`, так і в тегу `<li>` можна використовувати необов'язковий параметр **`type`**, за допомогою якого визначається тип маркера списку. Можливі значення цього параметра:

- **`disc`** – маркери відображаються заповненими колами;
- **`circle`** – маркери відображаються незаповненими колами;
- **`square`** – маркери відображаються заповненими квадратами.

Приклад використання різних значень параметра `type` в тегу `<li>` маркованого списку (відображення у вікні браузера на рис. 1.6):

```
<ul>
<li type="disc"> Перший пункт маркованого
списку. type="disc"
```

```
<li type="square"> Другий пункт маркованого
списку. type="square"
  <li type="circle"> Третій пункт маркованого
списку. type="circle"
  </ul>
```

Якщо параметр `type` в тегах `<ul>` та `<li>` не використовується, то за замовчуванням для всього списку використовується маркер типу "**`disc`**".

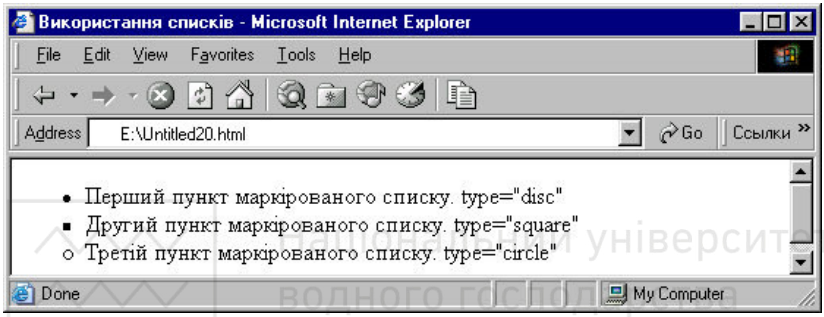


Рис. 1.6. Використання різних значень параметра `type` тегу

Нумеровані списки

Для визначення нумерованого списку використовується тег-контейнер **`<ol>`**, всередині якого розміщуються всі елементи списку. При цьому кожен пункт списку повинен починатись тегом **`<li>`**. Наведемо *приклад* HTML-коду нумерованого списку, відображення якого у вікні браузера показане на рис. 1.7:

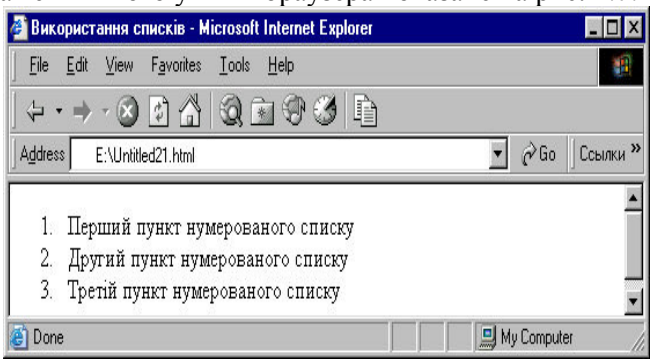


Рис. 1.7. Відображення нумерованого списку

```
<ol>
<li> Перший пункт нумерованого списку
<li> Другий пункт нумерованого списку
<li> Третій пункт нумерованого списку
</ol>
```

Часто тег <ol> використовується з необов'язковими параметрами type та start. При цьому параметр type використовується також в тегу <li>. Його завданням є визначення одного із стандартних типів маркера. Для нумерованого списку існує п'ять стандартних типів маркерів. Використання цього параметра в тегу <ol> поширюється на весь список, а використання в тегу <li> - тільки на поточний пункт списку. Можливі значення параметра type та відповідні цим значенням стандартні типи маркерів показані в таблиці 1.4.

Таблиця 1.4.

Використання параметра type

Значення параметра type	**Тип маркера**
1	у вигляді арабських цифр
A	у вигляді великих букв латинського алфавіту
a	у вигляді малих букв латинського алфавіту
I	у вигляді великих римських цифр
i	у вигляді малих римських цифр

Наведемо *приклад* нумерованого списку з маркерами у вигляді великих римських цифр. При цьому в другому пункті списку змінимо тип маркера на "великі латинські букви". Вигляд списку у вікні браузера показано на рис. 1.8.

```
<ol type="I">
<li> Перший пункт нумерованого списку.
<li type="A"> Другй пункт нумерованого
списку. type="A"
<li> Третій пункт нумерованого списку.
</ol>
```

Нумерація пунктів нумерованого списку не залежить від типу маркера. Якщо параметр `type` в тегах `<ol>` та `<li>` не зазначено, для всього списку використовується маркер типу "1".

Параметр `start` призначений для зміни початку нумерації пунктів списку і може використовуватись тільки в тегу `<ol>`.

Для зміни нумерації пунктів всередині списку в тегу `<li>` використовується параметр `value`. Значеннями параметрів `start` та `value` можуть бути тільки натуральні числа. Наведемо *приклад* визначення нумерованого списку, в якому нумерація пунктів починається з п'яти, а в третьому рядку списку початок нумерації змінюється на десять (відображення такого списку показане на рис. 1.9):

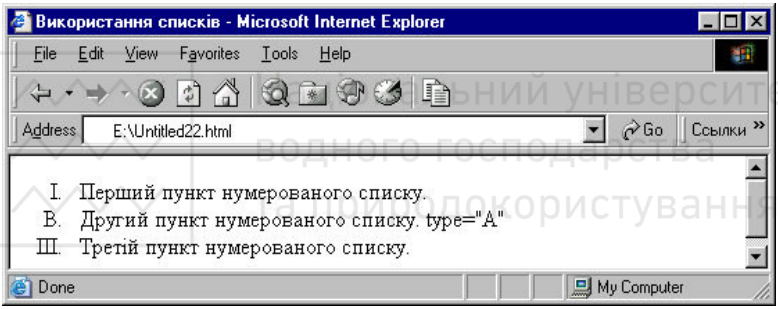


Рис. 1.8. Зміна типу маркерів в нумерованому списку

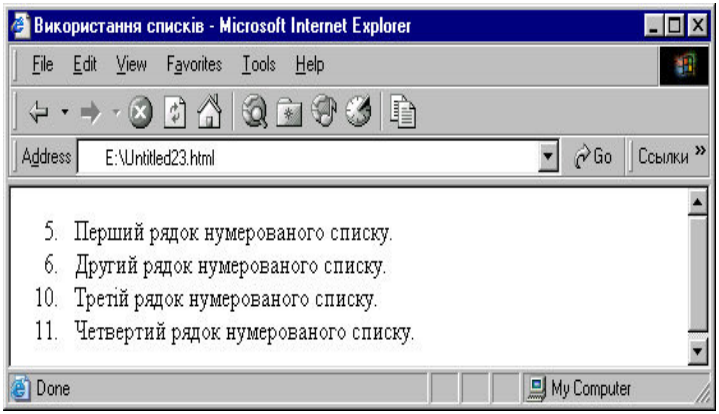


Рис. 1.9. Зміна початку нумерації пунктів нумерованого списку

```
<ol start="5">
<li> Перший рядок нумерованого списку.
<li> Другий рядок нумерованого списку.
<li value="10"> Третій рядок нумерованого
списку.
<li> Четвертий рядок нумерованого списку.
</ol>
```

Крім звичайних списків можна створювати вкладені списки. *Приклад*:

```
<html>
<body>
<p> Найбільші міста різних країн (приклад
вкладених списків): </p>
<ol>
<li> Україна
<ol>
<li> Київ </li>
<li> Харків </li>
</ol>
</li>
<li> США
<ol>
<li> Нью-Йорк </li>
<li> Лос-Анжелес </li>
</ol>
</li>
</ol>
</body>
</html>
```


Списки визначень

Список визначень використовується для розміщення на HTML-сторінці тексту, подібного до енциклопедії чи словника. Кожен пункт такого списку складається із двох частин: в першій частині записується термін, що потребує визначення, а в другій - текст, що пояснює зміст терміну.

Список визначень задається за допомогою тега-контейнера **<dl>.** Для визначення терміну використовується тег **<dt>**, а для визначення пояснення - тег **<dd>**. Наведемо *приклад* списку визначень, відображення якого показано на рис. 1.10:

```
<dl>
<dt>Інтернет (мережа Інтернет)
<dd>Сукупність мереж та обчислювальних засобів, які використовують стек протоколів TCP/IP (Transport Control Protocol/Internet Protocol), спільний простір імен та адрес для забезпечення доступу до інформаційних ресурсів мережі будь-якій особі;
<dt>Гіпертекстове посилання
<dd>Адреса іншого мережевого інформаційного ресурсу у форматі URL, який тематично, логічно або будь-яким іншим способом пов'язаний з документом, у якому це посилання визначене.
</dl>
```

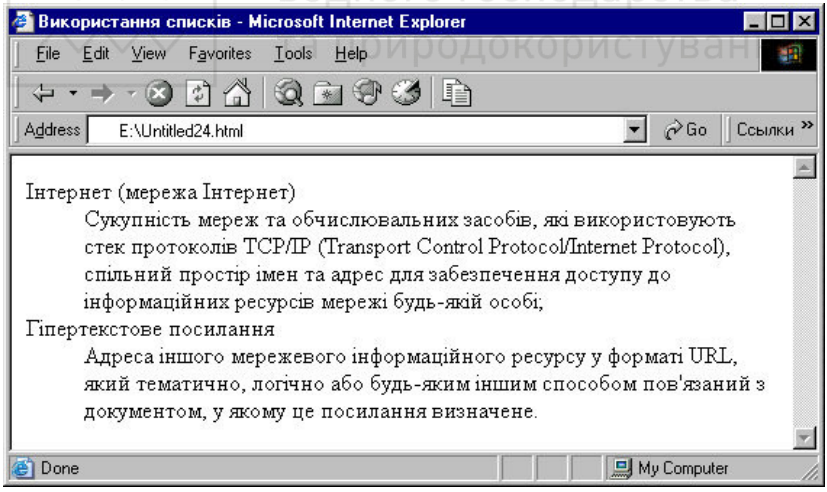


Рис. 1.10. Список визначень

Як видно з рисунка 1.10, у вікні браузера термін відділяється від свого тлумачення за допомогою переведення рядка та табуляції.

Питання для самоконтролю

1. Блокові теги HTML.
2. Рядкові теги HTML.
3. Параметри тега <font>.
4. Перенесення тексту на веб-сторінці у наступний рядок.
5. Вирівнювання текстів.
6. Застарілі теги HTML.
7. Мнемоніки.
8. Організація списків HTML.

1.3. Зображення. Звук. Відео

Розміщення зображень у HTML-документах

Комбінуючи зображення і звичайний текст, можна збільшити ефективність подання інформації. Вставити зображення у HTML-сторінку можна за допомогою тега **<img>**. В обов'язковому атрибуті цього тега **src** вказується адреса файлу, в якому зберігається рисунок. Цей же тег може служити для показу відео-роликів. В цьому випадку використовується параметр **dynsrc**, за допомогою якого вказується адреса відеофайлу. Найпоширенішими форматами файлів-рисунків є формати **jpg, gif, png,** а відеофайлів - **avi**. Завантаження зображень займає час, тому не варто планувати розміщення на одній сторінці їх великої кількості.

Адреса файлу–рисунка може бути вказана в абсолютному або відносному форматі. При використанні абсолютного формату в адресі повністю визначено комп'ютер, каталог та файл рисунка. *Наприклад*, для вставлення рисунка з файлу *logo.gif* за адресою *www.picture.com.ua/picture* необхідно записати:
<img src="www.picture.com.ua/picture/logo.gif">

При використанні відносного формату визначення місцезнаходження файлу-рисунка відбувається з урахуванням розміщення даної HTML-сторінки.

Зокрема, для вставки в HTML-сторінку *prim.html* рисунка, який зберігається у файлі *logo.gif,* що розміщений в одній папці з файлом *prim.html,* записуємо: <img src="logo.gif">

Графічні файли зручно зберігати в окремій папці, *наприклад*, в папці з ім'ям *img*. В цьому випадку для вставлення в HTML-сторінку *prim.html* рисунка *logo.gif* слід записати:

```
<img src="img/logo.gif">
```

Відзначимо, що каталог *img* та файл *prim.html* повинні міститися в одній папці. Крім обов'язкового параметра `src` тег `<img>` має декілька необов'язкових параметрів (табл. 1.5).

Таблиця 1.5.

Параметри тега **<img>**

Назва параметра	Призначення
`alt`	Задання альтернативного тексту
`width`	Встановлення ширини рисунка в пікселях
`height`	Встановлення висоти рисунка в пікселях
`hspace`	Визначення відступу від рисунка до інших об'єктів на HTML-сторінці по вертикалі
`vspace`	Визначення відступу від рисунка до інших об'єктів на HTML-сторінці по горизонталі
`border`	Вибір товщини рамки навколо рисунка
`lowsrc`	Вибір файлу з альтернативним зображенням
`align`	Сбосіб вирівнювання рисунка відносно інших об'єктів
`src`	Адреса графічного файлу
`dynsrc`	Адреса відеофайлу
`start`	Визначення моменту початку прокрутки відеоролика. Можливі значення: • `fileopen` – ролик починається відразу після завантаження відеофайлу; • `mouseover` – ролик починається при наведенні миші на зображення
`loop`	Призначення кількості повторів відеоролика. При значенні –1 програвання відбувається безперервно

При перегляді HTML-сторінок користувач може використовувати браузер, що працює в режимі відключення завантаження зображень (такий режим зменшує час відображення HTML-сторінки у вікні браузера). В цьому випадку замість зображення на екрані з'являється альтернативний текст, заданий у параметрі **alt**. Альтернативний текст відображається як підказка при наведенні курсора миші на рисунок. Крім того, альтернативний текст використовується пошуковими роботами.

Приклад використання параметра `alt`*:*

```
<img src="logo.gif" alt="Альтернативний текст" >
```

Параметри **width** та **height** призначені для визначення розмірів рисунка у вікні браузера. При зміні розмірів зображення розмір файлу не змінюється, як і час завантаження рисунка. Наведемо *приклад* застосування параметрів `alt`, `width` та `height` для одного і того ж рисунка (результат показано на рис. 1.11).

```
  <img src="logo.gif" width="184" height ="164"
alt="Альтернативний текст" >
  <img src="logo.gif" width="92" height ="82"
alt="Альтернативний текст" >
  <img src="logo.gif" width="150" height ="40"
alt="Альтернативний текст" >
```

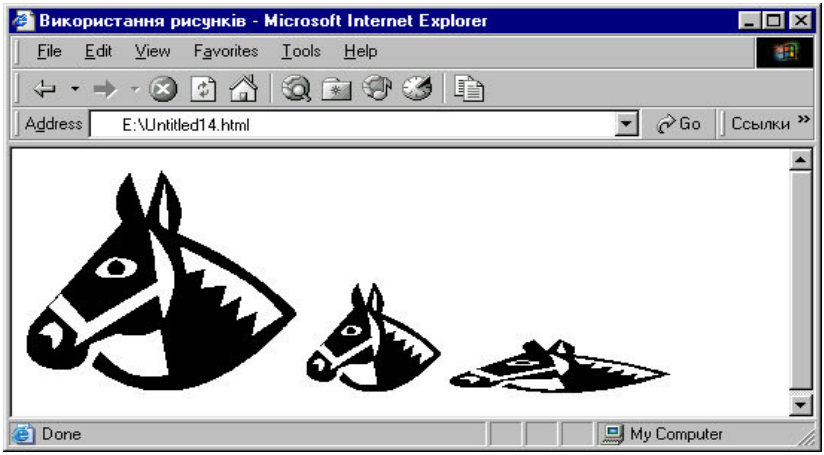


Рис. 1.11. Використання параметрів `alt`, `width` та `height` тега `<img>`

Параметр **border** дозволяє визначити товщину рамки навколо рисунка. Застосуємо цей параметр із значенням 2 до попереднього *прикладу* (рис. 1.12):

```
<img src="logo.gif" width="184" height ="164"
border="2">
  <img src="logo.gif" width=" 92" height =" 82"
border="2">
  <img src="logo.gif" width="150" height ="40"
border="2">
```

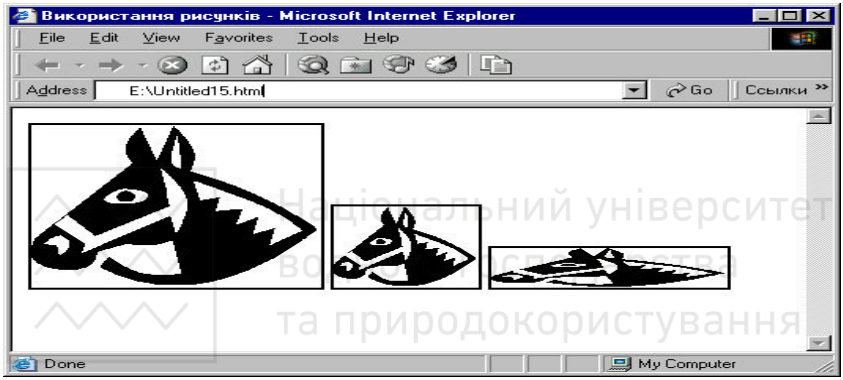


Рис. 1.12. Використання параметрів `border`, `width` та `height` тега `<img>`

Параметри **hspace** та **vspace** дозволяють у пікселях задавати відступи від рисунка до інших об'єктів на HTML-сторінці. Покажемо застосування цих параметрів із значеннями 20 та 20 до попереднього *прикладу* (рис. 1.13):

```
  <img src="logo.gif" width="184" height="164"
hspace="20" vspace="20" border="2">
  <img src="logo.gif" width="92" height="82"
hspace="20" vspace="20" border="2">
  <img src="logo.gif" width="150" height="40"
hspace="20" vspace="20" border="2">
```

Параметр **lowsrc** дозволяє визначити файл з альтернативним зображенням, яке з'являється у вікні браузера до появи основного зображення. Альтернативне зображення, як правило, це рисунок того ж змісту, що і основний, але менш якісний. З цієї причини файл

альтернативного зображення менший за розміром, а тому завантажується набагато швидше ніж основний. Це дозволяє швидше показати загальні риси зображення.

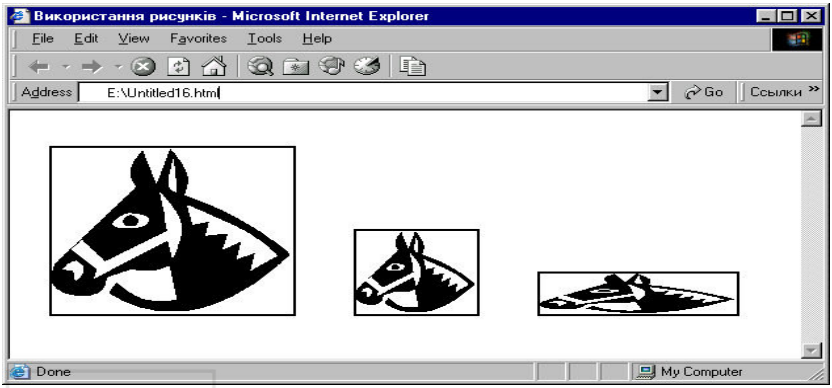


Рис. 1.13. Використання параметрів `hspace`, `vspace`, `border`, `width` та `height` тега `<img>`

Приклад використання:

```
<img src="logo.gif" width="184" height="164"
lowsrc ="small.gif">
```

Параметр **`align`** використовується для визначення горизонтального або вертикального вирівнювання рисунка відносно інших об'єктів на HTML-сторінці. Параметри вирівнювання залежать від значень параметра `align`. При використанні цього параметра із значеннями **`left`** або **`right`** рисунок розміщується відповідно до лівої або правої сторони вікна браузера, а текст "обтікає" зображення. В цьому випадку текст може займати декілька рядків.

Приклад використання параметра `align` (результат на рис. 1.14):

```
<body>
<img src="logo.gif" width="182" height="162"
border="1" align="right">
Приклад використання параметра align="right".
Рисунок розміщений біля правої сторони вікна
браузера. Текст обтікає зображення з лівої
сторони.
</body>
```

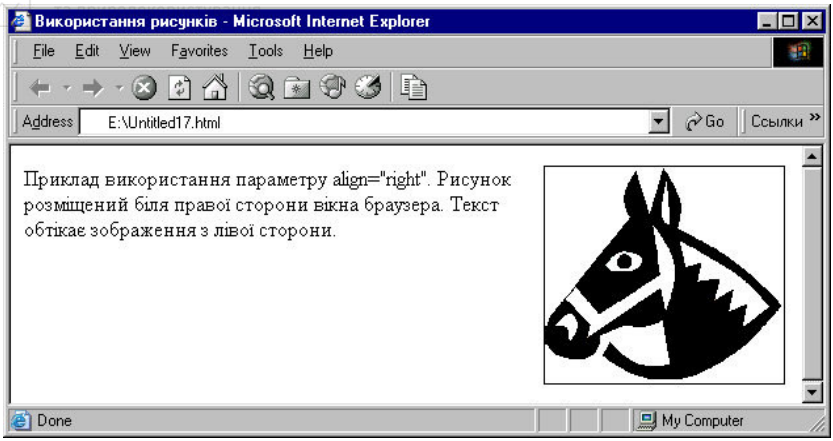


Рис. 1.14. Приклад вирівнювання рисунка до правого краю вікна браузера

Крім **left** та **right** параметр **aling** може набувати таких значень:

- **top** – верхня межа зображення вирівнюється за найвищим елементом рядка;
- **texttop** – верхня межа зображення вирівнюється за найвищим текстовим елементом рядка;
- **middle** – середина зображення вирівнюється за базовою лінією рядка;
- **absmiddle** – середина зображення вирівнюється за серединою рядка;
- **bottom** – нижня межа зображення вирівнюється за базовою лінією рядка;
- **absbottom** – нижня межа зображення вирівнюється за нижньою межею рядка.

В цих випадках зображення можна розглядати як звичайний елемент рядка.

Відзначимо, що базова лінія тексту – це нижня лінія рядка тексту без врахування нижньої частини деяких символів, зокрема, `p, j, y`.

Наведемо *приклад* використання параметра `align` тега <`img`> із значеннями `top`, `middle` та `bottom`:

```
<body>
  <img src="logo.gif" alt="" width="91"
height="81" border="1" align="top">
  Приклад використання параметра align="top".
<br>
  <img src="logo.gif" alt="" width="91"
height="81" border="1" align="middle">
  Приклад використання параметра align="middle".
<br>
  <img src="logo.gif" alt="" width="91"
height="81" border="1" align="bottom"> Приклад
використання параметра align="bottom".
  </body>
```

Змістові й фонові зображення у web

Існує два основних способи додавання зображень у документ: змістові зображення за допомогою елемента **<img>** і фонові зображення, що застосовуються до елементів засобами CSS. Коли які зображення використовувати?

1. Якщо зображення є критично важливим для змісту документа, *наприклад*, фотографія автора або графік, що показує певні дані, то його треба додати як елемент <img> з відповідним альтернативним текстом.

2. Якщо зображення служить як "прикраса", можна використовувати фонові зображення CSS. Такі зображення не супроводжуються альтернативним текстом, крім того, CSS має більше можливостей для оформлення зображення.

Парний блоковий тег **<figure>** використовується для створення зображення з підписом. Саме зображення визначається за допомогою тега **<img>**, розташованого всередині цього тега, а підпис задається за допомогою тега **<legend>**:

```
  <figure>
  <legend> Зображення, яке змінить ваше життя
</legend>
  <img alt="" src="/img/pict.png" />
  </figure>
```

Цей тег може використовуватися для групування будь-яких даних, а не тільки зображення та підпису.

Парний тег **<figcaption>** дозволяє задати опис групи, створеної тегом **<figure>**. Відповідно, він повинен знаходитися усередині тега **<figure>**. Причому він повинен бути або першим, або останнім тегом всередині тега **<figure>**:

```
<figure>
<figcaption> Опис </figcaption>
<legend> Зображення, яке змінить ваше життя
</legend>
<img alt="" src="/img/pict.png" />
</figure>
```

Створення альтернативного тексту за допомогою `alt`

Елемент **<img>** повинен мати атрибут **alt**. Цей атрибут містить альтернативний текст, що виводиться, якщо з певних причин зображення недоступне. Щоб зробити зображення зрозумілішим, слід додати альтернативний текст відповідного змісту.

Інформація в атрибуті `alt` не повинна виводитися, коли зображення було успішно завантажене й показане. Однак Internet Explorer показує її як спливаючу підказку, коли покажчик миші потрапляє на зображення. Якщо необхідно додати інформацію, замість атрибута `alt` використовують атрибут `title`, до розгляду якого ми переходимо далі.

Додавання інформації за допомогою атрибута `title`

Більшість браузерів виводять значення атрибута **title** елемента `<img>` як спливаючу підказку, коли на зображення потрапляє вказівник миші (див. рис. 1.15). Це може допомогти відвідувачу Web-сторінки більше довідатися про зображення. Атрибут `title` може запропонувати зміст або опис зображення. Атрибут `title` можна використовувати і в інших елементах, якщо необхідно надати їм спливаючу підказку.

Використання `longdesc` для опису складних зображень

Якщо зображення є складним (*наприклад*, графік), можна запропонувати його детальний опис, використовуючи **longdesc**, щоб навіть при відключеному виведенні зображень

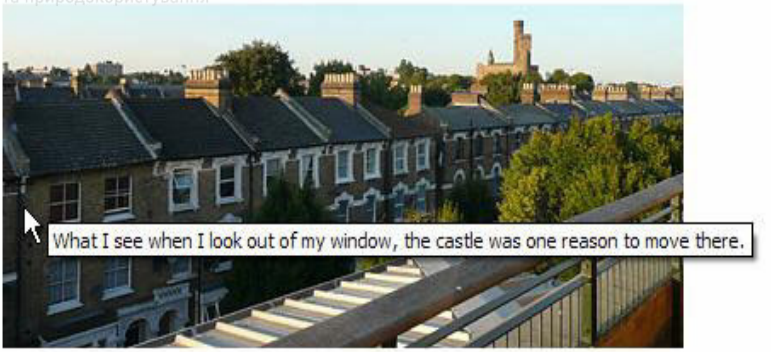


Рис. 1.15. Атрибут `title` виводиться як спливаюча підказка в багатьох браузерах.

можна було дістати інформацію щодо його вмісту.

Цей атрибут містить URL, що вказує на документ, де розміщена відповідна інформація. *Наприклад*, діаграму, яка відображає залежність даних, можна з'єднати з таблицею цих же даних за допомогою **`longdesc`**.

На рис. 1.16 показано виведення двох різних подань даних.

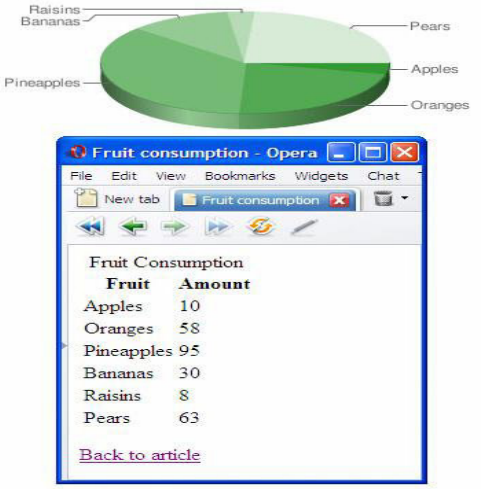


Рис. 1.16. З'єднання документа, що містить дані, з зображенням за допомогою атрибута `longdesc`

Фонове зображення на сторінці

Адресу фонового зображення для сторінки можна задати в атрибуті `background` тега `<body>`. Фонове зображення відображається в натуральну величину. Якщо розмір зображення менший, ніж размір вікна браузера, то рисунок повторюється по горизонталі вправо і по вертикалі вниз.

Наприклад, задамо фоновим зображенням сторінки рисунок *bg1.jpg*:

Рис.1.17. Файл *bg1.jpg*

Фрагмент коду матиме вигляд:

```
<html><head><title>Тест фону</title></head>
<body background="bg1.jpg"></body></html>
```

Таблиця 1.6.

Стандартні кольори HTML

Назва в HTML	Назва українською	Код в RGB
aqua	морська хвиля	#00ffff
black	чорний	#000000
blue	синій	#0000ff
rey	сірий	#808080
green	зелений	#008000
lime	яскраво-зелений	#00ff00
maroon	темно-бордовий	#800000
navy	темно-синій	#000080
olive	оливковий	#808000
purple	пурпуровий	#800080
red	червоний	#ff0000
silver	сріблястий	#c0c0c0
teal	бірюзовий	#00808
white	білий	#ffffff
yellow	жовтий	#ffff00

Значення кольору (коди стандартних кольорів наведені в таблиці 1.6) вказується в тегу після символа #. *Наприклад*:

```
<font color="#808080"> сірий текст</font>
```

Для фону всієї сторінки в тегу **<**body**>** використовують артибут bgcolor:

```
<body bgcolor="# FFFF00"> жовтий фон </body>
```

Таблиця 1.7.

Вибір формату графічного файлу

Характеристики зображення	**Обраний формат**
анімоване зображення	тільки GIF
маленьке зображення з невеликою кількістю кольорів	GIF або PNG-8
зображення з напівпрозорістю	тільки PNG
зображення з великою кількістю кольорів, *наприклад*, фотографія	JPEG
зображення з великою кількістю кольорів з дрібними деталями, *наприклад*, скріншот (знімок екрана)	PNG-24

Бажано уникати використання інших форматів зображень (*наприклад*, BMP або TIFF), оскільки вони можуть не підтримуватися окремими типами браузерів.

Створення карт-зображень

В HTML можна створювати спеціальні карти-зображення, які містять список областей-посилань і прив'язуються до зображень. Карти в HTML створюються за допомогою тега <**map**>, а області-посилання на них з допомогою тега <**area**>. Форма і координати областей-посилань задаються з допомогою атрибутів **shape** і **coords** тега **<area>**, а місце, на яке вони посилаються, - за допомогою атрибута **href**. Можливі форми областей посилань:

Форма	Атрибути
Прямокутна область-посилання	**shape="rect", coords="x1,y1,x2,y2"**, де **x1,y1** - координати верхнього лівого кута прямокутної області, а **x2,y2** - координати нижнього правого кута області
Кругла область-посилання	**shape="circle", coords="x,y,радіус"**, де **x,y** - координати центра області
Область-посилання многокутник	**shape="poly", coords="x1,y1,x2,y2..xz,yz"**, де **x1,y1** - координати першого кута многокутника, **x2,y2** - другого і т.д. Якщо координати першого і останнього кута не збігаються, браузер автоматично додасть координати останнього кута, щоб завершити многокутник

Після того, як карта створена, необхідно “прив’язати” її до якогось зображення. Для цього необхідно додати до зображення атрибут **usemap**="#ім’я_карти", а до карти додати атрибут **name**="ім’я_карти".

HTML5 Canvas

За допомогою парного блочного тега **<canvas>** можна засобами JavaScript створювати малюнки, анімацію, ігри. Він дозволяє малювати різні зображення або ж виводити вже створені зображення, при цьому трансформуючи їх або змінюючи їх властивості. Атрибути **`width`** і **`height`** цього тега дають можливість задати висоту і ширину створюваного полотна. Але найбільш важливий для тега **`canvas`** атрибут **`id`**. Він дозволяє задати ідентифікатор, який надалі буде використовуватися JavaScript для роботи з цим тегом.

Вміст, розміщений між тегами, буде відображатися, якщо браузер користувача не підтримує елемент **`canvas`**:

```
<canvas id='draw' width='300' height='200'>
Ви бачите це повідомлення, тому що Ваш браузер
не підтримує canvas! </canvas>
```

Зауважимо, що сам по собі **<canvas>** нічого не малює. Він являє собою полотно, яке надає можливості для малювання клієнтським скриптам.

Методи для малювання прямокутників

Метод	Опис
fillRect(x,y,ширина,висота)	Малює зафарбований прямокутник
strokeRect(x,y,ширина,висота)	Малює контур прямокутника
clearRect(x,y,ширина,висота)	Очищає вказану зону і робить її повністю прозорою

Зауваження: параметри **x** і **y** задають величину зсуву прямокутника по горизонталі **(x)** і вертикалі **(y)** від верхнього лівого кута полотна у пікселях.

Тепер спробуємо намалювати що-небудь на полотні.

Приклад:

```
<html>
<body>
<canvas id='draw' width='300' height='200'
style='border:1px solid'></canvas>
<script>
var canvas=document.getElementById("draw")
var x=canvas.getContext("2d");
x.fillRect(50,40,55,55);
x.strokeRect(150,70,55,55);
x.clearRect(68,57,20,20);
</script>
</body>
</html>
```

Пояснення прикладу:

`var canvas=document.getElementById("draw")` - знаходимо потрібне полотно;

`var x=canvas.getContext("2d")` - звертаємося до вбудованого об'єкта, який містить методи для малювання (перші два кроки є стандартними для малювання будь-якого об'єкта в canvas);

`x.fillRect(50,40,55,55)` - малюємо зафарбований прямокутник;

`x.strokeRect(150,70,55,55)` - малюємо не зафарбований прямокутник;

x.clearRect(68,57,20,20) - очищаємо зону в зафарбованому прямокутнику.

Малювання складених фігур

Складені фігури компонуються з декількох з'єднаних простих об'єктів (лінії, кола тощо).

```
beginPath();
/* Прості об'єкти розміщують тут */
closePath(); // Автоматично завершує фігуру (з'єднує кінцеву точку з початковою)
//Тепер необхідно викликати один з методів для малювання фігури, визначеної вище
stroke();  //намалює фігуру не зафарбованою
fill();    //намалює фігуру зафарбованою
```

Зауваження: метод `closePath()` може бути відсутнім.

Приклад:

```
<html>
<body>
<canvas id='draw' style='border:1px solid'
width='300' height='200'></canvas>
<script>
var canvas=document.getElementById("draw")
var x=canvas.getContext("2d");
x.beginPath();
x.moveTo(20,20);
x.lineTo(70,70);
x.lineTo(20,70);
x.closePath();
x.fill();
```

```
</script>
</body>
</html>
```

Методи для виведення простих об'єктів

Об’єкти	Опис
`moveTo(x,y)`	Встановлює координати точки, з якої почнеться малювання наступного об’єкта
`lineTo(x,y)`	Малює пряму лінію
`arc(x,y,радіус, поч_кут,кін_кут)`	Малює круг або його частину. Кут задається в радіанах (*радіани=(Math.PI/180)*градуси*)
`rect(x, y, ширина, висота)`	Малює прямокутник

Кольори

Намальовані за допомогою **<canvas>** об'єкти можна оформляти. Для розмальовування побудованих у **<canvas>** фігур передбачені властивості: **fillStyle** і **strokeStyle**. Властивість **fillStyle** використовується для застосування кольору до замальованої, а **strokeStyle** - для застосування кольору до не замальованої фігур.

Приклад:

```
<html>
<body>
<canvas id='draw' width='300' height='200'
style='border:1px solid'></canvas>
<script>
var canvas=document.getElementById("draw")
var x=canvas.getContext("2d");
x.fillStyle="green";
x.fillRect(10,40,65,65);
x.strokeStyle="#FF45FF";
```

```
x.strokeRect(100,40,65,65);
x.fillStyle="rgb(255,73,73)";
x.fillRect(190,40,65,65);
</script>
</body>
</html>
```

Як видно з попереднього прикладу, кольори можуть задаватися таким же способом, як і в CSS. Крім цих способів, в **<canvas>** колір і прозорість елемента може задаватися з використанням моделі RGB.

Оформлення ліній

В HTML5 є кілька властивостей для оформлення ліній, намальованих за допомогою методу **lineTo(): lineWidth, lineCap, lineJoin.** За допомогою властивості **lineWidth** можна встановити ширину лінії (за замовчуванням лінії мають ширину 1 піксель). Властивість **lineCap** дозволяє оформляти кінці лінії (заокруглені, стандартні тощо). За допомогою властивості **lineJoin** можна згладжувати стики двох ліній.

Текст

В **<canvas>** можуть бути вставлені зображення і відображений довільний текст. Метод **fillText("текст",x,y)** дозволяє відображати в елементі **<canvas>** довільний текст. Текст, що відображається в **<canvas>**, може бути оформлений за допомогою властивості **font**.

Приклад:

```
<html>
<body>
<canvas id="draw" width="300" height="160"
style="border:1px solid;"></canvas>
<script>
var canvas=document.getElementById("draw");
var x=canvas.getContext("2d");
x.font='15px Verdana';
```

```
  x.fillStyle='#60016d';
  x.fillText("Зараз Ви можете відображати",10,
40);
  x.font='25px Arial';
  x.fillStyle='#007439';
  x.fillText("довільний текст", 10, 80);
  x.fillStyle='#a67800';
  x.font='20px Comic Sans MS';
  x.fillText("в елементі canvas.", 50, 120);
  </script>
  </body>
  </html>
```

Вставка зображень

В **<canvas>** можуть бути вставлені зображення форматів PNG, GIF і JPEG. Для вставлення зображення необхідно:

1. Створити посилання на малюнок, що знаходиться на сторінці;
2. Намалювати його на полотні з допомогою методу `drawImage ('посилання на малюнок',x,y).`

Створити посилання на малюнок із скриптів можна з допомогою методу `document.getElementById` (також можна використовувати методи `getElementsByTagName` та `images`). Якщо потрібно, щоб малюнок не був відображений на сторінці, його можна приховати за допомогою CSS.

Використання мультимедіа

Можливості мультимедіа. Раніше для того, щоб вбудувати в HTML-сторінку програвач аудіо або відео, доводилося або підключати флеш-програвач, або використовувати які-небудь інші плагіни, встановлені в браузері користувача. Але з приходом HTML5 все значно спростилося. Тепер для відтворення файлів мультимедіа призначені спеціальні теги.

Звук. Для програвання звуку на Web-сторінках можна використовувати тег **<embed>**, основним параметром якого є **src**, що дозволяє визначити адресу звукового файлу.

Таблиця 1.8.

Параметри тега **<embed>**

Назва параметра	**Призначення**
`width`	Ширина відображення звукового програвача в пікселях
`height`	Висота відображення звукового програвача в пікселях
`hspace`	Відступ від звукового програвача до інших об'єктів на HTML-сторінці по вертикалі
`vspace`	Відступ від звукового програвача до інших об'єктів на HTML-сторінці по горизонталі
`hidden`	Показувати/не показувати (false/true) звуковий програвач у вікні браузера
`lowsrc`	Задає файл з альтернативним зображенням
`align`	Вирівнювання рисунка відносно інших об'єктів на HTML-сторінці
`src`	Адреса графічного файлу
`dynsrc`	Адреса відео файлу
`autostart`	Визначення моменту початку прокрутки відеоролика. Можливі значення: `true` – програвання починається відразу після завантаження звукового файлу; `false` – програвання починається після команди користувача
`loop`	Визначення кількості повторів відеоролика. Можливі значення: `true` – програвання відбувається безперервно; `false` - кількість повторів визначається користувачем

Наприклад, для того, щоб при перегляді у браузері web-сторінки прозвучав звук, записаний в файлі *1.wav,* необхідно записати такий HTML-код:

```
<embed src=1.wav >
```

Основні параметри цього тега показані в таблиці1.8.

Використання аудіо. Для відтворення аудіофайлів служить парний блоковий тег **<audio>:**

```
<audio src="daleka_doroga.mp3"
autoplay="autoplay" controls=
"controls" loop="3">Інформація про аудіофайл
</audio>
```

Всередині тега **<audio>** міститься опис аудіофайлу або інший текст, який буде виводитися в тих браузерах, які не підтримують тег **<audio>**. Сам же аудіофайл і параметри його відтворення можуть міститися в атрибутах тега **<audio>**:

- `src` - вказує шлях до аудіофайлу, розміщеного в Інтернеті;
- `autoplay` - наявність цього атрибута свідчить про те, що аудіофайл буде відтворено автоматично після завантаження web-сторінки;
- `controls` - при наявності цього атрибута буде відображена панель управлінняня відтворенням файлу;
- `loop` - містить число, що вказує, скільки разів потрібно повторити відтворення аудіофайлу.

Налаштування аудіофайлу можуть також визначатися в атрибутах **src**, **type** і **codecs** одинарного тега **<source>**, який можна розмістити всередині тега **<audio>**. Всередині тега **<audio>** може перебувати будь-яка кількість тегів **<source>**. Ця особливість часто використовується, щоб вказати кілька варіантів аудіофайлу, створених за допомогою різних кодеків.

Тег **<audio>**, як і тег **<source>**, може працювати з будь-якими аудіофайлами, незалежно від того, яким кодеком вони створені. Однак цього не можна сказати про браузери. Вони підтримують обмежений набір кодеків. Причому, принаймні на

даний момент, список підтримуваних кодеків відрізняється для кожного браузера:

- *Internet Explorer* підтримує кодеки: MP3, AAC;
- *Google Chrome* - ogg/vorbis, MP3, AAC , WAV ;
- *Opera* - ogg/vorbis, WAV;
- *Safari* - MP3, AAC, WAV;
- *Mozilla Firefox* - ogg/vorbis, WAV.

З цієї причини для коректного відтворення аудіофайлу у всіх останніх версіях браузерів доводиться публікувати на сторінці кілька його варіантів, створених за допомогою різних кодеків.

Працюємо з відео. Одна хвилина відео іноді може розповісти користувачам більше, ніж десятки сторінок тексту. В HTML5 парний блоковий тег **<video>** дозволяє розмістити на web-сторінці відеофайл. При цьому усередині тега **<video>** зберігається опис відеофайлу (або інший текст), який буде відображений на сторінці, якщо браузер не підтримує тег **<video>**. Шлях до відеофайлу, який потрібно відтворити, може розташовуватися в атрибуті `src` цього тега:

```
<video src="/myvideo/video.avi">Опис
відеофайла </video>
```

Атрибут `controls` відображає у плеєрі кнопки управління відео, а атрибути `height` (висота) і `width` (ширина) визначають розміри плеєра.

Відмітимо, що відео підтримують браузери: *Internet Explorer 9+, Opera 10.50+, Firefox 3.5+, Chrome 3.0+, Safari 3.1+*. Однак різні браузери підтримують різні формати файлів. Opera і Firefox підтримують формат Ogg (з відео кодеком Theora і аудіо кодеком Vorbis) і WebM (з відео кодеком VP8 і аудіо кодеком Vorbis), а Internet Explorer і Safari підтримують MPEG4 (відео з кодеком H.264 та аудіокодеком AAC). Браузер Chrome підтримує всі перераховані формати.

Таким чином, для того, щоб відео нормально відтворювалося у всіх браузерах, необхідно додати джерела у форматі Ogg і MPEG4 (або WebM і MPEG4).

Шлях до відеофайла може також міститися в атрибуті **src** вкладеного одинарного тега **<source>**. Причому за допомогою

тега <**source**> можна надати кілька джерел відео в різних форматах для відтворення. Тоді браузер буде використовувати перший формат, який він підтримує. *Приклад*:

```
<html>
<body>
<p>Панорама гори Білуха.</p>
<video width="300" height="200"
controls="controls">
<source src="mountvideo.ogv"
type="video/ogg" />
<source src="mountvideo.mp4"
type="video/mp4" />
                <source src="mountvideo.webm"
              type="video/webm"/>
</video>
</body>
</html>
```


1.4. Гіперпосилання

Як відомо, гіперпосилання призначені для зв'язування одного web–ресурсу з іншим. Гіперпосилання складається із двох частин. Перша частина – це об'єкт, який користувач бачить у вікні браузера, і вибір якого призводить до переходу на цільовий web-ресурс, URL-адреса якого вказана у другій частині гіперпосилання. Цим об'єктом може бути текст або/та зображення. В залежності від цього класифікують текстові та графічні гіперпосилання. При типових налаштуваннях браузера гіперпосилання виділяються на HTML-сторінці за допомогою кольору. Крім того, текст гіперпосилання виділяється за допомогою підкреслення. Для визначення гіперпосилання в коді HTML - сторінки використовують тег **<a>.**

Основним параметром тега **<a>** є параметр **href**, що задає URL-адресу гіперпосилання. Синтаксис запису гіперпосилання такий:

```
<a href=" URL-адресу "> Текстовий або
графічний об'єкт </a>
```

Приклади запису текстового та графічного гіперпосилання показані нижче:

```
<a href="www.meta.ua"> Текстове гіперпосилання </a>
<a href="www.meta.ua"><img src="logo.gif"></a>
```

Вказана URL-адреса може бути відносна або абсолютна. При використанні **відносної** адреси повний шлях до цільового ресурсу не задається. В цьому випадку визначення місцезнаходження цільового ресурсу відбувається з урахуванням місцезнаходження HTML-сторінки, в якій використане гіперпосилання.

Наприклад, якщо у файлі *1.html* визначене гіперпосилання:

```
<a href="2.html "> Текстовий або графічний об'єкт </a>
```

то мається на увазі, що цільова HTML-сторінка *2.html* розміщена в тій же папці того ж комп'ютера, що і сторінка *1.html*.

Абсолютною називається URL-адреса, в якій повністю визначено комп'ютер, папка та файл цільового ресурсу.

Наприклад:

```
<a href="www.example.com/html/3.html">
Абсолютне гіперпосилання</a>
```

Важливим параметром тега **<a>** є атрибут **target**, за допомогою якого можна вказати, як буде відкриватися документ, на який вказує посилання. Цей атрибут може набувати таких значень:

- **_top** - документ відкриється в поточному вікні браузера;
- **_blank** - документ відкриється в новому вікні браузера;
- **_self** - документ відкриється в поточному фреймі;
- **_parent** - документ відкриється у батьківському фреймі.

Наприклад, для визначення текстового гіперпосилання, вибір якого призводить до відображення в новому вікні браузера файлу *2.html*, необхідно записати:

```
<a href="2.html " target="_blank"> Текст </a>
```

При відсутності атрибута **target** цільовий ресурс буде завантажуватися у вікно браузера, в якому відображається HTML-сторінка з гіперпосиланням.

Цікавою є можливість гіперпосилань задати як цільовий ресурс певну частину власної HTML-сторінки. Зокрема, у великому

документі можна зробити закладинки (“якорі”) з гіперпосиланнями на певні розділи. В цьому випадку перехід за гіперпосиланням до певного розділу (який може міститися в кінці документа) призведе до відображення цього розділу на екрані. Такі гіперпосилання називаються **внутрішніми**. Для їх побудови необхідно за допомогою тега **<a>** вказати місце переходу (покажчик) та за допомогою параметра **name** тега **<a>** присвоїти йому ім’я.

Наприклад, для переходу на 12 розділ документа на початку тексту цього розділу розмістимо покажчик з іменем "go12":

```
<a name="go12"></a>Текст розділу 12
```

В цьому випадку параметр `href` не використовується, а ім’я покажчика повинно бути записане за допомогою літер англійського алфавіту та цифр.

Особливістю гіперпосилання для переходу всередині HTML-сторінки є використання в параметрі `href` імені покажчика з перфіксом #. *Наприклад*:

```
<a href="#go12">Перехід до розділу 12</a>
```


Питання для самоконтролю

1. Розміщення зображень у в HTML-документах.
2. Змістові і фонові зображення.
3. Створення альтернативного тексту.
4. Використання <canvas>.
5. Можливості мультимедіа в HTML5.
6. Які параметри має тег <embed>?
7. Для чого підключають кілька варіантів одного аудіофайлу?
8. Назвіть кодеки, які підтримують популярні браузери.
9. Який браузер краще обирати при перегляді відео і чому?
10. Гіперпосилання в HTML5.

1.5. Таблиці. Форми

Великі обсяги однорідної інформації подають у вигляді таблиць з даними. Таблиці є одним із основних засобів формування HTML-документів. Розміщення інформації в HTML-документі, виконане за допомогою інших засобів (тегів), може по-різному відображатися браузерами різних типів.

Для побудови таблиці використовують теги-контейнери: тег таблиці **<table>**, тег комірки заголовка **<th>**, тег рядка **<tr>** та тег комірки даних **<td>**. Таблиця має бути розміщена в тілі HTML-документа, будується вона рядками, причому кількість комірки в рядках повинна бути однакова. Таким чином одній таблиці відповідає один тег **<table>** та стільки тегів **<tr>**, скільки рядків має таблиця. Якщо в таблиці немає об'єднаних комірок, то сумарна кількість тегів **<td>** та **<tr>** дорівнює добутку кількості рядків на кількість колонок таблиці.

Різниця між тегами <**td**> та <**th**> полягає у форматуванні та вирівнюванні розміщеного в них тексту. В тегу <**td**> текст відображається стандартним шрифтом з горизонтальним вирівнюванням "до лівого краю". В тегу <**th**> текст відображається напівжирним шрифтом з горизонтальним вирівнюванням "до середини". Для розміщення інформації (тексту або графіки) всередині таблиці необхідно помістити її в тег комірки. Таблиця може мати заголовок, якому відповідає парний тег **<caption>**, він має бути розміщений після тега **<table>** до першого тега <**tr**>. Для створення підсумку таблиці використовують тег **<tfoot>**. Наведемо приклади HTML-коду для визначення таблиці.

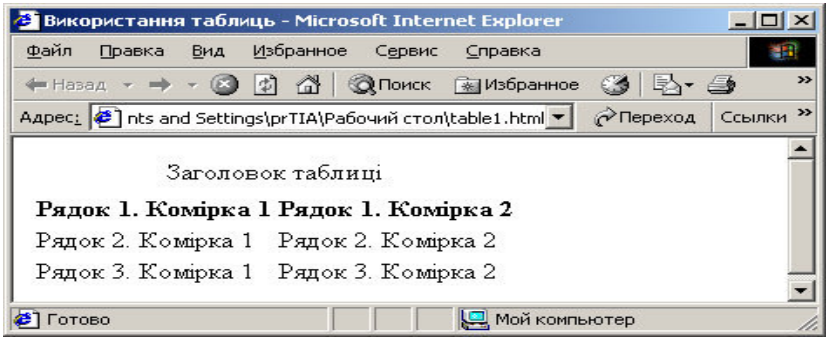


Рис. 1.18 Використання таблиць

Приклад 1. Таблиця складається із заголовка та трьох рядків, в кожному із яких має бути дві комірки:

```
<table>
```

```
<caption>Заголовок таблиці </caption>
  <tr><th>Рядок 1. Комірка 1</th><th> Рядок 1.
Комірка 2</th></tr>
  <tr><td>Рядок 2. Комірка 1</td><td> Рядок 2.
Комірка 2</td></tr>
  <tr><td>Рядок 3. Комірка 1</td><td> Рядок 3.
Комірка 2</td></tr></table>
```

В цьому прикладі для зручності ідентифікації комірок в них розміщено текст, що відповідає номеру рядка та номеру комірки в рядку. Відображення такої таблиці у вікні браузера показано на рис. 1.18.

Приклад 2.

```
  <table>
  <tr> <td>1</td> <td>2</td> <td>3</td> </tr>
  <tr> <td>4</td> <td>5</td> <td>6</td> </tr>
  </table>
```

В браузері:

Приклад 3.

```
  <table cellpadding="15" border="1">
  <tr><td colspan="2">1</td></tr>
  <tr><td>2</td><td>3</td></tr>
  </ table >
```

В браузері:

При відсутності параметра `border` у тегах, що визначають таблицю, її межі не відображаються.

Розглянемо призначення деяких параметрів тегів <**table**>, **<caption>, <tr>, <td> та <th>.**

Тег <**caption**> найчастіше використовується тільки з параметрами **align** та **valign**, що визначають відповідно горизонтальне та вертикальне вирівнювання заголовка таблиці.

Можливі значення параметра **align** тега <**caption**>:

- **right** - заголовок вирівнюється до правого краю таблиці;
- **center** - заголовок вирівнюється до центру таблиці;
- **left** - заголовок вирівнюється до лівого краю таблиці.

Параметр **valign** дозволяє розміщувати заголовок над таблицею (значення **top**) або під таблицею (значення **bottom**). При відсутності параметрів **align** та **valign** вирівняний до центру заголовок розміщується над таблицею. Параметри тега <**table**> і їх призначення наведені в табл. 1.9.

Таблиця 1.9.

Параметри тега <**table**>

Параметр	Призначення
border	Товщина межі таблиці в пікселях
bordercolor	Колір меж таблиці
cellspacing	Відстань між комірками таблиці в пікселях
cellpadding	Відстань від меж комірок до даних, що розміщені в цих комірках
width	Ширина таблиці в пікселях або відсотках від ширини HTML-документа
height	Висота таблиці в пікселях
align	Горизонтальне вирівнювання таблиці
bgcolor	Колір фону таблиці
background	Фоновий рисунок таблиці

Розглянемо типові приклади використання параметрів тега таблиці.

Приклад 1. Побудуємо таблицю, що складається з двох рядків та двох колонок шириною 400 пікселів, висотою 200

пікселів, товщиною меж 5 пікселів, відстанню між комірками 7 пікселів та відстанню від меж комірки до даних в комірках 3 пікселі.

Відповідний HTML-код може бути таким:

```
<table width=400 height=200 border=5 cellspacing=7 cellpadding=3 >
<tr><th>Рядок 1. Комірка 1</th><th> Рядок 1. Комірка 2</th></tr>
<tr><td>Рядок 2. Комірка 1</td><td> Рядок 2. Комірка 2</td></tr>
</table>
```

Відображення створеної таблиці у вікні браузера показане на рис. 1.19.

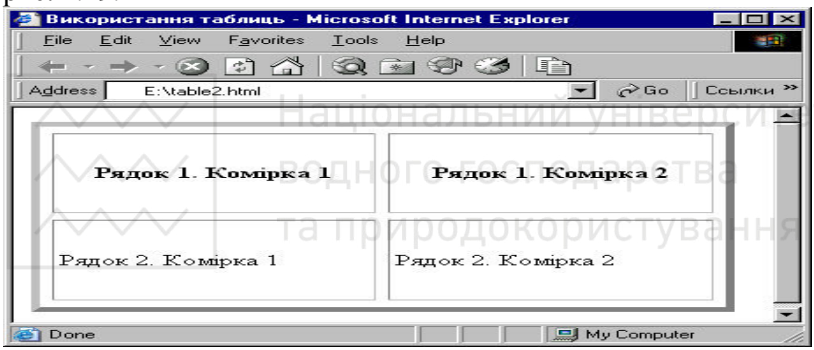


Рис. 1.19. Використання параметрів тега <table>

Приклад 2. Розглянемо, як можна змінити взаємне розміщення таблиці і інших елементів на HTML-сторінці за допомогою параметра **align**. Можливі значення цього параметра для тега <**table**>:

- **align="center"** - горизонтальне вирівнювання "до центру", таблиця "не обтікається" текстом;
- **align="left"** - горизонтальне вирівнювання "до лівого краю", таблиця "обтікається" текстом з правого боку;
- **align="right"** - горизонтальне вирівнювання "до правого краю", таблиця "обтікається" текстом з лівого боку.

HTML-код для розміщення в HTML-документі тексту та таблиці, вирівняної "до правого краю":

```
<table border="1" align="right">
  <tr><th>Рядок 1. Комірка 1</th><th>Рядок 1.
Комірка 2</th></tr>
  <tr><td>Рядок 2. Комірка 1</td><td>Рядок 2.
Комірка 2</td></tr>
  </table>
```

Таблиця вирівняна "до правого краю". Відображення цієї таблиці та тексту у вікні браузера показано на рис 1.20.

Відзначимо, що наведені в табл. 1.9 параметри **bordercolor, bgcolor** та **background** використовуються також у тегах рядків (**<tr>**) та комірок (**<td>**, **<th>**) для визначення кольору меж, кольору фону, фонових рисунків рядків та комірок. Крім того, як в тегах рядків, так і в тегах комірок для горизонтального та вертикального вирівнювання розміщеної в них інформації можна використовувати параметри **align** та **valign**. В тегах <**tr**>, <**td**> та <**th**> параметр **align** може набувати значення **left, center** та **right** для

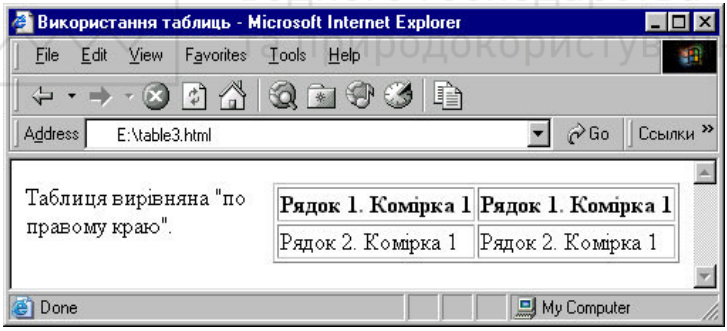


Рис. 1.20. Спільне розміщення таблиці та тексту при `align="right"`

горизонтального вирівнювання інформації, відповідно, "до лівого краю", "до центру" та "до правого краю" комірок.

В тегах <**tr**>, <**td**> та <**th**> параметр **valign** може набувати значення **top, middle, bottom,** та **baseline** для вертикального вирівнювання інформації, відповідно, "до верхнього краю", "до середини", "до нижнього краю" та "за базовою лінією тексту" комірок.

Якщо параметри **bordercolor, bgcolor, background, align** та **valign** визначені в тегу <**tr**>, то їх дія поширюється на весь рядок таблиці, але ці параметри можуть бути перевизначені в будь-якій комірці.

Особливістю тегів <**td**> та <**th**> є можливість використання параметрів **rowspan** та **colspan** для об'єднання декількох сусідніх комірок в одну.

Параметр **rowspan** використовується для об'єднання комірок "по вертикалі", а параметр **colspan** – для об'єднання комірок "по горизонталі". Синтаксис запису цих параметрів такий:

```
rowspan="n"
colspan="m"
```

де `n` та `m` – кількість комірок, що об'єднуються.

Допускається сумісне використання параметрів **rowspan** та **colspan** для однієї комірки. Наведемо HTML-код таблиці, що складається із двох рядків та трьох колонок, при цьому середня комірка займає обидва рядки:

```
<table border="1" > <tr><td> Рядок 1. Комірка 1</td>
<td rowspan="2"> Рядки 1,2. Комірка 2</td>
<td> Рядок 1. Комірка 3</td></tr>
<tr><td> Рядок 2. Комірка 1</td>
<td> Рядок 2. Комірка 3</td></tr></table>
```

Відображення таблиці у вікні браузера показано на рис. 1.21.

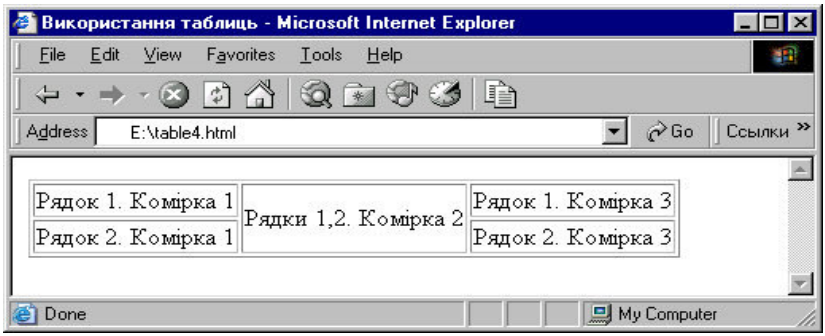


Рис. 1.21. Вертикальне об'єднання комірок в таблиці

Наведемо HTML-код таблиці, що складається із двох рядків та трьох колонок, при цьому всі комірки нижнього рядка об'єднані в одну:

```
<table border="1"> <tr><td>Рядок 1. Комірка 1</td>
<td>Рядок 1. Комірка 2</td><td>Рядок 1. Комірка 3</td></tr>
<tr><td colspan="3">Рядок 2. Комірки об'єднані в одну 1</td></tr>
</table>
```

Відображення цієї таблиці у вікні браузера показано на рис. 1.22.

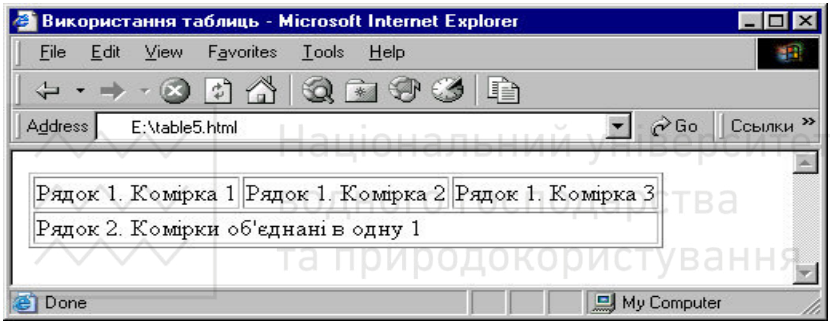


Рис. 1.22. Горизонтальне об'єднання комірок в таблиці

Створення форм

HTML-форми використовуються для прийому даних, введених користувачем, і подальшої передачі їх на сервер. Зазвичай, результатом такої передачі є відображення у клієнта нового HTML-документа, сформованого web-сервером на основі переданої інформації. HTML-форми можуть містити такі елементи введення:

- текстові поля
- прапорці
- радіо-кнопки
- кнопки відправлення та ін.

Форми також можуть містити списки вибору, багаторядкові текстові поля, мітки та ін.

Форма в HTML-документі створюється за допомогою тега-контейнера <**form**> та складається із набору полів введення, що визначені за допомогою тегів <**input**>, <**select**> та <**textarea**>. Дані на web-сервер передаються у вигляді:

змінна = значення,

де **змінна** – ім'я поля введення, задане за допомогою параметра **name**, *наприклад*:

```
<input type="text" name="mytext">
```

Тег **<form>** у вікні браузера не відображається, а поля введення відображаються за допомогою стандартних графічних елементів управління.

Приклад:

```
<html>
<body>
<form>
<p>Введіть Ваше ім'я:</p>
<input type='text' />
<p>Введіть пароль: </p>
<input type='password' />
</form>
</body>
</html>
```

Елементи введення

Елементи введення використовуються для прийому даних і відрізняються один від одного значеннями атрибута **type**.

Текстове поле <**input type="text" /**> визначає однорядкове текстове поле, у якому користувач може вводити інформацію. За замовчуванням текстове поле вміщує 20 знаків.

Приклад:

```
<html>
<body>
<form>
<p> Введіть прізвище,ім'я, по-батькові в
наступні поля: </p> <br />
Ім'я: <input type='text' name='firstname'
/><br />
```

```
  Прізвище: <input type='text' name='lastname'
/><br />
  По-батькові: <input type='text'
name='lastname' />
  </form>
  </body>
  </html>
```

Поле пароля `<input type="password" />` визначає поле для введення пароля. Текст, що вводиться у це поле, приховується іншими символами (зазвичай, крапки чи зірочки), що дозволяє вводити паролі навіть у присутності сторонніх.

Приклад:

```
  <html>
  <body>
  <form>
  Введіть пароль: <input type='password'
name='pass' />
  </form>
  </body>
  </html>
```


Прапорець: `<input type="checkbox" />`. Прапорці дозволяють користувачам вибирати декілька (або жодного) пунктів з попередньо заповненої інформацією групи. *Приклад*:

```
  <html>
  <body>
  <form>
  <p> Як ви ставитесь до польотів у космос?
</p>
  <input type='checkbox' name='space'
value='1'/> Позитивно<br />
  <input type='checkbox' name='space'
value='2'/> Байдуже<br />
  <input type='checkbox' name='space'
value='3'/> Негативно<br />
  </form>
  </body>
  </html>
```

Зауваження. Разом з `type="checkbox"` може використовуватися додатковий атрибут `checked`, який відмічає даний пункт за замовчуванням. За замовчуванням можна відмічати як всі пункти, так і жодного.

Радіо-кнопка визначається так: `<input type="radio"/>`. Радіо-кнопки дозволяють користувачам вибрати тільки один пункт з попередньо заповненої інформацією групи. *Приклад*:

```
<html>
<body>
<form>
<p> Вкажіть Вашу стать:</p>
<input type='radio' name='s' value='m' />
Чоловіча <br />
<input type='radio' name='s' value='f' />
Жіноча
</form>
</body>
</html>
```


Зауваження. Разом з `type="radio"` може використовуватися атрибут `checked`, що відмічає даний пункт за замовчуванням, але тільки для одного пункту.

Тег **`<textarea>`** призначений для створення всередині форми поля для введення тексту із декількох рядків. Крім обов'язкового параметра `name` в цьому тегу використовуються параметри **`cols`** та **`rows`**. Ці параметри дозволять визначити кількість колонок та рядків тексту, що відображається в даному елементі управління.

Кнопка відправлення. Запис `<input type="submit"/>` визначає кнопку відправлення. Після натискання на цю кнопку дані, введені користувачем, будуть відправлені на сервер. Адреса, на яку будуть пересилатися дані форми, вказується в атрибуті **`action`** тега **`<form>`**. Якщо цей атрибут відсутній, то необхідно передбачити опрацювання даних на поточній сторінці (наприклад, засобами JavaScript). Зверніть увагу: обробка даних, отриманих сервером в результаті відправки

форм, буде проводитися за допомогою серверних мов (зокрема, PHP, Ruby або Python). *Приклад*:

```
<html>
<body>
<form name="input" action="form.php"
method="get">
Введіть Ваше ім'я: <input type="text"
name="name" />
<input type="submit" value=" Відправити " />
</form>
</body>
</html>
```

Список, що розкривається. Для створення списків, що розкриваються, використовується тег <select>, а елементи такого списку визначаються за допомогою тега <option>.

Зауваження: користувачі неохоче вводять дані вручну у текстові поля, тому їх бажано замінювати випадаючими списками, радіо-кнопками або прапорцями там, де це можливо.

Приклад:

```
<html>
<body>
<p> Вкажіть Вашу стать: </p>
<form>
<select name='sex' >
<option value='m'> Чоловіча </option>
<option value='f'> Жіноча </option>
</select>
</form>
</body>
</html>
```

За допомогою атрибута **multiple** можна вказати, що у випадаючому списку буде можливість обрати одночасно кілька елементів.

Приклад:

```
<html>
<body>
```

```
<p> В цьому списку може бути вибрано
одночасно кілька значень (для цього натисніть
клавішу Ctrl і клацніть на необхідних
елементах): </p>
  <form>
  <select name='city' multiple='multiple'>
  <option value='london'>Лондон</option>
  <option value='kyiv'> Київ </option>
  <option value='newyork'>Нью-Йорк</option>
  </select>
  </form>
  </body>
  </html>
```

Зауваження. В HTML5 було введено багато нових параметрів тега <input>, що дозволяють проводити валідацію введення даних та надають нові можливості для візуалізації даних (таких як дата, рядок прогресу тощо). Детальніша інформація про ці параметри вміщена далі в пункті "Нові типи <input> в HTML5-формах".

Заголовки у формах. Для того, щоб назвати групу елементів форми, слід з допомогою тега **<**fieldset**>** згрупувати потрібну частину форми і за допомогою тега **<**legend**>** встановити заголовок.

Основними параметрами тега <form> є action, enctype та method. Обов'язковий параметр action визначає URL-адресу, куди передається інформація. Параметр **enctype** визначає формат кодування даних при їх передачі. Можливими значеннями цього параметра є **application/x-www-form-urlencoded** (використовується при стандартних установках) та **multipart/form-data** (використовується при передачі файлів). Якщо у формі передбачено поле для передачі файлів, значення **multipart/form-data** слід використовувати обов'язково, бо за його відсутності буде передаватися лише ім'я файлу, а не його вміст.

Параметр **method** визначає метод передачі інформації. Можливими значеннями цього параметра є **get** та **post**. Метод **get** рекомендується використовувати, якщо обсяг інформації, що

Таблиця 1.10.

Основні параметри тега <**input**>

Параметр	**Можливі значення**	**Мета використання**
name	визначаються Web-програмістом	Визначає ім'я елементу управління
type	text	Створює поле для введення рядка тексту
	password	Створює поле для введення пароля
	file	Створює поле для введення імені файлу на комп'ютері користувача
	submit	Створює кнопку, натискання якої ініціює відправку даних форми
	reset	Створює кнопку, натискання якої відновлює значення полів форми, які були на момент завантаження
	button	Створює кнопку
	checkbox	Створює елемент управління типу checkbox
	radio	Створює радіо-перемикач. Всі елементи групи радіо-перемикачів повинні мати однакове значення параметра name
width, height	визначаються Web-програмістом	Задає ширину та висоту елемента управління типу submit, reset, button
size		Задає максимальну кількість символів, що будуть відображатись в елементах управління типу text, password та file

продовження таблиці 1.10

maxlength		Задає максимальну кількість символів, які можна ввести в елементах управління типу text та password
value		Початковий напис та початкове значення елементів управління

передається, не перевищує 255 символів. Метод **post** слід застосовувати при передачі великого обсягу інформації.

Тег <**input**> використовується для генерації елементів управління: кнопки, поля для введення тексту, паролі, імена файлів. Основні параметри тега наведені в таблиці 1.10.

Наведемо приклад HTML-коду для визначення форми, що містить елементи управління для введення рядка тексту, пароля, кнопки типу submit для відправлення даних на сервер та кнопки типу reset для відновлення початкового значення полів (відображення форми у вікні браузера показано на рис. 1.23):

```
  <body>
  <table>
  <form action="password.php" method="get">
  <tr><td><input type="text" name="login"
size="21" maxlength="21"> Введіть ім'я
</td></tr>
  <tr><td><input type="password" name="passw"
size="21" maxlength="21"> Введіть пароль
</td></tr>
<tr><td><input type="reset" value="Очистити"
width="50" height="30">
<input type="submit" value="Відправити"
width="50"
height="30"></td></tr>
</form>
</table>
</body>
```

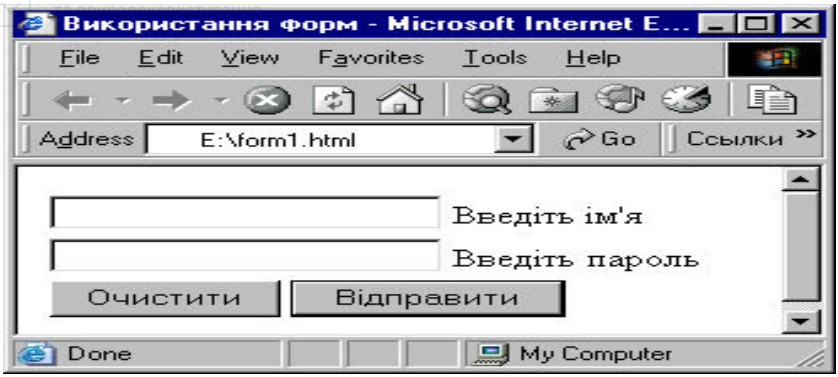


Рис. 1.23. Використання форми введення ідентифікаційних даних

Призначення цієї форми:

- ідентифікація користувача на Web-сервері за допомогою персональних даних (ім'я, пароль);
- використання таблиці для розміщення елементів управління;
- використання текстових пояснень про призначення елементів управління.

Такі форми передаються методом `get` та призначені для обробки Web-додатком *password.php,* який розміщений в тій же папці, що і HTML-документ з формою.

Зауваження. В наведеному прикладі використано табличну верстку документа, яка на даний момент вважається застарілою. Блочну верстку документа ми розглянемо при вивченні CSS.

Нові елементи форм, введені в HTML5

Зауваження: нові типи `<input>` підтримуються тільки у браузерах *Opera 11+* і *Chrome 12+*.

`input type=email` визначає поле, що має містити email-адресу. Значення, введене в поле, автоматично перевіряється перед відправкою на сервер.

`input type=url` визначає поле, яке повинно містити url-адресу. Значення, введене в поле, автоматично перевіряється перед відправкою на сервер.

input type=tel визначає поле для введення телефонного номера. За допомогою атрибута **pattern** можна встановити формат допустимого телефонного номера.

input type=number визначає поле, яке повинно містити числа. Діапазон прийнятих чисел можна обмежувати за допомогою атрибутів **min** (мінімальне допустиме число) і **max** (максимальне допустиме число). За допомогою атрибута **step** можна вказати крок для формування списку допустимих чисел.

input type=range визначає поле, яке може містити значення в певному інтервалі. Відображається як повзунок, який можна перетягувати мишею. Обмежувати діапазон прийнятих чисел можна за допомогою атрибутів **min** і **max**. За допомогою атрибута **step** можна задавати крок формування списку допустимих чисел.

input type=search визначає поле пошуку (*наприклад*, пошук на сайті).

datalist дозволяє прив'язати список полів форми. Значення списку виводиться як пошукові підказки під час введення інформації в поле пов'язане з ним.

keygen дозволяє генерувати відкриті і закриті ключі, які використовуються для безпечного зв'язку з сервером.

output може використовуватися для відображення різної інформації. За допомогою атрибута **for** можна вказати пов'язані поля.

Нові атрибути в HTML5-формах

Зауваження: Нові атрибути елементів форми підтримуються тільки в браузерах Firefox 4+, Opera 11+ і Chrome 10+.

autofocus робить поле активним після завантаження сторінки (може використовуватися із всіма типами **input**).

form вказує форму, якій належить це поле (може використовуватися із всіма типами **input**).

multiple вказує, що дане поле може приймати кілька значень (може використовуватися з **input** типів **email** і **file**).

novalidate вказує, що це поле не повинно перевірятися перед відправкою (може використовуватися з **form** і **input**).

placeholder відображає текст-підказку в полі (використовується з **input** таких типів: **text, search, url, tel, email** і **password**).

required вказує, що це поле має бути обов'язково заповнено перед відправкою.

Вибір дати. В HTML5 були додані нові елементи введення, які дозволяють зручно вибирати дату і час.

Зауваження: Поля для вибору дати підтримуються тільки в браузерах *Opera 9+, Chrome 10+* і *Safari 5.1+*.

date дозволяє вибрати дату у форматі рік-місяць-день;

time дозволяє вибрати час;

datetime дозволяє вибрати дату у форматі рік-місяць-деньТчасZ (відлік ведеться за глобальним часом);

datetime-local дозволяє вибрати дату у форматі рік-місяць-деньТчас (відлік ведеться за місцевим часом);

month дозволяє вибрати дату у форматі рік-місяць;

week дозволяє вибрати дату у форматі рік-тиждень.

Приклад:

```
<html>
<body>
<form action='html5.php'>
Виберіть дату: <input type='date' /><br />
Виберіть час: <input type='time' /><br />
Виберіть глобальний час і дату: <input
type='datetime' /><br />
Виберіть місцевий час і дату: <input
type='datetime-local' /><br />
Виберіть місяць: <input type='month' /><br
/>
Виберіть тиждень: <input type='week' />
</form>
</body>
</html>
```

Питання для самоконтролю

1. Які теги використовують для побудови таблиць у HTML?
2. У чому полягають відмінності застосування <td> і <th>?
3. Вирівнювання даних у таблицях.
4. Призначення параметрів `rowspan` і `colspan`.
5. Виведення границь таблиці.
6. Для чого використовуються форми?
7. Перелічіть параметри тега <form>.
8. Що являють собою поля введення форми?
9. Який метод слід використовувати при передачі великого обсягу інформації?
10. Назвіть основні параметри тега <input>.

1.6. Додаткові можливості HTML5

Web-сховища

Web-сховища дозволяють зберігати дані на стороні клієнта.

Зауваження: всі HTML5-технології, які тут розглядаються, вимагають знання JavaScript для їх використання. Ознайомитися з JavaScript можна в розділі 3 посібника.

Web-ховища можна вважати функціональною альтернативою `cookie`.

Переваги використання web-сховищ:

- можливість зберігання необмежених обсягів інформації;
- інформація, що зберігається у сховищах, доступна навіть без підключення до інтернету;
- дані, що знаходяться у сховищі, не надсилаються при кожному запиті сторінок;
- сховища безпечніші, ніж `cookie`;
- web-сховища підтримуються в наступних браузерах: *IE8+, Safari 4+, Chrome 4+, Firefox 3.5+, Opera 10.50+*.

Зберігання даних. Для зберіганя даних використовують два спеціальних об'єкти: **`sessionStorage`** і **`localStorage`**. Звертатися до них можна за допомогою JavaScript чи інших клієнтських скриптів. Кожен web-сайт має доступ тільки до

своїх даних у сховищі і тому не може звертатися до даних, які були збережені іншими сайтами.

Об'єкт **sessionStorage** зберігає дані протягом користувацької сесії. Коли браузер користувача буде закритий, дані, збережені в об'єкті, будуть вилучені. Дані у сховищі доступні з усіх сторінок сайту, а не тільки з тієї, з якої вони були збережені.

Об'єкт **localStorage** зберігає дані на необмежений період часу. Дані, збережені в цьому об'єкті, будуть доступні навіть без підключення до інтернету.

Drag and Drop

Технологія **drag and drop** (іноді може згадуватися як drag'n'drop) дозволяє користувачеві переміщувати елементи на web-сторінці. Раніше це було можливим лише за допомогою реалізації складних функцій JavaScript або підключенням jQuery. Зараз технологія drag and drop є частиною специфікації HTML5 і її підтримка вбудована в усі сучасні web-браузери.

Щоб надати елемету (малюнок, посилання, звичайний текстовий елемент) властивість "переміщення", необхідно додати до нього HTML5-атрибут **draggable** із значенням **true**.

Приклад:

```
<html><body>
<p>Спробуйте перетягнути елемент.</p>
<img src="gimg3.gif" draggable="true">
</body>
</html>
```

Обробка елемента під час перетягування може бути умовно розділена на два кроки:

1. Збереження даних перетягуваного елемента до початку операції перетягування;

2. Добування раніше збережених даних, після того, як елемент, що перетягується, буде переміщений у приймаючий його елемент.

В процесі перетягування елемент можна опрацьовувати за допомогою спеціальних **подій перетягування**, які були додані в HTML5:

Подія	Опис
`dragstart`	Виконується на початку операції перетягування
`ondrag`	Виконується під час перетягування елемента
`ondragenter`	Виконується, коли елемент, що перетягується, буде переміщений на елемент, який може його прийняти
`ondragleave`	Виконується, коли елемент, що перетягується, буде виведений за межі границь елемента, який може його прийняти
`ondragover`	Виконується, коли елемент, що перетягується, буде переміщуватися в межах границь елемента, який може його прийняти
`ondrop`	Виконується, коли елемент, що перетягується, буде переміщено в елемент, який його приймає
`ondropend`	Виконується в кінці операції перетягування

Для того, щоб зберегти дані, ми повинні скористатися методом **`setData("тип_даних", дані _зберігання")`** HTML5-об'єкта **`DataTransfer`**. Дані елемента необхідно зберігати на початку перетягування, тобто під час події **`ondragstart`**.

Щоб добути раніше збережені дані, необхідно скористатися методом **`getData("тип_даних")`** об'єкта **`DataTransfer`**. Дані перетягуваного елемента слід добувати після того, як елемент буде переміщений в область елемента приймання, тобто під час події **`ondrop`**. Зауважимо, що тип даних під час збереження і добування повинен збігатися, тобто якщо зберігались дані типу **`"text/plain"`**, то необхідно використовувати **`"text/plain"`** і для їх добування.

За замовчанням елементи не можуть приймати інші елементи, тому якщо ми хочемо створити приймаючий елемент, необхідно прив'язати до нього обробник події **ondragover** та передати йому **event.preventDefault().**

Використання web worker

В HTML всі скрипти JavaScript на сторінці могли виконуватися тільки по черзі. В деяких випадках користувачі повинні були чекати завершення виконання скрипта, щоб продовжити взаємодію з web-сторінками. Завдяки технології web worker в HTML5 стало можливим створення скриптів, які можуть виконуватися у фоновому режимі і не будуть впливати на швидкість завантаження основної сторінки. *IE10+, Chrome, Firefox, Opera* і *Safari* мають підтримку даної технології.

Створення web worker. Оскільки скрипти в технології **web worker** працюють паралельно з основною сторінкою, їх код завжди має бути винесений в окремий .js- файл. Після цього на основній сторінці необхідно створити новий об'єкт **worker** з посиланням на зовнішній файл скрипта:

```
var worker=new worker(посилання_на_файл_скрипта);
```

Якщо вказаний файл існує, браузер почне завантаження скрипта новим потоком асинхронно. Після того, як скрипт буде завантажений, можна запустити **worker** з допомогою:

```
worker.postMessage();
```

Взаємодія скриптів **web worker** з основною web-сторінкою відбувається з допомогою спеціальних подій і методу **postMessage()**.

Наприклад, подія **message** спрацьовує, коли **worker** отримує якесь повідомлення.

Слід мати на увазі, що **web worker** не мають доступу до:

- DOM структури основної сторінки;
- об'єктів **window, document, parent.**

Зазначимо, що використання **web worker** виправдано для завдань, які потребують значних ресурсів комп'ютера для їх виконання, *наприклад* кешування великих масивів з даними для

подальшого використання, перевірка тексту на помилки, підсвічування синтаксису або інші операції з форматування тексту в реальному часі, оновлення великих обсягів інформації в базах даних, аналізування відео- або аудіофайлів та ін.

Вимкнення web worker. Після створення **worker** буде чекати повідомлення від основної сторінки, навіть коли виконання його коду завершиться. Тому необхідно явно завершувати роботу **worker**, коли він більше не потрібен. Щоб завершити роботу **worker** і звільнити ресурси комп'ютера, використовують метод **termitane():**

```
worker.terminate();
```

Геолокація

З допомогою HTML5-геолокації можна визначити місце розташування користувача. Оскільки інформація про місцезнаходження може порушувати конфіденційність, перш ніж сервер зможе отримати таку інформацію, користувач повинен явним чином підтвердити, що не має заперечень з цього приводу.

Відмітимо, що координати місця розташування визначаються більш точно на мобільних пристроях, тому що вони мають свій вбудований GPS-приймач. Розташування користувачів на стаціонарних комп'ютерах визначається шляхом комбінованого аналізу IP-адреси комп'ютера та IP-адрес оточуючих WI-FI роутерів, в радіусі покриття яких комп'ютер перебуває в даний момент.

Поточна позиція користувача може бути визначена за допомогою методу **getCurrentPosition()** об'єкта **navigator.geolocation**:

```
navigator.geolocation.getCurrentPosition(suc
cess_function error_function, options);
```

success_function - ім'я функції, яка буде викликана, якщо координати користувача будуть зчитані вдало;

error_function - ім'я функції, яка буде викликана, якщо при зчитуванні координат станеться помилка;

options - дозволяє задати параметри, які будуть використані при зчитуванні координат.

Можливі значення:

enableHighAccuracy - якщо набуває значення `true`, браузер буде намагатися визначити розташування якомога точніше;

timeout - встановлює максимально допустимий час, який браузер може використовувати для зчитування даних (за замовчуванням час зчитування не обмежено); **maximumAge** – визначає, як довго браузер буде зберігати в кеші попереднє збережене значення.

Якщо користувач дозволив використовувати дані про його місцезнаходження і вони були вдало зчитані браузером, у функцію **success_function** як параметр буде передано об'єкт, що містить властивість **timestamp** (час зчитування координат) і об'єкт **coords**, який має такі властивості:

Подія	Опис
`latitude i longitude`	Широта и довгота місцезнаходження користувача
`altitude`	Висота над рівнем моря в метрах
`accuracy`	Точність визначення широти і довготи (чим більше число, тим менша точність)
`altitudeAccuracy`	Точність визначення висоти (чим більше число, тим менша точність)
`heading`	Показує напрямок користувача в градусах (0 градусів – напрям на північ, 180 - на південь і т.д.)
`speed`	Швидкість переміщення в метрах за секунду

Якщо при зчитуванні координат користувача виникла помилка, буде викликана функція **error_function**. Можливі помилки при зчитуванні координат:

Помилка	Опис
PERMISSION_ DENIED	Користувач заборонив зчитувати інформацію про його місцезнаходження
POSITION_ UNAVAILABLE	Браузер не може визначити місцезнаходження користувача
TIMEOUT	Браузер не встиг визначити місцезнаходження користувача у наданий йому час
UNKNOWN_ ERROR	Під час визначення місцезнаходження виникла невідома помилка

Відстеження зміни розташування. За допомогою методу **`watchPosition()`** можна зчитувати дані про місцезнаходження користувача через певні інтервали часу. Після запуску цей метод повертає спеціальний індикатор, який може використовуватися пізніше для завершення зчитування за допомогою методу **`clearWatch()`.**

Вдосконалене кешування сторінок

HTML5 розширює можливості браузерного кешування. Тепер web-сторінки можуть бути повністю доступні для користувачів навіть без підключення до інтернету.

Використання HTML5-кешування дає наступні переваги:

- можливість перегляду web-сторінок користувачами навіть без підключення до інтернету;
- збільшення швидкості завантаження сторінок - сторінки зберігаються локально на комп'ютері користувача і тому будуть завантажуватися набагато швидше.

Зменшення навантаження на сервер - серверу не доведеться обробляти деякі з запитів користувачів.

Декларування HTML5-кешування. Для того, щоб використовувати механізм кешування на web-сторінках, необхідно додати до тега <html> атрибут **`manifest`** і його значенням вказати шлях до спеціального файлу, в якому декларуються параметри кешування:

```
<html manifest="example.appcache">
```

Якщо цей атрибут не був заданий на web-сторінці і посилання на неї відсутнє у файлі кешування, сторінка не буде кешована. Файл кешу може мати будь-яке розширення (*наприклад*, `.appcache` або `.mf`), але зобов'язаний мати спеціальний тип MIME: "`text/cache-manifest`".

У деяких випадках web-серверу може знадобитися додаткова надбудова, щоб обслуговувати тип MIME. *Наприклад*, щоб налаштувати web-сервер Apache, необхідно додати наступний код у `.htaccess`-файл:

```
AddType text/cache-manifest .appcache
```

Вміст файлу кешування. Файл кешування є звичайним текстовим файлом, який вказує браузеру, які файли необхідно кешувати. Файл може містити три секції:

- CACHE MANIFEST - в даній секції зазначаються посилання на файли, які необхідно кешувати. Браузер буде автоматично кешувати всі перелічені файли відразу після першого завантаження;
- NETWORK - в цій секції зазначаються файли, які вимагають постійного підключення до інтернету. Браузер не буде кешувати файли, вказані в цій секції;
- FALLBACK – якщо файли, зазначені у цій секції, виявляться з якоїсь причини недоступними, користувачі автоматично будуть перенаправлятися на інший вказаний файл.

Секція CACHE MANIFEST обов'язково повинна бути присутня у всіх файлах кешування. Секції NETWORK і FALLBACK можуть бути відсутніми.

Оновлення кешованих файлів. Після того, як файли будуть кешованими, браузер продовжить показувати їх кешовану версію знову і знову, навіть якщо змінити вміст цих файлів на сервері. Для того, щоб оновити кешований вміст, необхідно зробити одну з наступних дій:

- очистити кеш в браузері користувача;
- оновити вміст файлу кешування.

Оновити кеш браузера програмно (за допомогою JavaScript).

Для оновлення вмісту файлу можна використовувати наступний прийом: замість того, щоб оновлювати безпосередньо вміст файлу, можна додати до нього коментар із

зазначенням дати зміни та/або версією файлу і в майбутньому змінювати вміст цього коментаря кожен раз, коли необхідно оновити кешований вміст:

Приклад:

```
CACHE MANIFEST
# 29.09.2013 v1.2
index.html
flower.png
somescript.js
```

Інтерактивні можливості

З появою мови JavaScript і об'єктної моделі DOM сайти стали **динамічними** (**інтерактивними**). З'явилася можливість змінювати вміст web-сторінки без її перезавантаження. Але тільки з появою HTML5 інтерактивність стала частиною мови HTML, в яку були додані окремі теги для створення інтерактивних елементів.

Парний блоковий тег **<menu>** призначений для створення меню, пункти якого дозволяють виконувати який-небудь код мовою JavaScript. Тег **<menu>** лише визначає саме меню. А пункти цього меню створюються з допомогою одинарного тега **<command>**:

```
  <menu><command onclick="alert('команда 1')"
label="Пункт перший"/>
  <command    onclick="alert('команда    2')"
label="Пункт другий"/>
  </menu>
```

Ім'я пункту меню міститься в атрибуті **label** тега <command>. Команда, яку буде виконано при клацанні на цьому пункті меню, розміщена в атрибуті **onclick** тега <command>. Замість тега <command> можна використовувати теги <li> або <button>.

Парний блоковий тег **<datagrid>** дозволяє створювати інтерактивні таблиці та списки. Такі таблиці можуть редагувати відвідувачі сайту: вилучати або створювати нові стовпці і рядки таблиці, правити вміст комірок, сортувати і згортати комірки.

Власне, тег `<datagrid>` лише вказує, що таблиця або список, розташований в ньому, є інтерактивним. Відповідно, всередині тега `<datagrid>` можна розміщувати теги:

- `<table>, <tr>, <td>, <th>, <caption>, <thead>, <tfoot>, <tbody>` - теги для створення таблиць;
- `<select>, <option>, <optgroup>` - теги для створення списків, що розкриваються.

Парний блоковий тег **`<details>`** призначений для виведення додаткової інформації, яка повинна відображатися тільки за бажанням відвідувача. За замовчуванням додаткова інформація спочатку прихована і відображається після клацання на показаній браузером виносці. Тегу `<details>` можна присвоїти атрибут `open` з однойменним значенням, щоб додаткова інформація відображалася відразу.

Питання для самоконтролю

1. Які переваги використання web -сховищ?
2. Який обсяг даних може бути збережений у сховищі?
3. Опишіть часові обмеження, які стосуються зберігання даних у web-сховищах.
4. Які браузери підтримують роботу із сховищами?
5. Як надати елемету властивість “переміщеня”?
6. Перелічіть події перетягування.
7. Який виграш отримують за рахунок застосування технології drag and drop?
8. Які методи дозволяють працювати з переміщуваними об’єктами?
9. Які можливості створює застосування технології web woker?
10. Назвіть місце зберігання тексту woker.
11. Перелічіть об’єкти, недоступні для технології web woker.
12. Як відбувається взаємодія web worker з основною web-сторінкою?
13. Як реалізується принцип геолокації?
14. Назвіть об’єкти і методи, які використовуються при геолокації.

15. Що передбачає аутентифікація?
16. Які властивості має об'єкт coords?
17. Які переваги дає HTML5-кешування?
18. Що зберігається у файлі кешування?
19. Для чого потрібна очистка кешу в браузері користувача?
20. У чому полягає принцип інтерактивності сайту?
21. Призначення тега <datagrid>.
22. Призначення тега <details>.

2.Таблиці стилів CSS

Загальні відомості

HTML структурує документ і повідомляє браузеру про функцію того чи іншого елемента. CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) - технологія управління зовнішнім виглядом елементів web-сторінок (видає браузеру інструкції про те, як вивести певний елемент – оформлення, розміщення, позиціонування тощо). Нині CSS використовується практично на всіх сайтах Всесвітньої павутини. Якщо HTML є "балками" й "цеглинами", які становлять структуру будинку, то CSS є "штукатуркою" і "фарбуванням" для його оздоблення.

CSS було запропоновано разом з HTML 4.0 насамперед для вирішення проблем з оформленням, що виникали у попередніх версіях HTML. Коли в HTML 3.2 були додані такі теги як `<font>` (нагадаємо, що цей тег використовувався для зміни шрифту елементів), оформлення і розмітка змішалися, внаслідок чого оформлення елементів стало займати чимало часу.

Для вирішення цієї проблеми наприкінці 1996 року W3C (Консорціум Всесвітньої Павутини, який займається розробкою web-стандартів) запропонував CSS. CSS дозволяє зберігати інформацію про оформлення HMTL-документа в окремому зовнішньому файлі з розширенням .css. Редагування лише одного цього файла створило можливість для швидкої зміни оформлення цілого web-сайту. Нині CSS є стандартом оформлення HTML-документів і підтримується всіма сучасними браузерами.

CSS використовується розробниками web-сторінок для задання кольорів, шрифтів, розташування та інших аспектів подання документа. Основною метою розробки CSS було розділення вмісту (написаного HTML або іншою мовою розмітки) та подання документа (написаного з використанням CSS). Це розділення може збільшити доступність документа, надати більшу гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність у структурі вмісту.

Нововведення CSS3

CSS3 - це новий стандарт оформлення HTML документів, який значно розширює можливості попереднього стандарту CSS2.1. Багато можливостей, які раніше були важкодоступні, оскільки вимагали залучення додаткових зовнішніх програм, зокрема, *Adobe Photoshop*, скриптів *JavaScript* або інших спеціальних "хитрощів" можуть легко досягатися у CSS3 завдяки новим властивостям оформлення.

У CSS3 є можливість:

- створювати елементи із згладженими кутами;
- створювати лінійні і сферичні градієнти;
- гнучко оформляти фон елементів;
- додавати до елементів і тексту різноманітні тіні;
- використовувати не лише"безпечні" шрифти (не боячись при цьому, що вони не будуть підтримуватися браузером користувача);
- створювати анімацію та різні ефекти переходів;
- задавати кольори кількома новими способами та ін.

Зауваження: нові елементи CSS3, розглянуті в цьому посібнику, підтримуються тільки в сучасних браузерах: *IE9+, Firefox 3.6+, Opera 10+, Chrome 12+, Safari 5+*. Особливі випадки будуть спеціально обумовлюватися.

Стиль – це все те, що визначає зовнішній вигляд HTML-сторінки при її відображенні у вікні браузера. За допомогою стилів можна змінювати більшість властивостей тегів HTML-сторінки.

Наприклад, для таблиці можна змінити шрифт, колір фону, товщину та колір меж, видимість рядків та комірок тощо. Форматування тега за допомогою стилів дещо нагадує зміну атрибутів тега, але можливості стилів набагато ширші.

Таблиця стилів – це шаблон, який керує форматуванням тегів HTML-сторінки. Концепція таблиці стилів для HTML-сторінки схожа на використання стилів у текстовому редакторі MS Word.

CSS дозволяє подати один і той же документ у різних стилях та пропонує різні методи виведення: екранне подання, друк, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), виведення пристроями, що використовують Шрифт Брайля (шрифт для незрячих).

CSS при відображенні сторінки може бути вибрана з різних джерел:

1. **Авторські стилі** (інформація стилів, що надається автором сторінки) у вигляді:

- **`зовнішніх таблиць стилів`**, тобто окремого файлу *.css*, на який робиться посилання в документі;
- **`вбудованих стилів`** - блоків CSS всередині самого HTML-документа;
- **`inline-стилів`**, коли в HTML-документі інформація стилю для окремого елемента вказується у його атрибуті `style`.

2. **Власні стилі**. Локальний файл *.css*, вказаний користувачем у налаштуваннях браузера, який опрацьовує авторські стилі, і застосовується до всіх документів.

3. **Стиль браузера**. Стандартний стиль, що використовується браузером за замовчуванням для подання елементів.

Стандарт CSS визначає пріоритети, за якими застосовуються правила стилю, якщо для якогось елемента підходять кілька правил одночасно. Це називається «каскадом», в якому для правил визначаються пріоритети (переваги), що робить результати передбачуваними. В наведеному вище списку стилі розміщені в порядку спадання пріоритетів.

2.1. Синтаксис CSS3. Селектори

Таблиці стилів складаються з набору правил. Кожне правило складається з одного або декількох **селекторів** і **блоку визначення**.

Блок визначення містить набір правил відображення документа і записується в фігурних дужках у вигляді однієї чи кількох пар виду "*властивість : значення*", які відокремлюються одна від одної крапкою з комою (;). Після останньої властивості крапка з комою необов'язкова.

CSS, як і HTML, ігнорує пробіли. Таблиці стилів можуть містити коментарі, які використовуються для створення пояснень. Коментарі ігноруються браузером при аналізі таблиць

стилів. В CSS коментар починається з "/*" і закінчується "*/", *наприклад*:

```
/* Правило перефарбує абзаци HTML-документа в зелений колір */
p {color : green;}
```

Селектор визначає, до яких елементів слід застосовувати вказані сукупності значень властивостей:

```
Селектор
{
властивість1 : значення1;
властивість2 : значення2;
властивість3 : значення3 значення4;
}
```

У цьому записі селектора можна побачити наступні деталі:

- селектор визначає елемент HTML, до якого буде застосовуватися правило, за назвою елемента, *наприклад*, `body` або іншими засобами, такими як значення атрибута `class`;
- властивості визначають, що планується робити з виділеними елементами: задавати колір елемента, колір фону, позицію, поля, заповнення, тип шрифту та ін.;
- значення залежать від властивості.

Наприклад, властивості, які впливають на колір, можуть набувати значення кольору, яке може бути записано у шіснадцятковій системі числення (*#ffff00),* вказано відповідно моделі RGB *rgb(12,13,22)* або записано у вигляді назви кольору (*red, green* чи *blue)*.

Властивості, які впливають на положення, поля, ширину, висоту тощо, можуть вимірюватися у *пікселях, em, %, сантиметрах або інших одиницях вимірювання*.

Приклад 1:

```
p {
margin : 5px;
font-family : Arial;
color : blue;
}
```

Це правило для кожного елемента <p> у документі HTML впливає на властивості, які визначають поля навколо параграфа

(margin), шрифт тексту в параграфі (font-family) і колір цього тексту (color). Як бачимо, поля встановлюються розміром у *5 пікселів*, шрифт задається *Arial*, а колір тексту - *blue*.

Приклад 2:

```
p {
font-family: "Garamond", serif;
}
h2 {
font-size: 110%;
color: red;
background: white;
}
note {
color: red;
background: yellow;
font-weight: bold;
}
p#paragraph1 {
margin: 0;
}
a:hover {
text-decoration: none;
}
#news p {
color: blue;
}
```


В цьому прикладі наведено шість правил із селекторами p, h2, .note, p#paragraph1, a:hover і #news p.

У перших двох правилах HTML-елементів <p> (параграфа) і <h2> (заголовка другого рівня) призначаються **стилі**. Параграфи будуть відображатися шрифтом *Garamond* або, якщо такий шрифт недоступний, яким-небудь іншим шрифтом із засічками (“*serif*”). Заголовок другого рівня буде відображатися *червоним на білому тлі із збільшеним кеглем*.

Третє правило буде застосоване до елементів, атрибут class яких містить слово “*note*”. *Наприклад*:

```
<p class="note">Цей параграф буде виведений напівжирним шрифтом червоного кольору на жовтому фоні.</p>
```

Четверте правило буде застосовуватися тільки до елементів <p>, атрибут id яких дорівнює *paragraph1*. Такі елементи не будуть мати зовнішніх відступів (margin).

П'яте правило визначає стиль hover для елементів <a>. За замовчуванням у більшості браузерів текст елементів <a> підкреслюється. Це правило вилучатиме підкреслення, коли вказівник миші знаходиться над цими елементами.

Останнє шосте правило застосовується для елементів <p>, які розміщені всередині елемента з атрибутом id, рівним "news".

Як **селектори** можна використовувати:

- **назву тега** – тоді стиль застосовується до всіх таких тегів.

Приклад:

```
a {font-size: 12pt; text-decoration: none;}
table {border: black solid 1px;}
```


Перший рядок цього CSS-коду задає всім посиланням *12-й розмір шрифту* і *вилучає підкреслення*. В другому рядку вказується, що у всіх таблиць *границя буде чорного кольору суцільна* (solid) *шириною 1 піксель*.

- **кілька тегів через кому** - тоді стиль буде застосовуватися для всіх перелічених тегів.

Приклад:

```
h1,h2,h3,h4,h5,h6{color:red;}/*робимо всі заголовки червоними*/
```

- **кілька тегів через пробіл**:

```
table a {font-size: 120%;}
```

Правило стосується всіх тегів <a>, вкладених в тег <table>. Розмір шрифту збільшиться на 20% від базового.

- **id елемента** (в стилях унікальний ідентифікатор вказується після знака #) - правила застосуються до тега з атрибутом id="ідентифікатор".

Приклад:

```
<html>
<head>
<style type='text/css'>
/* Оформляємо елемент з id='test1' */
#test1
{
color:green;
font-family:verdana;
font-size:1.2em;
}
</style>
</head>
<body>
<p id='test1'>Цей абзац буде оформлений з допомогою CSS. </p>
<p>А цей абзац змінюватись не буде. </p>
</body>
</html>
```

Зауваження: Не можна вносити в документ декілька елементів з однаковим `id`!

- **символ** `*` (маска) - правила застосуються до всіх елементів документа. В наступному *прикладі* стиль з певним кеглем шрифту застосовується до всіх дескрипторів, які розміщені на сторінці:

```
* {font-size: 14pt;}
```

Іншим символом маски є символ `>`. Таким знаком браузеру дається інструкція шукати дочірні селектори в межах певного батьківського. В наступному прикладі стиль застосовується тільки до елементів `<li>` списку `<ol>`:

```
ol > li {list-style-type: decimal;}
```

Використовуючи селектори класів, можна до одного і того ж дескриптора застосовувати різні стилі. Після загального селектора слідують крапка і ім'я класу. Тоді стиль застосовується до того дескриптора, чий атрибут `class` відповідає цьому імені. У наступному прикладі стиль

застосовується до всіх дескрипторів h2, у яких атрибут class дорівнює "myblue":

```
h2.myblue {background-color: blue;}
<h2 class="myblue">У цього заголовка синій фон.</h2>
```

Питання для самоконтролю

1. Охарактеризуйте каскадні таблиці стилів.
2. Які методи виведення даних підтримуються каскадними таблицями стилів?
3. Як визначено пріоритети, за якими застосовуються правила стилю?
4. Що таке селектор, блок визначення?
5. Яка форма запису блоку визначення?
6. Наведіть приклади використання селекторів.
7. Поясніть призначення *, >, # у селекторах.

2.2. Включення CSS у HTML-документ CSS-верстка

До появи CSS оформлення web-сторінок здійснювалося безпосередньо всередині вмісту документа. З появою CSS стало можливим розділення змісту і подання документа. За рахунок цього нововведення здійснюється застосування єдиного стилю оформлення для багатьох схожих документів, а також швидка зміна цього оформлення.

Переваги CSS-верстки:

1) Кілька дизайнів сторінки для різних пристроїв перегляду. *Наприклад*, на екрані дизайн буде розрахований на велику ширину, під час друку меню не буде виводитися, а на КПК і мобільному телефоні меню буде слідувати за вмістом.

2) Зменшення часу завантаження сторінок сайту за рахунок перенесення правил подання даних в окремий CSS-файл. В цьому випадку браузер завантажує тільки структуру документа та дані, збережені на сторінці, а подання цих даних завантажується браузером тільки один раз і кешується.

3) Простота подальшої зміни дизайну. Не потрібно правити кожну сторінку, а лише змінити CSS-файл.

4) Додаткові можливості оформлення. Зокрема, за допомогою CSS-верстки можна зробити блок тексту, який решту тексту буде обтікати (*наприклад*, для меню) або зробити так, щоб меню було завжди видно при скролінгу сторінок.

Зв'язані (зовнішні) стилі – найбільш оптимальний спосіб застосування CSS. В цьому випадку CSS-код зберігається в окремому файлі *.css*, а в HTML-документі робиться посилання на цей файл. Завдяки зв'язаним стилям досягаються такі переваги: структура HTML-документа стає наочнішою; файл із CSS-кодом скачується лише при першому відкриванні сайту, що збільшує швидкість відкривання його сторінок.

Після того як CSS-код створений, його потрібно підключити до HTML-документа. Для цього в тег `<head>` документа додають одинарний тег `<link>`:

```
<link rel="stylesheet" type="text/css"
href="адреса_стилю">
```

Для прикладу визначимо стиль, що задає чорний фон HTML-сторінки. HTML-код на сторінці матиме вигляд:

```
<head><link rel="stylesheet" type="text/css"
href="hi.css">
```

Правило стилю записане у файлі *hi.css*:

```
body {background-color: black;}
```

Особливістю запису правила стилю у файлі з розширенням *.css* є те, що тег `<style>` не використовується.

Основна перевага застосування зовнішніх стилів: один стиль може використовуватися відразу в декількох HTML-документах. У зовнішніх файлах зберігають стилі, які використовуються для всього сайту, вони можуть впливати відразу на декілька тегів у багатьох документах. Це стає зручним, якщо сайт містить багато сторінок.

Наприклад, необхідно поміняти на всіх сторінках сайту колір тла і розмір шрифту. Якщо всі сторінки підключають один і той же зовнішній стиль CSS, достатньо в ньому задати новий колір тла і розмір шрифту. Інакше доведеться редагувати кожну сторінку окремо. Коли ж на сайті кілька десятків або сотень сторінок, то це завдання стає трудомістким.

Якщо *css*-файл розміщений на іншому сайті, то необхідно вказати його абсолютну URL-адресу. Розглянемо *приклад*. Створимо файл *style.css*:

```
  class1 {text-decoration: underline; font-
size: 80%;}
  a.class1 {text-decoration: none;}
```

Тепер створимо саму сторінку `links.html`:

```
  <html>
  <head>
  <link    rel="stylesheet"    type="text/css"
href="style.css">
  </head>
  <body>
   <h1 class="class1">Мої улюблені сайти</h1>
  <a href="http://meta.ua" class="class1">
  meta.ua </a><br>
  <a href="http://google.com" class="class1">
  Google</a><br>
  <a href="http://ukr.net" class="class1">
ukr.net </a>
  </body>
  </html>
```

При відкриванні цієї сторінки браузер клієнта завантажить також файл *style.css* і застосує правила CSS до документа.

Нагадаємо, що використання CSS дозволяє розділити оформлення і вміст документа. У нашому прикладі правила оформлення розміщені у файлі *style.css*, а зміст – у файлі *links.html*. Таке розділеня істотно спрощує редагування сайту в подальшому. Рекомендується для оформлення використовувати тільки засоби CSS, тобто відмовитися від використання тегів `<font>`, `<s>`, `<u>`, `<center>` і атрибутів `align`, `border`, `color`, `height`, `width` тощо.

Внутрішні стилі (стилі рівня документа) задаються безпосередньо в самій HTML-сторінці і застосовуються до всього документа. Вони записуються всередині тега `<style>...</style>`, який вкладається в тег

`<head>...</head>`. Такий спосіб задання стилів використовується, коли потрібно застосувати однакові стилі відразу до багатьох HTML-елементів (тегів) в одному документі.

Наприклад:

```
<head><style type="text/css"> hr { width: 120px; }</style>
```

На цій HTML-сторінці всі лінії, визначені за допомогою тега `<hr>`, будуть мати довжину 120 пунктів.

Рядкові (вбудовані в тег документа) стилі використовуються тоді, коли потрібно вказати стилі конкретного елемента. Такий стиль інколи називають **інлайновим**. Рядковий стиль записується в атрибуті `style` і застосовується тільки до вмісту цього тега.

Наприклад:

```
<td style="background-color: black; color: red;"> Текст, розміщений в цій комірці, відображається червоним кольором на чорному фоні.</td>
```

Рядковий стиль має вищий пріоритет, ніж зовнішні стилі і стиль рівня документа. Однак бажано не використовувати такий спосіб задання стилю, оскільки він не відповідає принципу розділення змісту і оформлення. Вбудований стиль доцільно використовувати тільки для точкової (одноразової) "підгонки" дизайну сторінки.

Порядок застосування стилів

При роботі з CSS необхідно пам'ятати, що більш специфічні правила мають пріоритет над менш специфічними, *наприклад*:

- спосіб, зазначений в атрибуті `style`, перекриває спосіб, зазначений у тегу `<style>` або зовнішньому файлі CSS;
- селектор ID (#) має вищий пріоритет, ніж селектор класу (.), а той, в свою чергу, –вищий, ніж звичайний селектор тега;
- важливою особливістю CSS є те, що деякі атрибути успадковуються від батьківського елемента до дочірнього.

Наприклад, якщо атрибут `font-size` заданий для тега

`<body>`, то він успадковується усіма елементами на сторінці. Якщо властивість розміру задається у відсотках, вона буде обчислена виходячи із значення для батьківського елемента. Дізнатися, чи є атрибут успадкованим, можна в довіднику з атрибутів CSS (*наприклад*, за адресою http://htmlbook.ru).

Питання для самоконтролю

1. Визначення каскадних таблиць стилів.
2. Способи з'єднування таблиці стилів з HTML-сторінкою.
3. Ієрархія стилів.
4. Який стиль називають інлайновим?

2.3. Класи. Псевдокласи

Часто потрібно, щоб стиль не застосовувався до всіх тегів на сторінці, а тільки до деяких елементів (*наприклад*, не до всіх посилань на сторінці, а тільки до тих, які розташовані в меню сайту). Для цього використовуються **класи**:

```
ТЕГ.ім'я_класу {...}
```

Правила, зазначені після такого селектора, будуть діяти лише на теги з атрибутом

```
class="ім'я_класу":<ТЕГ class="ім'я_класу">
</ТЕГ>
```

Можна не вказувати ім'я тега, тоді правила будуть застосовуватися до всіх тегів з відповідним значенням атрибута `class`.

Селектор class

Цей вид селекторів дозволяє вибирати для оформлення не один елемент, а групу (клас) елементів. За допомогою атрибута `class` можна вказати, що елемент належить групі:

```
<p class="ім'я_класу">текст</p>
```

Для оформлення цього класу необхідно в таблицях стилів звернутися до імені класу, поставивши перед ним символ "."

Наприклад: `.ім'я_класу {color:red}`.

*Зауваження***:** ім'я класу та ідентифікатор можуть містити лише латинські букви і не можуть починатися з цифр. Розглянемо *приклад*:

```
<html>
<head>
<style type='text/css'>
/* Властивості будуть застосовані до
елементів з class='test1' */
.test1 {
color : green;
font-family : verdana;
font-size : 1.2em;
}
</style>
</head>
<body>
<p class='test1'> Абзац буде оформлено з
допомогою CSS. </p>
<p>А цей абзац змінений не буде. </p>
<p class='test1'>І цей абзац буде оформлено
з допомогою CSS. </p>
</body>
</html>
```

Результат виконання коду:

Абзац буде оформлено з допомогою CSS. (зелений)

А цей абзац змінений не буде.

І цей абзац буде оформлено з допомогою CSS. (зелений)

Псевдокласи

Псевдокласи є особливою групою, що дозволяє об'єднувати декілька стилів для якого-небудь об'єкта.

Наприклад, задамо властивості для першої літери параграфа. Для цього дескрипторові `p` призначаємо псевдоклас: `first-letter`, в якому встановлені різні стилі:

```
p : first-letter
```

```
{
float: right;
font-size: 2em;
color: red;
}
```

В CSS визначені наступні **псевдокласи**:

- `first-child` - перший дочірній елемент іншого елемента;
- `link` - ще не відвідані посилання;
- `visited` - відвідані посилання;
- `hover` - елемент, над яким в даний час знаходиться курсор;
- `active` - активний в даний момент елемент;
- `focus` - елемент, що має фокус введення;
- `lang` - цей псевдоклас визначає поточну мову;
- `first-line` - перший рядок абзацу;
- `first-letter` - перша буква абзацу;
- `before` - визначає вміст перед елементом;
- `after` - визначає вміст після елемента.

Наведемо *приклад* застосування псевдокласу `first-child`, який дозволяє вибрати елемент, що є першим нащадком батьківського елемента.

```
<html>
<head>
<style type='text/css'>
p:first-child
{
color:red;
}
</style>
</head>
<body>
<p>Це перший абзац, тому він буде
оформлений.</p>
```

```
<p>Це другий абзац, він не буде
оформлений.</p>
<p>Це третій абзац, він теж не буде
оформлений.</p>
</body>
</html>
```

Питання для самоконтролю

1. Класи у CSS.
2. Яке призначення селектора `class`?
3. Псевдокласи у CSS.
4. Приклади псевдокласів CSS.

2.4. Оформлення таблиць

Властивості CSS можуть застосовуватися до таблиць, їх рядків та комірок для задання властивостей тексту та шрифту, управління фоном, полями, границями, розмірами та ін. Розглянемо властивості, які використовуються для оформлення таблиць.

`width` - встановлює ширину таблиці. Ширина встановлюється у *пікселях або %*, але можна також використовувати *сантиметри і ет*.

`height` - дозволяє встановити висоту таблиці. Висота зазвичай вказується в *пікселях*, але можна використовувати *сантиметри і ет*.

`border` використовується для оформлення границь таблиці.

`text-align` – вирівнює текст табличних комірок по горизонталі.

Значення: `left, right, center`.

`padding` - встановлює величину відступу між межею комірки і її вмістом.

`vertical-align` – вирівнювання вмісту колонок.

Значення: `baseline` – вирівняти базову лінію елемента за базовою лінією батьківського об'єкта; `bottom` - вирівняти основу елемента за нижньою частиною елемента найнижчого рядка; `middle` - вирівняти середню точку елемента за базовою

лінією батьківського об'єкта плюс половина висоти батьківського об'єкта; `sub` - вирівняти елемент як нижній індекс; `super` - вирівняти елемент як верхній індекс; `text-bottom` - вирівняти нижню границю елемента за найнижчим краєм поточного рядка; `text-top` - вирівняти границю елемента за найвищим текстовим елементом поточного рядка; `top` - вирівняти верхній край елемента за верхом найвищого елемента рядка; числові значення; проценти.

Приклад: `vertical-align : bottom`

`border-collapse -` дозволяє вилучити подвійні границі навколо колонок таблиці. Значення: `collapse` — об'єднує суміжні границі колонок; `separate` — відображає суміжні границі колонок кожну окремо.

`border-spacing.` Якщо властивості `border-collapse` задано значення `separate`, з допомогою властивості `border-spacing` можна вказати відстань між границями комірок таблиці, причому, якщо задано два значення через пробіл, то перше визначає відстань по горизонталі, а друге — по вертикалі.

Синтаксис: `border-spacing`: *значення1* [*значення2*];

`caption-side -` дозволяє розмістити заголовок таблиці (вміст тега `<caption>`) до вехнього чи до нижнього краю таблиці. Значення: `top`, `bottom`.

Синтаксис: `caption-side : top | bottom | left | right;`

`empty-cells -` визначає спосіб відображення порожніх комірок таблиці. Значення: `show` — відобразити границю і фон порожньої комірки; `hide` — приховати границю і фон порожньої комірки або весь рядок, якщо в ній всі комірки порожні.

Синтаксис: `empty-cells : show | hide`

`table-layout` – задає спосіб обчислення ширини таблиці. Значення: `auto` — ширина колонок таблиці визначається браузером після аналізу їх вмісту; `fixed` — ширина колонок задається тегом `col` або на основі вмісту першого рядка

таблиці. Якщо з деяких причин визначити ширину колонок не вдається, таблиця ділиться на колонки рівної ширини.

Синтаксис: `table-layout : auto | fixed | inherit`

Створимо таблицю і застосуємо до неї CSS-стилі. Внесемо в таблицю дані про популярність браузерів. Для шапки таблиці використаємо тег `<th>...</th>`.

HTML-код може мати вигляд:

```
<html>
<head>
<title>Популярність браузерів у
світі</title>
</head>
<body>
<table>
<tr>
<th>Рік\Браузер</th>
<th>IE</th>
<th>Firefox</th>
<th>Safari</th>
<th>Opera</th>
</tr>
<tr>
<td>2010</td>
<td>61.43%</td>
<td>24.40%</td>
<td>4.55%</td>
<td>2.37%</td>
</tr>
<tr>
<td>2009</td>
<td>69.13%</td>
<td>22.67%</td>
<td>3.58%</td>
<td>2.18%</td>
</tr>
<tr>
<td>2008</td>
```

```
<td>77.83%</td>
<td>16.86%</td>
<td>2.65%</td>
<td>1.84%</td>
</tr>
<tr>
<td>2007</td>
<td>79.38%</td>
<td>14.35%</td>
<td>4.70%</td>
<td>0.50%</td>
</tr>
</table>
</body>
</html>
```

Поки що маємо таблицю без CSS-оформлення. За замовчуванням вміст комірок шапки відображається жирним шрифтом з вирівнюванням до центру. Додамо у тег `<head>...</head>` тег `style>...</style>`, а до тега `<table>...</table>` атрибут `id="browser_stats"`. Запишемо CSS-правила для таблиці. Для шапки встановимо сірий фон і відступ вмісту від границь (`padding`) у половину висоти рядка, для комірок з даними - вирівнювання до правого краю і `padding` три десятих від висоти рядка. Навколо таблиці задамо подвійну рамку, а для комірок - звичайну одинарну.

Код:

```
<style>
/* стиль таблиці */
TABLE#browser_stats {
 border: 3px double black;
}
/* стиль комірок  шапки */
TABLE#browser_stats TH{
 border: 1px solid black;
 background-color: gray;
 padding: 0.5em;
}
```

```
/* стиль комірок з даними */
TABLE#browser_stats TD{
 border: 1px solid black;
 padding: 0.3em;
 text-align: right;
}
</style>
```

В цьому варіанті таблиці є істотний недолік: у кожної комірки з'явилася власна рамка. Щоб цього не відбувалося, необхідно вказати в правилах для таблиці властивість `border-collapse` із значенням `collapse`.

Питання для самоконтролю

1. Які властивості використовують для встановлення розмірів таблиць?
2. Як оформити границі таблиці?
3. Перелічіть властивості, що стосуються вирівнювання вмісту таблиці.

2.5. Шрифт. Властивості шрифту

CSS надає наступні можливості управління шрифтом – змінювати його сімейство, кегель чи товщину. Все це дозволяє збільшити "блиск" і неповторність web-сторінок. Відзначимо, що шрифти відрізняються між собою зовнішнім виглядом, розміром, стилем (прямий, курсив), "жирністю". Розміри в CSS можна задавати в різних одиницях вимірювання: `em` – поточна висота шрифту; `pt` – пункти (типографська одиниця); `px` – піксель; `%` – відсоток.

CSS3-властивість @font-face

CSS3 надає розробникам повну свободу при виборі шрифту. У попередніх версіях CSS розробники були змушені використовувати тільки ті шрифти, які гарантовано встановлені на комп'ютері користувача. В CSS3 розробники можуть використовувати будь-які шрифти. Знайдений потрібний шрифт слід розмістити на web-сервері і підключити з допомогою нового CSS3-правила `@font-face`. Підключений шрифт буде

завантажений і відображається автоматично при відвідуванні сторінки користувачем.

Зауваження: браузери *IE9+, Chrome, Firefox, Opera і Safari* підтримують шрифти у форматі *.woff* (*Web Open Font Format - Відкритий Формат Шрифтів Всесвітньої Павутини*). Браузери *Chrome, Firefox, Opera і Safari*, крім того, підтримують шрифти у форматі *TTF* і *OTF*, а *IE* - у форматі *EOT*.

Синтаксис:

```
  @font-face {
font-family:opensans; /* Задаємо ім'я шрифту */
  src:url('opensans.woff') /* Вказуємо
місцезнаходження нашого шрифту */
  }
```

Для того, щоб потім використовувати підключений шрифт, необхідно просто додати до бажаного елемента властивість `font-family`, яка містить ім'я шрифту.

Приклад.

```
  <html>
  <head>
  <style type='text/css'>
  @font-face {
  font-family:"opensans"; /* Ім’я шрифту */
  src:url('opensans.woff') /* Розміщення
шрифту */
  }
  div
  {
  font-family:"opensans";
  font-size:1em;
  }
  </style>
  </head>
  <body>
  <div>Текст цього елемента написаний шрифтом
opensans. </div>
  </body>
  </html>
```

Крім того, можна додати курсивне або жирне накреслення підключеного шрифту. Для цього необхідно додати в `@font-face` властивість `font-style:italic` або `font-weight:bold` і вказати шлях до шрифту у відповідному накресленні.

Приклад.

```
<html>
<head>
<style type='text/css'>
@font-face {
font-family:"opensans"; /* Ім'я шрифту */
src:url('opensans.woff') /* Розміщення
шрифту */
}
/* Підключим курсивну версію шрифту */
@font-face {
font-family:"opensans";
font-style:italic;
src:url('opensans-italic.woff'),;
}
/* Підключим жирну версію шрифту */
@font-face {
font-family:"opensans";
font-style:bold;
src:url('opensans-bold.woff'),;
}
/* Застосуємо підключений шрифт до абзаців */
p {
font-family:"opensans";
font-size:1em;
}
</style>
</head>
<body>
<p>Для звичайного тексту буде використаний
шрифт opensans.</p>
```

```
<p><i>Для курсивного тексту буде
використаний шрифт opensans-italic.</i></p>
  <p><b>Для жирного тексту буде використаний
шрифт opensans-bold.</b></p>
  <p>А це звичайний текст із
<i>"вставками"</i> курсивного і <b>жирного</b>
тексту.</p>
  </body>
  </html>
```

Розглянемо інші властивості шрифтів у CSS.

font-family. Визначає сімейства шрифтів, яких через кому може бути вказано кілька. Значення: ім'я сімейства (імена, що складаються з кількох слів, беруться в лапки); ім'я типового шрифту, що застосовується для виведення тексту: *serif, sans-serif, fantasy і monospace*.

Приклад: `p {font-family: "Times New Roman", Courier, serif;}`

font-style. Визначає накреслення шрифту (курсив або похиле). Значення: *normal* - звичайне накреслення (за замовчуванням); *italic* - курсив; *oblique* - похиле накреслення виключно екранного шрифту (він має дещо менший нахил, ніж курсив і застосовується до шрифтів без зарубок).

Приклад: `p {font-style: italic;}`

font-variant. Визначає, як буде виведений шрифт: у вигляді малих чи великих літер. Значення: *normal* - звичайне накреслення (за замовчуванням); *small-caps* – створює капітель - відображає шрифт у вигляді малих прописних літер (не реалізоване для шрифтів кирилиці).

Приклад: `p {font-variant: small-caps;}`

font-weight. Визначає товщину виведеного шрифту. Значення: *normal* - звичайне накреслення (за замовчуванням); *lighter* - світлий шрифт; *bold* - напівжирне накреслення; *bolder* - жирний шрифт. 100-900 - число, яке вказує товщину шрифту, 100 відповідає найсвітлішій товщині - *lighter*, 400 - *normal*, 700 - *bold*, 900 – *bolder*. *Приклад:* `p {font-weight: bold;}`

font-size. Визначає кегель (висоту символів) шрифту. Значення: *абсолютний розмір: xx-small* (дуже малий), *small* (малий), *medium* (середній) (за замовчуванням), *large* (великий), *x-large* (дуже великий), *xx-large* (дуже великий); *відносний розмір: larger* (менше), *smaller* (більше); будь-яка відповідна стандарту *висота шрифту* (від'ємне значення не допускається); будь-яке відповідне стандарту *процентне значення*.

Абсолютний спосіб дозволяє задавати розмір шрифту в абсолютних одиницях, а саме: *пікселі (px)* чи *відсотки (%). Приклад:*

```
<html>
<head>
<style type='text/css'>
p.fz1 {font-size:20px;}
p.fz2 {font-size:30px;}
p.fz3 {font-size:13px;}
</style>
</head>
<p class='fz1'>Величина розміру шрифту цього абзацу 20 px.</p>
<p class='fz2'> Величина розміру шрифту цього абзацу 30 px.</p>
<p class='fz3'> Величина розміру шрифту цього абзацу 13 px.</p>
<br />
<b>Величина стандартного розміру шрифту 16 px.</b>
</html>
```

Відносний спосіб допомагає уникнути проблем з невідповідністю при відображенні сторінки в різних браузерах, оскільки всі розміри встановлюються відносно. Для задання розміру шрифту цим способом використовується одиниця `em`. `1em` еквівалентний розміру шрифту в браузері за замовчуванням і дорівнює `16px`. W3C рекомендує для задання шрифту використовувати саме цей спосіб.

Властивість font-size не варто використовувати для того, щоб оформляти абзаци як заголовки. Для визначення абзаців слід використовувати теги <p>, а для заголовків <h1>-<h6>.

Приклад:

```
<html>
<head>
<style type='text/css'>
p.fz1 {font-size:1.2em;}
p.fz2 {font-size:1.5em;}
p.fz3 {font-size:1.15em;}
</style>
</head>
<p class='fz1'> Розмір шрифту цього абзацу
1.2 em.</p>
<p class='fz2'> Розмір шрифту цього абзацу
1.5 em.</p>
<p class='fz3'> Розмір шрифту цього абзацу
1.15 em.</p>
<br />
<b>Величина стандартного розміру шрифту 16
пікселів.</b>
</html>
```

font. Зручна властивість для встановлення відразу всіх параметрів шрифту. Якщо якісь значення пропущені, використовується їх значення за замовчуванням. Значення: font-style - шрифт; font-variant - значення, що визначає виведення шрифту у вигляді малих прописних літер; font-weight - товщина; font-size - розмір; line-height - інтерліньяж; font-family - сімейство шрифтів. *Приклади:*

1)

```
p {font: oblique 12pt "Helvetica Nue",
serif; font-stretch: condensed;}
```

2)

```
<html>
<head>
<style type='text/css'>
p.italic {font-style:italic;}
```

```
p.bold {font-weight:bold;}
  </style>
  </head>
  <p class='italic'>Цей абзац написано
курсивом</p>
  <p class='bold'>Цей абзац написано жирним
шрифтом</p>
  </html>
```

Безпечні шрифти

Безпечними шрифтами називають шрифти, вірогідність підтримки яких на будь-якому комп'ютері з будь-якою встановленою ОС близька до 100%. Список безпечних шрифтів: *Arial; Arial Black; Courier New; Comic Sans MS; Georgia; Impact; Times New Roman; Trebuchet MS; Verdana*.

В CSS шрифти поділяються на такі "***сімейства***":

Serif - шрифти цього сімейства мають невеликі зарубки на кінцях символів. *Приклади* шрифтів цього сімейства: *Times New Roman, Georgia*.

Sans-Serif - шрифти цього сімейства не мають зарубок. Приклади: *Arial, Arial Black, Trebuchet MS, Verdana*.

Monospace - всі символи шрифтів цього сімейства займають однакову ширину. Приклад: *Courier New*.

Fantasy - сімейство "химерних" шрифтів, що використовуються в основному для створення барвистих заголовків. Приклад: *Impact*.

Cursive - сімейство шрифтів рукописного шрифту. Приклад: *Comic Sans MS*.

Питання для самоконтролю

1. У яких випадках ім’я сімейства шрифтів вказується в лапках?
2. Опишіть можливості `font-weight`.
3. Поясніть способи регулювання розмірів шрифту.
4. Поділ шрифтів на сімейства у CSS.

2.6. Оформлення тексту. Властивості для списків. Розбивання тексту на стовпці. Використання тіней

Текстова складова web-сторінки є головною. Тому текст повинен легко читатися, звертати на себе увагу і мати гарний вигляд. Все це в CSS забезпечують властивості тексту.

text-align. Вирівнювання тексту в елементі. Значення: *left, center , right, justify (по ширині)*,

inherit - застосовується значення батьківського елемента. Останнє значення використовується практично всіма властивостями, наведеними нижче.

Приклад: `p {text-align: center;}`

text-indent. Визначає довжину відступу в пікселях першого рядка блоку (абзацу).

Приклад: `p {text-indent: 20px;}`

text-decoration. Оформлення тексту. Значення: *none* - відсутність елементів оформлення (за замовчуванням); *underline* - підкреслення; *overline* - лінія над текстом; *line-through* - закреслення; *blink* – мерехтіння.

Приклад:

```
<html>
<head>
<style type='text/css'>
p.td1 {text-decoration:underline;}
p.td2 {text-decoration:line-through;}
p.td3 {text-decoration:overline;}
</style>
</head>
<p class='td1'>Текст цього елемента
підкреслений.</p>
<p class='td2'>Текст данного элемента
перечеркнут.</p>
<p class='td3'>Текст цього елемента під
лінією.</p>
</html>
```

text-transform. Переводить букви у верхній (нижній) регістр (не реалізовано для шрифтів кирилиці). Значення: *none* -

відсутність зміни регістра (за замовчуванням); *capitalize* - перетворює першу літеру кожного слова у верхній регістр; *uppercase* - переводить всі букви у верхній регістр; *lowercase* - переводить всі букви в нижній регістр.

Приклад: `p {text-transform: uppercase;}`

`letter-spacing`. Визначає інтервал між символами тексту. Значення: *none* - звичайний інтервал; будь-яка стандартна довжина інтервалу між буквами.

Приклад: `p {letter-spacing: 5px;}`

`word-spacing`. Визначає інтервал між словами. Значення: *none* - звичайний інтервал; будь-яка стандартна довжина інтервалу між словами.

Приклад: `p {word-spacing: 2px;}`

`white-space`. Визначає, як опрацьовувати пробіли. Значення: *normal* - пробіли згортаються, якщо це потрібно для розміщення елемента (за замовчуванням); *pre* - пробіли опрацьовуються так, як зазначено в коді (попередньо відформатований текст); *nowrap* - згортаються всі пробіли.

Приклад: `p {white-space: pre;}`

Колір тексту

З допомогою CSS-властивості `color` можна змінювати колір тексту HTML-елементів. Колір може бути заданий наступними способами:

1. З допомогою назви кольору (*наприклад*, `'red'` задає червоний колір);

2. За допомогою RGB значення (*наприклад*, `'rgb`(255,255,255)' - білий колір);

3. За допомогою шістнадцяткового значення (*наприклад*, '#00ff00' - зелений колір).

Перший спосіб зазвичай використовується для визначення основних кольорів, назви яких добре відомі.

Наприклад, `green` визначає зелений колір, `blue` - синій, `white` - білий.

Другий спосіб може використовуватися для визначення будь-яких кольорів і відтінків.

Синтаксис: `rgb`(червоний, зелений, блакитний)

червоний - число від 0 до 255 вказує, як багато червоного буде в остаточному відтінку;

зелений - число від 0 до 255 вказує, як багато зеленого буде в остаточному відтінку;

блакитний - число від 0 до 255 вказує, як багато блакитного буде в остаточному відтінку.

Наприклад, `rgb`(255,0,0) задасть червоний колір, а `rgb`(0,255,0) зелений. Змішуючи червоний з зеленим `rgb`(255,255,0), отримаємо жовтий.

Третій спосіб за функціональністю еквівалентний другому, але запис більш компактний. На практиці в основному використовують саме цей спосіб.

Синтаксис: `#червонийзеленийблакитний`

червоний - шістнадцяткове число від 0 до ff вказує, як багато червоного буде в остаточному відтінку;

зелений - шістнадцяткове число від 0 до ff вказує, як багато зеленого буде в остаточному відтінку;

блакитний - шістнадцяткове число від 0 до ff вказує, як багато блакитного буде в остаточному відтінку.

Наприклад, #ff0000 задасть червоний колір, а #00ff00 зелений. Змішуючи червоний і зелений #ffff00, отримаємо жовтий.

Тепер спробуємо "перефарбувати" абзаци в зелений колір усіма перерахованими вище способами:

```
p {color : green;}
p {color : rgb(0,255,0);}
p {color : #00ff00;}
```

Додавання тіні до тексту

За допомогою нової CSS3-властивості **`text-shadow`** можна додавати до тексту елементи тіні (до тексту одного елемента може бути додано одночасно кілька тіней). При встановленні тіні для тексту необхідно вказати: величину зміщення тіні від тексту по горизонталі і вертикалі (може бути від'ємною), а також радіус розмиття і колір тіні.

Приклад.

```
<html>
<head>
<style type='text/css'>
#shadow1,#shadow2,#shadow3 {
font-size:2.5em;
font-family:Arial;}
#shadow1 {
text-shadow:3px 2px #FFAE00;}
#shadow2 {
text-shadow:1px 1px 10px #FFAE00;}
#shadow3 {
text-shadow:2px 2px 2px #FFAE00, 2px 2px
15px #1435AD;}
</style>
</head>
<body>
<p id="shadow1">Приклад простої тіні</p>
<p id="shadow2">Приклад тіні з размиттям</p>
<p id="shadow3">Декілька тіней одночасно</p>
</body>
</html>
```

Властивість `text-overflow`

У CSS3 було додано нову властивість `text-overflow`, яка дозволяє вказати, що має статися з текстом, що вийшов за межі границь елемента.

Приклад.

```
<html>
<head>
<style type='text/css'>
#wrap1 {
width:450px;
border:1px #000 solid;
text-overflow:ellipsis;
overflow:hidden;
white-space:nowrap;
```

```
}
#wrap2 {
width:450px;
border:1px #000 solid;
text-overflow:clip;
overflow:hidden;
white-space:nowrap;
}
</style>
</head>
<body>
<p id="wrap1"><b>text-overflow:ellipsis</b>
- Текст, що виходить за межі батьківського
елемента</p>
<p id="wrap2"><b>text-overflow:clip</b> -
Текст, що виходить за межі батьківського
елемента</p>
</body>
</html>
```


Властивість word-wrap

За допомогою нової CSS3-властивості **word-wrap** можна вказати, що довгі слова, які виходять за межі границь елемента, повинні розділятися і переноситися на новий рядок.

Приклад.

```
<html>
<head>
<style type='text/css'>
#wrap1{
width:200px;
height:150px;
padding:10px;
border:1px #000 solid;}
#wrap2{
width:200px;
height:150px;
padding:10px;
```

```
border:1px #000 solid;
word-wrap:break-word;}
</style>
</head>
<body>
<p id="wrap1">Цей текст звичайний, але він
має <b>дуууууууууже довге слово</b>, яке
виходить за межі його вмісту.</p>
<p id="wrap2">З допомогою нової CSS3-
властивості word-wrap:break-word можна
розділити <b> дуууууууууже довге слово</b> і
перенести його на новий рядок.</p>
</body>
</html>
```

Властивості для списків

list-style-type. Дозволяє управління нумерованими і ненумерованими списками. Значення: за замовчуванням `disc` (для <ul>) і `decimal` (для <ol>); `circle`, `disc`, `square` — маркер у вигляді кола, точки, квадрата відповідно; `decimal` — арабські числа (1, 2, 3, ...); `lower-alpha` — малі латинські букви (a, b, c, ...); `upper-alpha` — великі латинські букви (A, B, C, ...); `upper-roman` — римські числа у верхньому регістрі (I, II, III, IV,…); `none` - приховати маркер.

Приклад:

```
<html>
<head>
<style type='text/css'>
ul.lis1 {list-style-type:square;}
ol.lis2 {list-style-type:upper-roman;}
</style>
</head>
<body>
<p> Найбільші міста України: </p>
<ul class='lis1'>
<li>Київ</li>
```

```
<li>Харків</li>
<li>Одеса</li>
</ul>
<i> Маркер списку має вигляд квадрата (за замовчуванням має вигляд круга). </i>
<p> Міста України,відсортовані за чисельністю населення: </p>
<ol class='lis2'>
<li>Київ</li>
<li>Харків</li>
<li>Одеса</li>
</ol>
<i> Елементи списку нумеруються римськими цифрами (за замовчуванням арабськими). </i>
</body>
</html>
```

list-style-image – використовується тоді, коли не підходить жоден із маркерів, запропонованих попередньою властивістю `list-type`. Значення: `none`; `url`('шлях до файлу').

Приклад: `ul { list-style-image: url("../img/marker.png");}`

list-style-position – вказує розміщення маркера відносно тексту. Значення: `outside` (за замовчуванням) - маркер виноситься за межі тексту ; `inside` - дозволяє розмістити маркер в текстовому масиві.

Приклад: `li { list-style-position: inside;}`

Розбивання тексту на стовпці

CSS3 має вбудовані властивості, за допомогою яких можна розбивати текст елементів на кілька стовпців.

Зауваження: ці властивості підтримуються в браузерах *IE10+* і *Opera*. Для браузерів *Chrome* і *Safari* перед властивістю потрібно додати префікс -**webkit**, а для *Firefox* префікс -**moz**. З допомогою CSS3-властивості **column-count** можна вказати

кількість стовпців, на які необхідно розбити текст вибраного елемента.

Приклад.

```
<html>
<style type='text/css'>
h1 {border-bottom:1px solid;}
div {
  column-count:3;
  -moz-column-count:3;
  -webkit-column-count:3;}
</style>
<body>
<h1>Титанік</h1>
<div> На Титаніку було 8 сталевих палуб, розташованих одна над одною на відстані 2,5—3,2 м. Верхня палуба була шлюпкова, під нею було сім інших, позначених зверху вниз літерами від A до G.Тільки палуби C, D, E і F проходили по всій довжині судна. Шлюпкова палуба і палуба A не доходили ні до носової частини, ні до корми, а палуба G розташовувалася тільки в передній частині лайнера — від котельних відділень до носа і в кормовій частині — від машинного відділення до зрізу корми. На відкритій шлюпкової палубі розміщувалися 20 рятувальних шлюпок, уздовж бортів перебували прогулянкові палуби.
</div>
</body>
</html>
```

З допомогою CSS3-властивості **column-gap** можна встановити величину інтервалу між стовпцями тексту. Властивість **column-rule** дає можливість задати ширину, колір і стиль оформлення простору між стовпцями. CSS3-властивість **column-width** дозволяє вказувати ширину стовпців тексту.

Приклад.

```
<html>
<style type='text/css'>
h1 {border-bottom:1px #7F0055 solid;}
div {
  column-count:4;
  -moz-column-count:4;
  -webkit-column-count:4;
  column-rule:2px dotted #7F0055;
  -moz-column-rule:2px dotted #7F0055;
  -webkit-column-rule:2px dotted #7F0055;}
</style>
<body>
<h1>Титанік</h1>
<div> На Титаніку було 8 сталевих палуб, розташованих одна над одною на відстані 2,5—3,2 м. Верхня палуба була шлюпкова, під нею було сім інших, позначених зверху вниз літерами від A до G.Тільки палуби C, D, E і F проходили по всій довжині судна. Шлюпкова палуба і палуба A не доходили ні до носової частини, ні до корми, а палуба G розташовувалася тільки в передній частині лайнера — від котельних відділень до носа і в кормовій частині — від машинного відділення до зрізу корми. На відкритій шлюпкової палубі розміщувалися 20 рятувальних шлюпок, уздовж бортів перебували прогулянкові палуби.
</div>
</body>
</html>
```

Питання для самоконтролю

1. Перелічіть можливі значення у `text-decoration`.
2. Опишіть способи призначення кольору тексту у CSS.
3. Як додати тінь до тексту засобами CSS?
4. Назвіть властивості управління списками.
5. Перелічіть властивості керування стовпцями тексту.

2.7. Використання кольору і фону

Присвоєння різним елементам колірних і фонових значень допомагають створити "видовищну" web-сторінку. В CSS є чимало властивостей, які нададуть web-сторінці необхідну привабливість.

color. Визначає колір тексту. Значення: будь-яке відповідне стандарту значення кольору.

Приклади:

```
p {color: red;}
p {color: rgb(255, 0, 0);}
p {color: #FF0000;}
```

background-color. Визначає колір фону. Значення: будь-яке стандартне значення кольору; `transparent` - фон елемента робиться прозорим (за замовчуванням).

Приклади:

1) `body {background-color: black;}`

2)

```
<html>
<head>
<style type='text/css'>
body {background-color:green;}
</style>
</head>
<body>
<h2 style='color:white;'>З допомогою CSS
колір фону сторінки змінено на зелений</h2>
</body>
</html>
```

background-image. Визначає фонове зображення елемента. Значення: будь-яке відповідне стандарту значення URL фонового зображення.

Приклад:

```
h1 {background-image : url(pictures.gif);}
```

background-repeat. Визначає напрямок, за яким екран буде заповнюватися копіями фонового зображення. Значення: `repeat-x` - фонове зображення повторюється по горизонталі;

`repeat-y` - фонове зображення повторюється по вертикалі; `no-repeat` - фонове зображення не повторюється. *Приклад:*

```
body {background-image: url(pictures.gif);
background-repeat: repeat-x;}
```

`background-attachment.` Визначає: фонове зображення буде зафіксовано у вікні браузера чи буде прокручуватися разом з документом (за замовчуванням).

Приклад:

```
body {background-image: url(pictures.gif);
background-attachment: fixed;}
```

`background-position.` Визначає положення зображення відносно верхнього лівого кута. Одиночне значення встановлює відстань по горизонталі і вертикалі. Зміщення за замовчуванням дорівнює *50%*. Можливе поєднання декількох ключових слів.

Значення:

– перше стандартне значення довжини - відстань по горизонталі від лівого краю елемента до фонового зображення; допускаються і від'ємні значення;

– друге стандартне значення довжини - відстань по вертикалі від лівого краю елемента до фонового зображення; допускаються і від'ємні значення;

– перше стандартне відсоткове значення - відношення (у %) зміщення фонового зображення від лівого краю по горизонталі до довжини елемента; за замовчанням 0% (верхній лівий кут);

– друге стандартне відсоткове значення - відношення (у %) зміщення фонового зображення від верхнього краю по вертикалі до висоти елемента;

– горизонтальне зміщення задається ключовими словами `left, right, center`;

– вертикальне зміщення вказується ключовими словами `top, bottom, center`.

Приклад:

```
body {background-image: url(pictures.gif);
background-position: center;}
```

`background.` Властивість для встановлення **відразу всіх параметрів фону**. Якщо певні параметри будуть пропущені, то

їх значення беруться за замовчуванням. Значення:

`background-color` - значення кольору фону;

`background-repeat` - значення повторення фонового зображення;

`background-attachment` - значення фіксації фонового зображення;

`background-position` - положення фонового зображення.

Приклад:

```
body {background: чорний url(graf/bomba.jpg)
center no-repeat;}
```

Задання кольору з допомогою HSL

У CSS3 було додано кілька нових способів задання кольору. Колір тут може задаватися **за моделлю HSL** (відтінок, насиченість, яскравість). Для того, щоб задати колір цим способом, необхідно вказати:

1) **відтінок** кольору в градусах повороту колірного круга (0 градусів - червоний, 120 градусів - зелений, 240 градусів - блакитний тощо);

2) **насиченість** кольору у відсотках (при зниженні відсотків колір буде блякнути);

3) **яскравість** кольору також у відсотках (0% - темний, 100% - світлий).

Приклад.

```
<html>
<style type='text/css'>
#wrap1,#wrap2,#wrap3 {
border:1px #000 solid;
width:230px;
height:120px;
margin:10px;
float:left;
padding:10px;
font-size:1.5em;
color:#000;
text-decoration:underline;
```

```
}
#wrap1 {background-color:hsl(0,30%,50%);}
#wrap2 {background-color:hsl(120,100%,80%);}
#wrap3 {background-color:hsl(240,100%,50%);}
</style>
<body>
<div id="wrap1">hsl(0,30%,50%)</div>
<div id="wrap2">hsl(120,100%,80%)</div>
<div id="wrap3">hsl(240,100%,50%)</div>
</body>
</html>
```

Задання кольору з допомогою RGBA

Цей спосіб дозволяє визначати колір і прозорість одночасно. Спочатку необхідно вказати значення RGB, а потім значення прозорості (*0* - максимальна прозорість, *1* - мінімальна прозорість).

Зауваження: задання прозорості за допомогою RGBA відрізняється від дії властивості `opacity` тим, що `opacity` робить прозорим цей елемент і всі його елементи-нащадки, а RGBA робить прозорим тільки цей елемент.

Приклад.

```
<html>
<style type='text/css'>
#exbody {
background-image:url(mountimg1.jpg);
background-repeat:no-repeat;
background-size:500px 370px;
}
#wrap1,#wrap2 {
width:200px;
height:120px;
margin:10px;
float:left;
padding:10px;
font-size:1.5em;
background-color:black;
```

```
color:white;
z-index:100;
}
#wrap1 {background-color:rgba(0,0,0,0.6);}
#wrap2 {opacity:0.6;}
</style>
<body id="exbody">
<div id="wrap1">Прозорість задана з
допомогою RGBA.</div>
<div id="wrap2">Прозорість задана з
допомогою opacity.</div>
</body></html>
```

Задання кольору з допомогою HSLA

Аналогічно до способу RGBA, колір разом з прозорістю можна задавати ще й з використанням HSLA.

Приклад.

```
<html>
<style type='text/css'>
#exbody {
background-image:url(mountimg2.jpg);
background-repeat:no-repeat;
background-size:500px 370px;
}
#wrap1,#wrap2 {
width:200px;
height:120px;
margin:10px;
float:left;
padding:10px;
font-size:1.5em;
background-color:black;
color:white;
z-index:100;
}
#wrap1 {
background-color:hsla(0,100%,0%,0.6);}
```

```
#wrap2 {opacity:0.6;}
</style>
<body id="exbody">
<div id="wrap1"> Прозорість задана з
допомогою HSLA.</div>
<div id="wrap2"> Прозорість задана з
допомогою opacity.</div>
<div id="under"></div>
</body>
</html>
```

Використання фону. Розмір фонового зображення

CSS3 надає кілька нових властивостей, які дозволяють більш гнучко управляти фоновими зображеннями. Зокрема, в CSS3 є можливість встановлювати розмір фонових зображень за допомогою властивості `background-size`. Розмір фонових зображень може бути вказаний у пікселях або у відсотках.

Приклад.

```
<html>
<head>
<style type='text/css'>
#wrap1,#wrap2,#wrap3
{
padding:10px;
border:1px #000 solid;
height:250px;
width:300px;
float:left;
margin:10px;
background-repeat:no-repeat;
}
#wrap1
{
background-image:url("spider2.gif");
background-size:150px 250px;
}
#wrap2
{
```

```
  background-image:url("spider2.gif");
  background-size:70% 70%;
  }
  </style>
  </head>
  <body>
  <div id="wrap1"></div>
  <div id="wrap2"></div>
  <p style="clear:both;"><b>Зверніть
увагу:</b> перше значення властивості
background-size задає ширину, а друге -
висоту фонового зображення.</p>
  </body>
  </html>
```

Кілька фонових зображень у CSS3

CSS3 розширює можливості властивості background-image: тут один елемент може мати кілька фонових зображень одночасно.

Приклад.

```
  <html>
  <head>
  <style type='text/css'>
  #wrap1 {
  background-
image:url(wislink.gif),url(mountimg3.jpg);
  background-position:bottom right, center;
  background-size:150px 40px,100% 100%;
  background-repeat:no-repeat,no-repeat;
  border:3px #000 solid;
  height:270px;
  width:350px;
  border-radius:15px;
  float:left;
  margin-right:40px;}
  </style>
  </head>
```

```
<body>
<div id="wrap1"></div>
<ol>
<li>Для того, щоб вставити в елемент декілька фоновых малюнків, необхідно вказати шлях до них через кому у властивості background-image.</li>
<li>Малюнки будуть накладатись один на одного у вказаній черговості(тобто перший вказаний малюнок буде відображатись над наступними).</li>
<li>Для того, щоб застосувати до фонових малюнків властивості оформлення, необхідно перерахувати необхідні значення в потрібному порядку через кому (наприклад, у властивості "background-position:bottom right, center;" bottom right буде застосовано до першого, а center - до другого фонового малюнка).</li>
</ol>
</body>
</html>
```

Властивість `background-origin`

За допомогою нової CSS3-властивості `background-origin` можна встановити, як повинно визначатись положення елемента щодо меж його батьківського елемента. Ця властивість може мати 3 різних значення:

- `border-box` положення елемента визначається відносно верхнього лівого кута межі елемента;
- `padding-box` положення елемента визначається відносно верхнього лівого кута блоку `padding`;
- `content-box` положення елемента визначається відносно верхнього лівого кута вмісту.

Приклад.

```
<html><head>
<style type='text/css'>
```

```
#wrap1,#wrap2,#wrap3 {
padding: 10px;
border: 7px #000 dotted;
height: 150px;
width: 200px;
float: left;
margin: 10px;
background-repeat: no-repeat;}
#wrap1 {
background-origin: border-box;
background-image: url("border-box.png");}
#wrap2 {
background-origin: padding-box;
background-image: url("padding-box.png");}
#wrap3 {
background-origin:content-box;
background-image:url("content-box.png");}
</style>
</head>
<body>
<div id="wrap1"></div>
<div id="wrap2"></div>
<div id="wrap3"></div>
<p style='clear:both;'><b>Зверніть увагу:</b> для елементів задана властивість padding:10px, щоб різниця між padding-box і content-box була помітною.</p>
</body>
</html>
```


Питання для самоконтролю

1. Які властивості CSS призначені для роботи з фоном?
2. Налаштування фонових зображень у CSS.
3. Нові способи задання кольору у CSS3.
4. Як встановити кілька фонових зображень одночасно?
5. Яким чином визначають положення елемента відносно батьківського у CSS3?

2.8. Відображення і розміщення елементів. Границі. Колір границь

Приховування елементів. В CSS приховати елементи можна за допомогою властивості **visibility:hidden** або за допомогою властивості **display:none.** Елемент прихований першим способом буде невидимий, але, як і раніше, займатиме місце на сторінці. Другий спосіб дозволяє повністю приховати елемент, щоб він більше не займав місця на сторінці.

Приклад:

```
<html>
<head>
<style type='text/css'>
#dis1 {display:none;}
</style>
</head>
<body>
<p id='dis1'>Цей абзац невидимий і не займає
місця.</p>
<p>Це другий абзац.</p>
<b>Перший абзац невидимий і він не займає
місця.</b>
</body>
</html>
```


Відображення елементів

З допомогою CSS можна встановити, як будуть відображатися елементи. В CSS зустрічаються два типи елементів за способом відображення:

- **блочні** елементи повністю займають всю ширину батьківського елемента. *Приклади* блочних елементів: `<p>, <h1>-<h6>, <div>`.
- **рядкові** елементи займають тільки необхідну ширину. *Приклади* рядкових елементів: `<a>, <span>.`

Розміщення елементів

Властивості позиціонування дозволяють розміщувати елементи там, де це необхідно розробнику.

Розташування елементів в CSS задається з допомогою таких властивостей:

- `top` - встановлює величину зміщення поточного елемента від верхнього краю батьківського елемента;
- `bottom` - встановлює величину зміщення поточного елемента від нижнього краю батьківського елемента;
- `left` - встановлює величину зміщення поточного елемента від лівого краю батьківського елемента;
- `right` - встановлює величину зміщення поточного елемента від правого краю батьківського елемента.

Описані вище властивості не вступлять в силу, поки не буде встановлений спосіб розміщення.

Спосіб розміщення визначає поведінку цих властивостей.

В CSS існує 4 різних способи розміщення елементів: **статичне, фіксоване, відносне** та **абсолютне**.

Статичне розміщення. Статичні елементи завжди з'являються там, де вони були оголошені. CSS-властивості `top`, `bottom`, `left` і `right` не працюють із статичними елементами. За замовчуванням всі елементи розміщуються цим способом.

Оголосити елемент статичним можна з допомогою `position : static`.

Фіксоване розміщення. Фіксовано розміщені елементи не змінюють свого розташування навіть при прокрутці вікна браузера. До фіксовано розміщених елементів можуть застосовуватися CSS-властивості `top`, `bottom`, `left`, `right`.

Елемент може бути оголошений фіксовано розміщеним за допомогою `position : fixed.`

Приклад:

```
<html>
```

```
<head>
<style type='text/css'>
#pos1
  {
  position : fixed;
  right : 40px;
  top : 17px;
  border : 1px solid;
  background-color : pink;
  }
</style>
</head>
<body>
<p id='pos1'>Цей абзац зафіксований.</p>
<p> Спробуйте прокрутити вікно. </p>
<br />
<br />
<p> Спробуйте прокрутити вікно. </p>
<br />
<br />
<p> Спробуйте прокрутити вікно. </p>
<br />
<br />
<p> Спробуйте прокрутити вікно. </p>
<br />
<br />
<p> Спробуйте прокрутити вікно. </p>
<br />
<br />
<p> Спробуйте прокрутити вікно.</p>
<br />
<br />
</body>
</html>
```

Відносне розміщення. Відносно розміщені елементи розміщуються відносно їх звичайної позиції.

Елемент може бути оголошений відносно розміщеним за допомогою `position: relative`.

Абсолютне розміщення. Абсолютно розміщені елементи розташовуються відносно першого батьківського елемента, спосіб розміщення якого відрізняється від статичного. Якщо такі елементи не були знайдені, то елемент буде розташований відносно базового елемента (`html`). Іноді для того, щоб домогтися бажаного ефекту, батьківський елемент спеціально визначається як відносно розміщений з нульовим зміщенням.

Оголосити елемент абсолютно розміщеним можна за допомогою `position:absolute.`

Накладання елементів. При застосуванні властивостей позиціонування елементи можуть накладатися один на одного. Властивість **`z-index`** дозволяє встановити, який елемент у випадку накладання буде згори, а який знизу. Елементи з більшим значенням властивості `z-index` розташовуються вище інших.

Зауваження: властивість `z-index` може набувати від'ємних значень.

Приклад:

```
<html>
<head>
<style type='text/css'>
#pos1
  {
  position:absolute;
  top:0px;
  left:100px;
  border:1px solid;
  width:80px;
  height:80px;
  background-color:red;
  font-size:4em;
  z-index:10;
  }
#pos2
```

```
    {
    position:absolute;
    top:36px;
    left:130px;
    border:1px solid;
    width:80px;
    height:80px;
    background-color:blue;
    font-size:4em;
    z-index:5;
    }
  #pos3
    {
    position:absolute;
    top:10px;
    left:170px;
    border:1px solid;
    width:80px;
    height:80px;
    background-color:pink;
    font-size:4em;
    z-index:-1;
    }
  </style>
  </head>
  <body>
  <p id='pos1'>10</p>
  <p id='pos2'>5</p>
  <p id='pos3'>-1</p>
  <p>Елемент з <i>z-index:10</i> - червоний,з
<i>z-index:5</i> - синій,а з <i>z-index:-1</i>
- рожевий.</b></p>
  </body>
  </html>
```

Таблця 2.1.

CSS-властивості, пов'язані з розміщенням

Властивості	**Опис**	**Значення**

`clip`	Обрізує елемент, розміщений абсолютно	`rect`
		`auto`
		`inherit`

`cursor`	Задає вигляд, якого буде набувати курсор при наведенні на елемент	`auto`
		`crosshair`
		`default`
		`pointer`
		`move`
		`e-resize`

		`ne-resize`
		`nw-resize`
		`n-resize`
		`se-resize`
		`sw-resize`
		`s-resize`
		`w-resize`
		`text`
		`wait`
		`help`

`overflow`	Встановлює, як повинен відображатися вміст, що виходить за межі елемента	`auto`
		`hidden`
		`scroll`
		`visible`
		`inherit`

Вирівнювання елементів

Вміння вирівнювати елементи CSS необхідне для створення якісних web-сторінок.

Центрування за допомогою **`margin`**. Блокові елементи можуть бути вирівняні до середини встановленням `margin` з

лівої і правої сторони значення *auto*.

Зауваження: якщо ширина блокового елемента дорівнює 100%, то він не буде вирівняний (оскільки доступного `margin` немає).

Вирівнювання за допомогою властивостей позиціонування.

Приклад:

```
<html>
<head>
<style type='text/css'>
.ali1 {
position:absolute;
right:0px;
}
.ali2 {
position:absolute;
left:0px;
}
.ali3 {
position:absolute;
top:100px;
left:230px;
}
</style>
</head>
<body>
<p class='ali1'>Цей абзац вирівняно до
правого краю.</p>
<p class='ali2'>Цей абзац вирівняно до
лівого краю.</p>
<p class='ali3'>Цей абзац вирівняно до
середини.</p>
</body>
</html>
```


Властивість **`float`** також дозволяє вирівнювати елементи CSS. Елемент, вирівняний за допомогою `float`, буде притиснутий до лівої або правої межі батьківського елемента (залежно від заданого значення) і змусить наступні за ним

елементи "обтікати" його з протилежного боку. Властивість `float` часто використовують з малюнками, але вона буває корисною і при звичайному вирівнюванні. HTML-елементи, які слідують за елементом із заданим `float`, будуть його "обтікати". Щоб уникнути цього, використовують властивість **`clear`**.

Блокова модель

Всі елементи в CSS є прямокутними блоками (рис. 2.1).

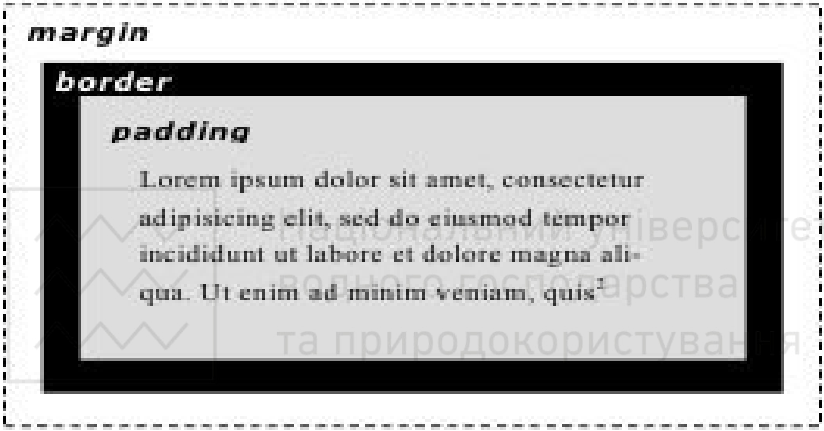


Рис 2.1. Бокс (box) елемента

Кожен такий блок має зону **`content`**, в якій розташовується вміст елемента (тобто текст, зображення тощо). Навколо зони `content` можуть розташовуватися необов'язкові зони: `padding`, `border` та `margin`. Використання цих властивостей допомагає задавати вигляд сторінки.

Зона **`padding`** (заповнення) оточує зону `content`. Ця зона використовується для визначення величини відступу вмісту елемента (зона `content`) від його межі (зона `border`). Зона може бути розбита на чотири частини, які можуть оформлюватися незалежно одна від одної: `padding-top`, `padding-right`, `padding-bottom`, `padding-left`.

Зона **`border`** (границя) оточує зону `padding`. Вона дозволяє задати елементу границю довільної ширини, стилю і

кольору. Зона може бути розбита на: `border-top`, `border-right`, `border-bottom`, `border-left`.

Зона **`margin`** (поля) оточує зону `border`. Ця зона дозволяє задати величину зовнішнього відступу даного елемента від оточуючих. Зона може бути розбита на: `margin-top` `margin-right`, `margin-bottom`, `margin-left`.

`margin-top`, `margin-right`, `margin-bottom`, `margin-left`. Встановлює ширину поля для певної сторони елемента (ці властивості встановлюють ширину відповідно до верхнього, правого, нижнього і лівого поля). Значення: будь-яка відповідна стандарту довжина - число, яке задає ширину поля (*за замовчуванням 0*); будь-яке відповідне стандарту процентне значення - відношення ширини поля до ширини елемента (для лівого і правого поля) чи до висоти (для верхнього і нижнього поля).

`margin`. Зручна властивість для встановлення ширини полів для всіх сторін елемента. У цієї властивості може бути від одного до чотирьох значень. Якщо є тільки одне значення, то воно буде присвоєно відразу до всіх полів. Якщо два значення, перше з них присвоюється верхньому і нижньому полю, а друге - лівому і правому. Якщо ж три, то перше значення присвоюється верхньому полю, друге - лівому і правому, а третє - нижньому. Властивість **`margin`** заміняє властивості `margin-top`, `margin-right`, `margin-bottom`, `margin-left`.

Приклад: `p {margin: 20px 10px 20px;}`

`padding-top`, `padding-right`, `padding-bottom`, `padding-left`. Встановлює ширину проміжку між вмістом елемента і певною його межею (відповідно, верхньою, правою, нижньою і лівою). Значення: будь-яка стандартна довжина - число, яке задає ширину проміжку (*за замовчуванням 0*); будь-яке відповідне стандарту процентне значення - відношення у відсотках ширини проміжку до ширини елемента (для лівого і правого поля) чи до його висоти (для верхнього і нижнього поля).

Приклад: `p {padding-top: 20px;}`

padding. Зручна властивість для встановлення ширини проміжку **відразу для всіх** сторін елемента**.** У цієї властивості може бути від одного до чотирьох значень. Якщо є тільки одне значення, то воно буде присвоєно відразу всім полям. Якщо два значення, перше з них присвоюється верхньому і нижньому полям, а друге - лівому і правому. Якщо ж три, то перше значення присвоюється верхньому полю, друге лівому і правому, а третє - нижньому. Поряд із властивістю `padding` існують властивості, які встановлюють відступ тільки для однієї границі: `padding-top, padding-right, padding-bottom, padding-left`.

Приклад: `h1 {padding: 5px 5px 3px;}`

Додавання тіні до елементів

За допомогою властивості **box-shadow** є можливість додавати тіні до елементів сторінки. Це робить дизайн сторінки більш "природним" (тобто імітує реальний світ, де об'єкти відкидають тіні). Тінь може бути **зовнішньою** і **внутрішньою**. Зовнішні тіні створюють ефект піднесеності елемента над іншим вмістом, а внутрішні - ефект втиснутості елемента.

Приклад.

```
<html>
<head>
<style type='text/css'>
#el1,#el2,#el3 {
margin:20px;
padding:10px;
border:1px solid;
width:400px;
height:130px;
}
#el1 {box-shadow:4px 4px black;}
#el2 {box-shadow:6px 6px 6px 2px black;}
#el3 {box-shadow:0px 0px 6px 2px black
inset;}
</style>
</head>
```

```
<body>
  <div id="el1">1. Перші два значення
властивості <i>box-shadow </i>задають величину
зміщення тіні по горизонталі і вертикалі в
пікселях, а третє значення задає колір
тіні.<br /><br />
  <span style="text-
decoration:underline;">box-shadow:4px 4px
black; </span>
  </div>
  <div id='el2'>2. Також ця властивість може
мати значення, що
  вказують радіус поширення тіні (третє
значення) і розмір тіні (четверте значення).
<br /><br />
  <span style="text-
decoration:underline;">box-shadow:6px 6px 6px
2px black;</span>
  </div>
  <div id='el3'>3. З допомогою значення inset
можна вказати, що тінь повинна бути не
зовнішньою, а внутрішньою.
  Елементи з внутрішніми тінями здаються
'втиснутими'. <br /><br />
  <span style="text-
decoration:underline;">box-shadow:0px 0px 6px
2px black inset;</span>
  </div>
  </body>
  </html>
```

Границі

Створення елементів із згладженими кутами

CSS3 надає кілька нових властивостей для оформлення границь елементів. Зокрема, за допомогою нової CSS3-властивості **border-radius** можна робити кути елементів заокругленими.

Приклад.

```
<html>
<head>
<style type='text/css'>
#el1,#el2,#el3,#el4 {
float:left;
border:2px #000000 solid;
padding:10px;
width:300px;
height:100px;
margin:20px;
background-color:#85004B;
color:#FFF;
font-weight:bold;
}
#el1 {border-radius:5px;}
#el2 {border-radius:10px;}
#el3 {border-radius:20px;}
#el4 {border-radius:15px;}
</style>
</head>
<body>
<p id='el1'>Я - перший елемент з
заокругленими кутами
<span style='text-
decoration:underline'>border-
radius:5px</span></p>
<p id='el2'> Я - другий елемент з
заокругленими кутами
<span style='text-
decoration:underline'>border-
radius:10px</span></p>
<p id='el3'> Я - третій елемент з
заокругленими кутами
<span style='text-
decoration:underline'>border-
radius:20px</span></p>
```

```
<p id='el4'> Я - четвертий елемент з
заокругленими кутами
  <span style='text-
decoration:underline'>border-
radius:15px</span></p>
  </body>
  </html>
```

Наступна властивість може застосовуватися не до всіх кутів елемента, а тільки до певних:

- `border-top-left-radius` робить згладженим тільки верхній лівий кут елемента;
- `border-top-right-radius` робить згладженим тільки верхній правий кут елемента;
- `border-bottom-left-radius` робить згладженим тільки нижній лівий кут елемента;
- `border-bottom-right-radius` робить згладженим тільки нижній правий кут елемента.

Приклад.

```
<html>
<head>
<style type="text/css">
#el1,#el2,#el3,#el4 {
float:left;
border:2px #000000 solid;
padding:10px;
width:300px;
height:100px;
margin:20px;
background-color:#85004B;
color:#FFF;
font-weight:bold;
}
#el1 {border-top-left-radius:20px;}
#el2 {border-top-right-radius:20px;}
#el3 {border-bottom-left-radius:20px;}
#el4 {border-bottom-right-radius:20px;}
```

```
</style>
</head>
<body>
<p id="el1">Мій верхній лівий кут
заокруглений.</p>
<p id="el2">Мій верхній правый кут
заокруглений.</p>
<p id="el3">Мій нижній лівий кут
заокруглений.</p>
<p id="el4">Мій нижній правий кут
заокруглений.</p>
</body>
</html>
```

Встановлення кольору границі

За допомогою нової CSS3- властивості **border-color** можна регулювати колір кожного пікселя границі.

Приклад.

```
<html>
<head>
<style type='text/css'>
#el1,#el2
{
margin:20px;
padding:16px;
width:380px;
height:170px;
}
#el1
{
border:8px solid;
-moz-border-top-color: #FF0000 #EB1010
#D22E2E #B03E3E;
-moz-border-right-color: #FF0000 #EB1010
#D22E2E #B03E3E;
-moz-border-bottom-color:#FF0000 #EB1010
#D22E2E #B03E3E;
```

```
-moz-border-left-color: #FF0000 #EB1010
#D22E2E #B03E3E;
  border-radius:10px;
  }
  </style>
  </head>
  <body>
  <div id="el1">З допомогою нової властивості
<b> border-color </b> можна контролювати колір
кожного пікселя границі. <br /><br />Якщо
товщина границі перевищує кількість пікселів,
для яких задано колір, то надлишкові пікселі
будуть заповнюватися кольором, який заданий
останнім.</div>
  <p><b>Зверніть увагу:</b> ця властивість
буде працювати тільки в браузері Mozilla
Firefox.</p>
  </body>
  </html>
```


Вставлення зображень у вигляді границі

У CSS3 була додана нова властивість **border-image**, яка дозволяє вставляти довільні зображення як межі елемента. *Синтаксис*:

`border-image`: шлях(1) відступ(2) ширина(3) повторення(4);

Для того, щоб зробити це, необхідно:

1. Вказати шлях до зображення-границі;
2. Вказати величину відступу від кожного краю зображення для того, щоб розрізати його на 8 частин (верхній лівий кут, верхній край, верхній правий кут, лівий край і т. д.);
3. Вказати ширину границі в пікселях;
4. Вказати, чи повинно зображення повторюватися (`repeat`), заокруглюватися (`round`) або розтягуватися (`stretch`), щоб заповнити границю елемента.

Приклад.

```
<html>
```

```
  <head>
  <style type="text/css">
  div{
  border:10px solid;
  width:350px;
  height:70px;
  padding:10px 20px;
  }
  #el1
  {
  border-image:url("imgborder.jpg") 30 30
round;
  -o-border-image:url("imgborder.jpg") 30 30
round;
  }
  #el2
  {
  border-image:url("imgborder.jpg") 30 30
stretch;
  -o-border-image:url("imgborder.jpg") 30 30
stretch;
  }
  </style>
  </head>
  <body>
  <p>Зображення, що використовується як
границя в наступних прикладах :</p>
  <img src="imgborder.jpg"
style="width:60px;"/>
  <hr />
  <div id="el1">В цьому  прикладі зображення
буде повторюватися (repeat), щоб заповнити
границю елемента.</div>
  <div id="el2"> В цьому  прикладі зображення
буде розтягуватися (stretch), щоб заповнити
границю елемента.</div>
  <br />
```

```
</body>
</html>
```

Розглянемо деякі інші властивості границь.

border-top-width, border-right-width, border-bottom-width, border-left-width. Кожна з цих властивостей встановлює ширину рамки для певної сторони (відповідно, верхньої, правої, нижньої чи лівої). Рамка проходить між полем і основною частиною елемента.

Значення:

`thin` - рамка у вигляді тонкої лінії;

`medium` - рамка у вигляді лінії середньої товщини (за замовчуванням);

`thick` - рамка у вигляді товстої лінії.

Приклад: `p {border-top-width: 5px;}`

border-width. Зручна властивість для встановлення ширини рамки відразу для всіх сторін елемента. У цієї властивості може бути від одного до чотирьох значень. Якщо є тільки одне значення, то воно буде присвоєно відразу всім сторонам рамки. Якщо два значення, перше з них присвоюється верхній і нижній стороні, а друге - лівій і правій. Якщо ж три, то перше значення присвоюється верхній стороні, друге - лівій та правій, а третє - нижній стороні рамки. Властивість **border-width** універсальна. Вона заміняє властивості `border-top-width, border-right-width, border-bottom-width, border-left-width`, які задають товщину конкретної границі тега.

Приклад: `p {border-width: 2px 2px 1px 1px;}`

border-top-color, border-right-color, border-bottom-color, border-left-color.

Встановлює колір рамки для певної сторони (відповідно верхньої, правої, нижньої чи лівої). Рамка проходить між полем і основною частиною елемента. Значення: будь-яке відповідне стандарту значення кольору, зокрема, за ім'ям кольору (*наприклад*, *red* задає червоний колір); з допомогою RGB значення (*наприклад*, *rgb(255,255,255)* задасть білий колір); з

допомогою шіснадцяткового значення (*наприклад*, *#00ff00* задає зелений колір).

Приклад: `p {border-top-color: red;}`

`border-color`. Властивість для встановлення кольору рамки відразу для всіх сторін елемента. У цієї властивості може бути від одного до чотирьох значень. Якщо є тільки одне значення, то воно буде застосоване відразу до всіх сторін рамки. Якщо два значення, перше з них стосується верхньої і нижньої сторони, а друге - лівої і правої. Якщо ж три, то перше значення стосується верхньої сторони, друге - лівої та правої, а третє - нижньої сторони рамки.

Приклад: `p {border-color: red blue red blue;}`

`border-top-style, border-right-style, border-bottom-style, border-left-style`.

Встановлює стиль рамки для певної сторони (відповідно верхньої, правої, нижньої чи лівої). Рамка проходить між полем і основною частиною елемента. Значення:

`none` - лінія відсутня;

`hidden` - лінія теж відсутня, але для таблиці це значення діє по іншому;

`dotted` - пунктирна лінія;

`dashed` – штрихпунктирна лінія;

`solid` - суцільна неперервна лінія;

`double` - подвійна суцільна лінія;

`groove` - тривимірна борозна;

`ridge` - тривимірний гребінь;

`inset` - тривимірна вирізка;

`outset` - тривимірний орнамент;

Приклад: `p {border-top-style: double;}`

Рис 2.2. Значення і відповідний вигляд рамок

`border-style`. Властивість для встановлення стилю рамки відразу для всіх сторін елемента. У цієї властивості може бути від одного до чотирьох значень. Якщо є тільки одне значення, то воно буде стосуватися відразу всіх сторін рамки. Якщо два значення, перше з них стосується верхньої і нижньої сторони, а друге - лівої і правої. Якщо ж три, то перше значення стосується верхньої сторони, друге - лівої та правої, а третє - нижньої сторони рамки.

Приклад: `p {border-style: double solid;}`

`border-top`, `border-right`, `border-bottom`, `border-left`. Зручна властивість для встановлення декількох властивостей лінії рамки для певної сторони (відповідно верхньої, правої, нижньої чи лівої). Кожна окрема властивість застосовується до зазначеної для нього сторони.

Покажемо такі визначення, *наприклад*, для верхньої сторони.

Значення:

`border-top-width` - ширина верхньої межі;

`border-top-style` - стиль верхньої межі;

`border-top-color` - колір верхньої межі.

Приклад: `p {border-top: 1px solid red;}`

`border`. Властивість для визначення рамки відразу з усіх боків елемента. Значення, які встановлюються однаковими для всіх сторін рамки:

`border-width` - товщина границь;

`border-style` - стиль границь;

`border-color` - колір границь.

Приклад: p {border: 1px solid red;}

Питання для самоконтролю

1. У чому відмінність мід блочними і рядковими
2. елементами?
3. Опишіть способи позиціонування елементів у CSS.
4. Призначення властивості `z-index`.
5. Для чого використовують `float` і `clear`?
6. Опишіть `content, padding, border` та `margin.`

2.9. Трансформування елементів

З допомогою CSS3-функцій трансформування можна переміщувати, повертати і розтягувати елементи.

Зауваження: властивості, розглянуті в цьому розділі, працюють у всіх сучасних браузерах (*IE9+, Safari, Chrome, Firefox, Opera*), але для деяких браузерів потрібно додавати спеціальні префікси. Для браузерів *Chrome* і *Safari* потрібно використовувати префікс `-webkit`, для браузера *IE версії 9* - префікс `-ms` (для *IE10* цей префікс не потрібний).

З допомогою CSS3-властивості **`transform`** можна трансформувати елементи. Значенням цієї властивості повинна вказуватись одна з функцій трансформування. На даний момент браузерами підтримуються тільки 2D трансформації, але в майбутньому будуть доступні і 3D трансформації. За допомогою функції **`translate(x,y)`** можна змістити елемент на вказану кількість пікселів по горизонталі і вертикалі. *Приклад*.

```
<html>
<head>
<style type='text/css'>
#el1,#el2 {
position:absolute;
top:10px;
left:10px;
background-color:#7A005C;
color:white;
width:200px;
height:150px;
font-size:1.5em;
border:1px #000 solid;
}
#el2 {
transform: translate(180px,180px);
-webkit-transform: translate(180px,180px);
/* для Chrome и Safari */
-ms-transform: translate(180px,180px); /*
для IE */
```

```
}
</style>
</head>
<body>
<div id='el1'>Початкова позиція</div>
<div id='el2'>Позиція після застосування

translate(180px,180px)</div>
</body>
</html>
```

За допомогою функції `rotate(градуси)` можна повернути елемент на вказану кількість градусів за годинниковою стрілкою. *Приклад*.

```
<html>
<head>
<style type='text/css'>
#el3,#el4
{
margin:40px;
background-color:#7A005C;
width:130px;
height:110px;
font-size:1.5em;
border:1px #000 solid;
color:white;
padding:10px;
float:left;
}
#el3 {
transform: rotate(45deg);
-webkit-transform: rotate(45deg); /* для
Chrome і Safari */
-ms-transform: rotate(45deg); /* для IE */
}
#el4 {
transform: rotate(120deg);
```

```
  -webkit-transform: rotate(120deg); /* для
Chrome і Safari */
  -ms-transform: rotate(120deg); /* для IE */
  }
  </style>
  </head>
  <body>
  <div id='el3'>Елемент повернуто на 45
градусів</div>
  <div id='el4'>Елемент повернуто на 120
градусів</div>
  </body>
  </html>
```

Метод `scale(x,y)` дозволяє розтягнути елемент в ширину або висоту.

Приклад.

```
  <html>
  <head>
  <style type='text/css'>
  #el5,#el6
  {
  margin-left:45px;
  margin-bottom:10px;
  background-color:#7A005C;
  width:200px;
  height:110px;
  font-size:1.5em;
  color:white;
  padding:10px;
  }
  #el6 {
  transform:scale(1.3,1);
  -webkit-transform:scale(1.3,1);/*для Chrome
і Safari*/
  -ms-transform:scale(1.3,1); /* для IE */
  }
  </style>
```

```
</head>
<body>
<div id='el5'>Елемент до розтягування</div>
<div id='el6'>Елемент після
розтягування</div>
</body>
</html>
```

За допомогою методу `skew(x,y)` можна зкосити елемент на вказану кількість градусів по горизонталі і вертикалі. *Приклад*.

```
<html>
<head>
<style type='text/css'>
#el7
{
margin-left:70px;
margin-top:50px;
background-color:#7A005C;
width:200px;
height:110px;
font-size:1.5em;a
color:white;
padding:10px;
transform:skew(40deg,20deg);
-webkit-transform:skew(40deg,20deg); /* для
Chrome і Safari */
-ms-transform:skew(40deg,20deg); /* для IE
*/
box-shadow:3px 3px 3px #000;
}
</style>
</head>
<body>
<div id='el7'>Елемент розвернуто з допомогою
методу skew.</div>
</body>
</html>
```

Таблця 2.2.

CSS3-функції трансформування

Функція	Опис
`translate(x,y)`	Зміщує елемент з початкової позиції по горизонталі і вертикалі
`translateX(x)`	Зміщує елемент по горизонталі
`translateY(y)`	Зміщує елемент по вертикалі
`scale(x,y)`	Розтягує елемент по вертикалі і горизонталі
`scaleX(x)`	Розтягує елемент по горизонталі
`scaleY(y)`	Розтягує елемент по вертикалі
`rotate(градусы)`	Повертає елемент за годинниковою стрілкою
`skew(x,y)`	Зкошує елемент по горизонталі і вертикалі
`skewX(x)`	Зкошує елемент по горизонталі
`skewY(y)`	Зкошує елемент по вертикалі
`matrix(x,x,x,x,x,x)`	Суміщає всі наведені вище методи в один

Питання для самоконтролю

1. Яке призначення CSS3-властивості **`transform`**?
2. Опишіть CSS3-функції трансформування
3. Поясніть призначення CSS3-функцій `translate`, `scale`, `matrix`.

2.10. Прозорість. Градієнти. Переходи

Створення прозорих елементів і малюнків

З допомогою CSS розробник може створювати прозорі елементи та малюнки. Для створення прозорих елементів у всіх браузерах, крім *Internet Explorer*, використовується властивість **`opacity :`** `x`, де `x` значення, яке може змінюватися від *0.0* (повністю прозорий елемент) до *1.0* (повністю непрозорий елемент).

Для створення прозорих елементів в *Internet Explorer* використовується властивість `filter : alpha(opacity = x)`, де `x` значення, яке може змінюватися від *0* (повністю прозорий елемент) до *100* (повністю непрозорий елемент).

Приклад.

```
<html>
<head>
<style type='text/css'>
op1
{
position:absolute;
top:231px;
background-color:black;
width:401px;
height:62px;
opacity:0.7;
filter:alpha(opacity=70);
}
op2
{
position:absolute;
top:243px;
left:19px;
width:380px;
color:white;
}
</style>
</head>
<body>
<img style='border:solid black 1px;'
src='mountimg1.jpg' alt='Малюнок озера'/>
<p class='op1'> </p>
<p class='op2'>Золота  долина (висота
вершини ~1700 метрів).</p>
</body>
</html>
```

Відображення малюнків з різними рівнями прозорості

Приклад.

```
<html>
<head>
<style type='text/css'>
op1
{
  opacity:0.8;
  filter:alpha(opacity=80);
}
op2
{
  opacity:0.2;
  filter:alpha(opacity=20);
}
op3
{
  opacity:0.5;
  filter:alpha(opacity=50);
}
</style>
</head>
<body>
<p>Малюнок  із заданою властивістю
opacity:0.8 (filter:alpha(opacity=80)).</p>
<img class='op1' src='mountimg1.jpg' />
<p> Малюнок  із заданою властивістю
opacity:0.2  (filter:alpha(opacity=20)).</p>
<img class='op2' src='mountimg1.jpg' />
<p> Малюнок  із заданою властивістю
opacity:0.5 (filter:alpha(opacity=50)).</p>
<img class='op3' src='mountimg1.jpg' />
</body>
</html>
```

Лінійні градієнти

CSS3 має вбудовані властивості для створення лінійних, сферичних і повторюваних градієнтів.

Зауваження: градієнти підтримуються у всіх сучасних браузерах, але вимагають додавання спеціального префікса. Для браузера *IE10+* слід додати префікс `-ms`, для *Chrome і Safari* - префікс `-webkit`, для *Opera* - префікс `-o` і для *Firefox* – префікс `-moz`.

В CSS2.1 градієнти реалізовувалися у вигляді окремих малюнків, що вставляються як фонові. В CSS3 є вбудовані властивості для створення градієнтів. Оскільки в CSS3 сам браузер розмальовує градієнти, нема потреби робити додаткові запити до сервера щодо градієнтних зображень, а це дозволяє збільшити швидкість завантаження сторінок.

Лінійні градієнти створюються за допомогою CSS3-методу **`linear-gradient`**, який повинен вказуватися як значення властивості `background`. Для того, щоб створити лінійний градієнт, необхідно вказати його напрямок (може задаватися за допомогою ключових слів або градусів) і кольори переходу.

Приклад.

```
<html>
<head>
<style type='text/css'>
#wrap1,#wrap2,#wrap3,#wrap4 {
float:left;
margin:5px;
width:300px;
height:150px;
padding:10px;
border:1px #000 solid;
text-decoration:underline;
}
#wrap1 {
background:linear-gradient(top,white,black);
background:-webkit-linear-
gradient(top,white,black); /* для Safari */
}
```

```
  #wrap2 {
  background:linear-
gradient(left,white,black);
  background:-webkit-linear-
gradient(left,white,black); /* для Safari */
  }
  #wrap3 {
  background:linear-
gradient(0deg,white,black);
  background:-webkit-linear-
gradient(0deg,white,black); /* для Safari */
  }
  #wrap4 {
  background:linear-
gradient(270deg,white,black);
  background:-webkit-linear-
gradient(270deg,white,black); /* для Safari */
  }
  </style>
  </head>
  <body>
  <p>1. Приклади задання напрямків градиентів
з допом. ключових слів: </p>
  <div id="wrap1">linear-
gradient(top,white,black)</div>
  <div id="wrap2">linear-
gradient(left,white,black)</div>
  <br style="clear:both;" />
  <p>2. Приклади задання напрямків градиентів
з допомогою  градусів: </p>
  <div id="wrap3">linear-
gradient(0deg,white,black)</div>
  <div id="wrap4">linear-
gradient(270deg,white,black)</div>
  </body>
  </html>
```

Кольори переходу - це кольори, яких набуває градієнт у певних точках. *Наприклад*, градієнт, який плавно змінює колір з білого на чорний, має білий колір переходу в початковій точці і чорний в кінцевій. Лінійні градієнти можуть мати необмежену кількість кольорів переходу. Координати розміщення кольорів можна вказувати з допомогою % (0% означає початок градієнта, 100% - його кінець).

Приклад.

```
<html>
<head>
<style type="text/css">
#wrap1,#wrap2 {
margin:10px;
height:200px;
width:600px;
border:1px #000 solid;
float:left;
}
#text1,#text2 {
background-color:rgba(0,0,0,0.7);
color:white;
padding:10px;
font-weight:bold;
}
#wrap1 {
background:linear-gradient(left,white
0%,green 50%,black 100%);
background:-webkit-linear-
gradient(left,white 0%,green 50%,black 100%);
/* для Safari */
}
#wrap2 {
background:linear-gradient(left,#8F04A8
0%,#7CE700 60%,#FFE100 100%);
background:-webkit-linear-
gradient(left,#8F04A8 0%,#7CE700 60%,#FFE100
100%); /* для Safari */
```

```
}
</style>
</head>
<body>
<div id="wrap1">
<div id="text1">linear-gradient(top,white
0%, green 50%,black 100%)</div>
</div>
<div id="wrap2">
<div id="text2">linear-gradient(left,#8F04A8
0%,#7CE700 60%,#FFE100 100%)</div>
</div>
</body>
</html>
```

Сферичні градієнти

За допомогою методу **radial-gradient** можна створювати сферичні градієнти. Синтаксис визначення сферичних градієнтів схожий на синтаксис лінійних, але вимагає задання форми градієнта (сферичної або еліпсоїдної).

Приклад.

```
<html>
<head>
<style type="text/css">
#wrap1,#wrap2 {
margin:10px;
height:300px;
width:350px;
border:1px #000 solid;
float:left;
}
#text1,#text2 {
background-color:rgba(0,0,0,0.7);
color:white;
padding:10px;
font-weight:bold;
```

```
  border-bottom:1px white solid;
  }
  #wrap1 {
  background:radial-gradient(white 20%,black
40%);
  background:-webkit-radial-gradient(white
20%,black 40%); /* для Safari */
  }
  #wrap2 {
  background:radial-gradient(circle,#8F04A8
0%,#5D016D 40%,black 60%);
  background:-webkit-radial-
gradient(circle,#8F04A8 20%,#5D016D 30%,black
45%); /* для Safari */
  }
  </style>
  </head>
  <body>
  <div id="wrap1">
  <div id="text1">radial-gradient(white
20%,black 40%)</div>
  </div>
  <div id="wrap2">
  <div id="text2">radial-
gradient(circle,#8F04A8 0%, #5D016D 40%,black
60%)</div>
  </div>
  </body>
  </html>
```


Повторювані градієнти

Повторювані градієнти задаються з допомогою CSS3-методів `repeating-linear-gradient` (створює повторюваний лінійний градієнт) і `repeating-radial-gradient` (повторюваний сферичний градієнт). Для того, щоб створити повторюваний градієнт, слід вказати напрямок

градієнта, а також кольори переходу і відстань, яку вони повинні займати.

Приклад.

```
<html>
<head>
<style type="text/css">
#wrap1,#wrap2 {
margin:10px;
height:200px;
width:700px;
border:1px #000 solid;
float:left;
}
#text1,#text2 {
background-color:rgba(0,0,0,1);
color:white;
padding:10px;
font-weight:bold;
}
#wrap1 {
background:repeating-linear-
gradient(50deg,white,white 5px,black 5px,black
10px);
background:-webkit-repeating-linear-
gradient(60deg,white,white 5px,black 5px,black
10px); /* для Safari */
}
#wrap2 {
background:repeating-radial-
gradient(circle,#8F04A8 0%,#5D016D 40%,black
60%);
background:-webkit-repeating-radial-
gradient(circle,#8F04A8 20%,#5D016D 30%,black
45%); /* для Safari */
}
</style>
</head>
```

```
<body>
  <div id="wrap1">
  <div id="text1">repeating-linear-
gradient(top left,white,white 5px,black
5px,black 10px)</div>
  </div>
  <div id="wrap2">
  <div id="text2">repeating-radial-
gradient(circle,#8F04A8 0%,#5D016D 40%,black
60%)</div>
  </div>
  </body>
  </html>
```

Переходи CSS3

У CSS3 можна створювати ефекти динамічного переходу.

Зауваження: ця властивість підтримується у браузерах *IE 10+, Chrome, Firefox і Opera*. Для браузера *Safari* потрібно додати префікс -webkit.

За допомогою нової CSS3-властивості **transition** є можливість створювати ефекти переходу. Для створення переходів необхідно вказати, яка CSS-властивість буде змінюватися і швидкість виконання цих змін в секундах.

Приклад.

```
<html>
<style type='text/css'>
#wrap1
{
border:1px #000 solid;
width:200px;
padding:10px;
font-size:1.5em;
transition: width 4s;
-webkit-transition: width 4s; /* Safari*/
}
#wrap1:hover
{
```

```
width:500px;
}
</style>
<body>
<div id="wrap1">Наведіть на мене курсор
миші.</div>
<p><b>Зверніть увагу:</b> після того, як
курсор вийде за межі елемента, його ширина
повернеться до початкової.</p>
</body>
</html>
```

Для того, щоб додати ефект переходу до кількох властивостей, необхідно вказати їх назви через кому. *Приклад*.

```
<html>
<style type='text/css'>
#wrap1
{
border:1px #000 solid;
background-color:#E869AA;
color:#000;
width:200px;
padding:10px;
font-size:1.5em;
transition: color 4s, width 4s, background-
color 4s;
-webkit-transition: color 4s, width 4s,
background-color 4s; /* Safari*/
}
#wrap1:hover
{
color:#FFFFFF;
width:400px;
background-color:#880045;
}
</style>
<body>
```

```
<div id="wrap1"> Наведіть на мене курсор
миші .</div>
  <p><b> Зверніть увагу:</b> після того, як
курсор вийде за межі елемента, його ширина
повернеться до початкової.</p>
  </body>
  </html>
```

Функції пом'якшення

Плавність виконання переходів контролюється за допомогою функцій пом'якшення. У CSS3 існують кілька видів таких функцій:

- `linear`
- `ease` (функція пом'якшення за замовчуванням)
- `ease-in`
- `ease-out`
- `ease-in-out`
- `cubic-bezier(x,x,x,x)` (поведінка функції контролюється переданими параметрами)

Приклад.

```
<html>
<head>
<style type="text/css">
div {
margin:3px;
padding:10px;
font-size:1.5em;
border:1px #000 solid;
width:230px;
transition:width 4s;
-moz-transition:width 4s;
-webkit-transition:width 4s;
-o-transition:width 4s;
}
div:hover {
width:600px;
```

```
  }
  #el1 {
  transition-timing-function:linear;
  -webkit-transition-timing-function:linear;
  }
  #el2 {
  transition-timing-function:ease;
  -webkit-transition-timing-function:ease;
  }
  #el3 {
  transition-timing-function:ease-in;
  -webkit-transition-timing-function:ease-in;
  }
  #el4 {
  transition-timing-function:ease-out;
  -webkit-transition-timing-function:ease-out;
  }
  #el5 {
  transition-timing-function:ease-in-out;
  -webkit-transition-timing-function:ease-in-
out;
  }
  #el6 {
  transition-timing-function:cubic-
bezier(0.6,0.2,0.5,0.6);
  -webkit-transition-timing-function:cubic-
bezier(0.1,0.2,0.5,0.6);
  }
  </style>
  </head>
  <body>
  <p>1. Наведіть курсор на елементи, розміщені
нижче, щоб побачити відповідні функції
пом’якшення в дії.</p>
  <div id="el1">linear</div>
  <div id="el2">ease</div>
  <div id="el3">ease-in</div>
```

```
  <div id="el4">ease-out</div>
  <div id="el5">ease-in-out</div>
  <p>2. Поведінка функції cubic-bezier
залежить від переданих в неї значень
(спробуйте ввести декілька своїх значень від 0
до 1, щоб краще зрозуміти принцип її
роботи).</p>
  <div id="el6">cubic-bezier(x,x,x,x)</div>
  </body>
  </html>
```

Таблця 2.3.

CSS3-властивості переходів

Властивість	**Опис**
`transition`	Дозволяє задати значення чотирьох різних властивостей переходу в одному визначенні
`transition-property`	Дозволяє вказати ім'я CSS-властивості, до якої буде застосований ефект переходу
`transition-duration`	Дозволяє вказати час виконання переходу (*за замовчуванням має значення 0*)
`transition-timing-function`	Дозволяє задати функцію пом'якшення, яка відповідає за плавність виконання переходу (*за замовчуванням має значення 'ease'*)
`transition-delay`	Дозволяє задати затримку початку виконання переходу (*за замовчуванням має значення 0*)

Питання для самоконтролю

1. Створення прозорих елементів у CSS3.
2. Які властивості вбудовані в CSS3 для створення градієнтів?
3. У чому полягають ефекти динамічного переходу?

2.11. Властивості, що впливають на друк

orphans. Визначає мінімальну кількість рядків, яка може залишатися на попередній сторінці при друкуванні документа.

Синтаксис: `orphans : число`.

widows. Визначає мінімальну кількість рядків, яка може залишатись на наступній сторінці при друкуванні документа.

Синтаксис: `widows : число`.

page-break-inside. Дозволяє заборонити або дозволити розрив сторінки всередині елемента при друку.

Значення за замовчуванням: `auto`.

Синтаксис: `page-break-inside : auto | avoid | inherit.`

page-break-before. Задає розрив сторінки перед вказаним елементом при друку.

Значення: `always` - завжди вставляти розрив сторінки; `auto` — вставляти розрив сторінки за необхідності; `avoid` — заборонити вставляння розрив сторінки; `left` — пропустити одну або дві сторінки, щоб наступна сторінка при друку була парною; `right` — пропустити одну або дві сторінки, щоб наступна сторінка при друку була непарною.

Приклад: `page-break-before : always`

page-break-after. Задає розрив сторінки після заданого елемента при друку. Значення аналогічне властивості `page-break-before`.

Питання для самоконтролю

1. Які властивості CSS впливають на друк?
2. Опишіть призначення `orphans` і `widows.`
3. Яке значення забороняє вставляння розривів сторінки при друку?
4. Поясніть призначення `page-break-inside: avoid`.

2.12. Анімація

З допомогою CSS3 можна створювати на сторінках повноцінну анімацію без використання *Adobe Flash* і *JavaScript.*

Зауваження: властивості анімації підтримуються браузерами *IE10+, Firefox і Opera*. Для браузерів *Chrome і Safari* перед властивістю потрібно додати префікс `-webkit`. Для створення анімації в CSS3 використовується правило **`@keyframes`**. Це правило являє собою контейнер, в якому повинні міститися різні властивості оформлення.

Синтаксис:

```
@keyframes ім'яАнімації
{
from {CSS властивості} /* Оформлення
елемента перед початком анімації */
to {CSS властивості} /* Оформлення елемента
після завершення анімації */
}
```

Після того, як анімація була створена, необхідно додати до елемента, який потрібно анімувати, CSS3-властивість **`animation`** і вказати у ній *ім'я анімації (перше значення)* і *час, протягом якого вона буде виконуватися (друге значення)*.Тут також можна встановлювати *кількість повторень анімації (третє значення)*.

Приклад.

```
<html>
<style type='text/css'>
@keyframes anim{
from {margin-left:3px;}
to {margin-left:500px;}
}
@-moz-keyframes anim{
from {margin-left:3px;}
to {margin-left:500px;}
}
@-webkit-keyframes anim{
from {margin-left:3px;}
to {margin-left:500px;}
}
#wrap1{
border:2px #000 solid;
```

```
background-color:#7F0055;
height:100px;
width:100px;
font-size:2em;
animation:anim 4s 3;
-webkit-animation:anim 4s 3;
}
</style>
<body>
<div id="wrap1"></div>
<p><b>Зверніть увагу:</b> ця анімація буде
повторюватися 3 рази.</p>
</body>
</html>
```

Хід виконання анімації

Хід виконання анімації може визначатися не тільки з допомогою ключових слів **from** і **to** (які використовувалися в попередньому прикладі), але і за допомогою **%.** Використання % дозволяє точніше контролювати хід виконання анімації.

Наприклад, можна вказати, що певний елемент на початку анімації (0%) повинен бути білим, до середини (50%) має забарвлюватися в помаранчевий колір, а до кінця (100%) ставати чорним.

Приклад.

```
<html>
<style type='text/css'>
@keyframes anim {
0% {margin-left:3px;margin-
top:3px;background-color:#7F0055;}
30% {margin-left:3px;margin-
top:250px;background-color:#7F0055;}
60% {margin-left:500px;margin-
top:250px;background-color:black;}
100% {margin-left:3px;margin-
top:3px;background-color:#7F0055;}
}
```

```
@-moz-keyframes anim {
  0% {margin-left:3px;margin-
top:3px;background-color:#7F0055;}
  30% {margin-left:3px;margin-
top:250px;background-color:#7F0055;}
  60% {margin-left:500px;margin-
top:250px;background-color:black;}
  100% {margin-left:3px;margin-
top:3px;background-color:#7F0055;}
  }
  @-webkit-keyframes anim {
  0% {margin-left:3px;margin-
top:3px;background-color:#7F0055;}
  30% {margin-left:3px;margin-
top:250px;background-color:#7F0055;}
  60% {margin-left:500px;margin-
top:250px;background-color:black;}
  100% {margin-left:3px;margin-
top:3px;background-color:#7F0055;}
  }
  #wrap1 {
  border:2px #000 solid;
  background-color:#7F0055;
  height:100px;
  width:100px;
  font-size:2em;
  animation:anim 6s 3;
  -webkit-animation:anim 6s 3;
  }
  </style>
  <body>
  <div id="wrap1"></div>
  </body>
  </html>
```

Таблця 2.4.

Властивості анімації

Властивість	**Опис**
`@keyframes`	Контейнер для визначення анімації
`animation`	Дозволяє задати всі значення для налаштування виконання анімації в одному визначенні
`animation-name`	Дозволяє вказати ім'я анімації
`animation-duration`	Дозволяє задати швидкість виконання анімації в секундах (*за замовчуванням значення 0*)
`animation-timing-function`	Дозволяє задати функцію пом'якшення, яка створює плавність виконання анімації (*за замовчуванням значення 'ease'*)
`animation-delay`	Вказує затримку початку виконання анімації (*за замовчув. значення 0*)
`animation-iteration-count`	Дозволяє задати кількість повторень анімації (*за замовчув. значення 1*)
`animation-direction`	При непарних значеннях `alternate` (*1,3,5* ...) анімація буде виконуватися у звичайному, а при парних (*2,4,6* ...) - в оберненому порядку. За замовчуванням ця властивість має значення `normal`, при якому анімація виконується в звичайному порядку

Питання для самоконтролю

1. Яка властивість створює анімацію?
2. Яка CSS3-властивість визначає елемент, який потрібно анімувати?
3. Скільки параметрів вказують при визначенні елемента анімації?
4. Які ключові слова визначають хід виконання анімації?

3. Використання JavaScript

3.1. Основи мови JavaScript

Загальний огляд мови JavaScript.

JavaScript – це мова програмування, що використовується в складі HTML-сторінок для збільшення їх функціональності та можливостей взаємодії з користувачем. JavaScript є однією із найпопулярніших мов у світі. Ця мова програмування була створена фірмами *Netscape* та *Sun Microsystems* на базі мови програмування *Sun's Java*. На сьогодні JavaScript суттєво розширила можливості web-технологій.

Код JavaScript розміщується або безпосередньо в HTML-документі, або в окремому файлі, що пов'язаний за допомогою спеціальних команд з HTML-сторінкою. Цей код, як правило, розміщується всередині тега HTML та завантажується в браузер разом з кодом HTML-сторінки.

Виконання програми JavaScript відбувається при перегляді HTML-сторінки в браузері, який містить інтерпретатор JavaScript. Відзначимо, що крім JavaScript на HTML-сторінках можна використовувати інші мови програмування. *Наприклад*, *VBScript* або *JScript*, яка є варіантом *JavaScript* від фірми *Microsoft*. Але виконання програм VBScript та JScript гарантовано коректне тільки при перегляді HTML-сторінки за допомогою браузера Microsoft Internet Explorer. Тому в більшості випадків використання JavaScript доцільніше, хоча функціональність програм VBScript та JScript дещо краща.

Часто програму JavaScript називають **скриптом** або **сценарієм**. Скрипти виконуються в результаті того, що відбулась певна подія, пов'язана з HTML-сторінкою, здебільшого, з ініціативи користувача.

Скрипт може бути пов'язаний з HTML-сторінкою за допомогою парного тега `<script>` або як оброблювач події, що стосується конкретного тега HTML.

Сценарій, вбудований в HTML-документ з використанням тега `<script>`, має такий формат:

```
<script>
// Код програми
```

```
</script>
```

Тег <script> може розміщуватися у будь-якому місці сторінки, зокрема, при наявності великої кількості скриптів на web-сторінці їх доцільно розміщувати в нижній частині тега <body>.

Все, що розміщується між тегами <script> та </script>, інтерпретується як код програми мовою *JavaScript*. Починаючи з HTML5, JS є офіційною скриптінговою мовою і атрибут <type> використовувати не потрібно. Обсяг вказаного коду не обмежений.

JavaScript-код можна також записувати в окремому файлі з розширенням js, після чого підключати його до HTML-документа приблизно так, як ми робили це з CSS-файлами. Файл з розширенням js є звичайним текстовим файлом з UTF-8-кодуванням, як і інші вже відомі нам CSS- і HTML-файли (js - файли зазвичай розміщують у папці js).

Для підключення js-файлів також використовується тег <script>. JS-код може розміщуватися як в тегу <head>, так і в тегу <body>. На відміну від підключення CSS-документа, тут слід використовувати інший синтаксис:

```
<script type= “text/javascript” src= “шлях
до js-файлу”>
</script>
```

Тег <script> має декілька необов’язкових параметрів. Найчастіше використовуються параметри **language** та **src**. Параметр **language** дозволяє визначити мову та версію мови сценарію. Параметр **src** дозволяє задати файл з кодом сценарію. Для пояснення використання параметрів тега <script> розглянемо задачу.

Задача. Необхідно для HTML-сторінки *hi.htm* створити сценарій мовою JavaScript для показу на екрані вікна повідомлення з текстом "Привіт!".

Зауважимо, що для показу на екрані вікна повідомлення можна використати функцію alert(). Для ілюстрації можливостей пов’язування скриптів з HTML-кодом покажемо два варіанти розв’язування задачі.

Варіант 1. Визначення сценарію безпосередньо на HTML-сторінці *hi.htm*.

```
<html><head>
<title>Використання JavaScript</title>
</head>
<body>
<script language="JavaScript">
alert('hi');
</script>
</body></html>
```

Варіант 2**.** Визначення сценарію у файлі *a.js*, пов'язаному з HTML-сторінкою *hi.htm* за допомогою параметра `src` тега `<script>`. Код HTML-сторінки *hi.htm*:

```
<html><head>
<title>Використання JavaScript</title>
<script language="JavaScript" src="a.js">
</script>
</head><body>
</body></html>
```

Програмний код, записаний у файлі *a.js*:

```
alert('hi');
```

Відмітимо, що мінімальним комплектом програмного забезпеченням для розробки та тестування програм JavaScript є текстовий редактор (*наприклад*, NotePad++, Brackets) та браузер з підтримкою JavaScript.

Синтаксис. Визначення та ініціалізація змінних

Сценарій JavaScript являє собою набір операторів (команд), що послідовно інтерпретуються браузером. Оператори можна розміщувати як в одному, так і в окремих рядках. В кінці оператора ставиться крапка з комою. Будь-який оператор можна розмістити в декількох рядках без символу продовження.

Будь-яка послідовність символів, розміщених в одному рядку після **//**, розглядається як **коментар**. Для визначення **багаторядкових коментарів** використовується конструкція:

```
/*
Багаторядковий коментар
*/
```

У мові JavaScript малі та великі букви вважаються різними символами. JavaScript використовує змінні для зберігання даних визначеного типу. При зміні типу даних автоматично змінюється і тип змінної.

Ім'я змінної повинно містити тільки букви латинського алфавіту, символ підкреслення _, арабські цифри та починатись з букви або символу підкреслення _.

Оголосити змінну можна оператором **var**, *наприклад*, `var ім'я`; Однак частіше змінну не тільки оголошують, але і задають їй значення:

```
var ім'я=значення;
```

Значення змінної можна змінити з допомогою оператора присвоєння.

Типи змінних

Від значення, яке зберігається в змінній, залежить тип змінної. У JavaScript використовуються такі типи змінних: `Number, String, Boolean, Date, Math, Array, Object, Error`. При оголошенні змінної можна вказати, до якого типу вона належить:

```
var ім'я = new тип();
```

Наприклад: `var myvar = new Date();`

// створити змінну типу `Date`.

Від типу змінної залежить не тільки формат її значення, але й властивості і методи, які вона буде підтримувати. Якщо змінна оголошується без оператора `new`, тип змінної визначається значенням, яке в ній розміщено.

Типи `Number, String, Boolean` називаються елементарними. При оголошенні змінних цих типів бажано не використовувати оператор `new`, а просто вказати значення:

```
var myvar = 34;
```

Інші типи змінних є об'єктами, хоча, в сучасному JavaScript всі типи змінних є об'єктами. Розглянемо типи змінних докладніше.

Number. Змінними цього типу є всі змінні, в яких зберігаються числові значення: цілі або десяткові. В JavaScript

результатом ділення на нуль буде не помилка і зупинка виконання сценарію, а значення `Number.POSITIVE_INFINITY` (+∞) або `Number.NEGATIVE_INFINITY` (-∞). Деякі помилки можуть також призводити до появи значення `NaN` (результат не є числом). При цьому виконання сценарію не буде зупинено.

String. Рядкові змінні в JavaScript містять у собі набір Unicode-символів (на один символ відводиться 2 байти). Крім звичайних символів, усередині рядка можуть використовуватися escape-послідовності. **Escape**-послідовністю називається набір символів, який починається із знака «\» і перетворюється в якийсь нестандартний символ або символ Unicode. Причому escape-послідовності, що кодують символи Unicode, починаються з «\u». *Приклади*:

1) `var str1 = "це просто перший рядок\n А це другий";`

2) `var str="привіт";`

`var len=str.length;/* довжина рядка, визначена властивістю length, =6 */`

3) `var str = "house";`

`var symb = str[2]; /* символ рядка symb дорівнює "u" */`

4) `var str = "house";`

`var symb = str.charAt(1);/*символ рядка symb дорівнює "o" */`

Boolean. Логічні змінні набувають одного з двох значень: `true` (істина) або `false` (хибність).

Date. Дата може містити не тільки рік, місяць і день, але і час. *Приклади:*

```
var date1 = new Date(); // поточна дата
var date2 = new Date(число);/*кількість мілісекунд з 01.01.1970 00:00*/
var date2 = new Date(рядок); /* рядок, що містить дату */
var date6 = new Date(число, число, число, число);
```

```
// рік, місяць, день, година
var date2 = new Date(2015, 4, 1); /* 1 квітня
2015 року */
```

Math. Змінні типу **Math** створювати не можна, тобто запис `var ім'я = new Math()` помилковий. В об'єкті класу **Math** зберігаються математичні функції, які можна використовувати для обчислень у сценаріях. Це робиться викликом виду `Math.метод`. При цьому ніяких змінних типу **Math** не створюється.

Приклади:

1) `Math.PI` — повертає значення числа «пі»;

2) `Math.E` - число Ейлера (основа натурального логарифма) ;

3)
```
var var1 = 1.5;
var var2 = Math.round(var1); /* var2=2 –
    заокруглення числа */
```

Array. З допомогою даних типу **`Array`** можна створювати масиви пронумерованих елементів. Детальніше ці типи даних будуть розглянуті нижче.

Object. Об'єкт без певного класу. Найчастіше об'єкти класу **`Object`** використовуються JavaScript для створення асоціативних масивів. Механізм спадкування на JS реалізований через об'єкти.

Error. Об'єкти класу **`Error`** створюються при різних помилках виконання сценарію. Об'єкт даного типу можна також оголосити наступним записом:

```
new Error (повідомлення про помилку);
```

Конвертування

У JavaScript значення одного типу можна перетворювати на значення іншого типу.

Конвертування в число.

`Number(рядок)` - конвертує рядок у число.

Приклади:

```
var tmp1 = 34+"56";          // tmp1=3456
var tmp2 = 34+ Number("56"); // tmp2=90
```

`parseFloat()` **–** інший спосіб перетворення рядка в число,

який допускає наявність у рядку нечислових символів. При їх виявленні частина рядка обрізається. *Приклади*:

```
var var2=parseFloat("34.43");  // var2=34.43
var var4=parseFloat("34.4sd3"); // var4=34.4
var var5=parseFloat("ds34.43"); // var5=NaN
```

parseInt() - конвертує рядок у ціле число в будь-якій системі числення. *Приклади*:

```
parseInt("010")=8     /*вісімкова     система
числення */
var var2=parseInt("34.43",10);  // var2=34
var var3=parseInt("3443sd",10); // var3=3443
```

Оператор + використовується як ще один спосіб неявного перетворення рядка в число. *Приклади*:

```
var tmp2=34+(+"56"); // tmp2=90
var tmp2=34+(-"56"); // tmp2=-22
```

Boolean(), !!. Для конвертування в тип Boolean записуємо **Boolean(**значення**)** або **!!**значення. Результатом конвертування буде **true** або **false**.

Методи

Крім властивостей, змінні можуть підтримувати різні методи. Список доступних методів залежить від типу змінної. **Методом** називається послідовність команд, в результаті виконання яких повертається певне значення. Як правило, метод стосується змінної, для якої він викликається. Синтаксис виклику методу наступний: змінна**.**метод**()**.

Наприклад:

```
var num = 12.3456;
var num2 = num.toFixed();   // num2=12
var num3 = num.toFixed(2); // num3=12.34
```

Вирази та оператори

Вираз – це комбінація змінних, літералів та операторів, в результаті обчислення якої можна отримати тільки одне значення, що може бути числовим, рядковим чи булівським. Для виконання обчислень у JavaScript використовують арифметичні, рядкові, логічні вирази та декілька типів операторів.

Результатом обчислення **арифметичного виразу** є число, *наприклад*, ***a***=7+5. **Рядкові вирази** оперують з рядками символів. Результатом обчислення **логічного виразу** є `true` (істина) або `false` (хибність). **Оператор присвоювання** (знак =) лівому операнду присвоює значення правого операнда чи виразу.

До стандартних **арифметичних операторів** належать: оператори додавання (**+**), віднімання (**-**), множення (*****), ділення (**/**), остача від ділення чисел (**%**), одиничний інкремент (**++**) - збільшення числової змінної на 1, одиничний декремент (**- -**) - зменшення числової змінної на 1. Відмітимо, що оператор додавання можна використовувати ще й для додавання (контрактації / конкатенації) символьних (текстових) рядків.

Для створення логічних виразів використовуються **логічні оператори** та **оператори порівняння**. До логічних операторів належать логічне ***І*** (***&&***), логічне ***АБО*** (||), логічне ***НІ*** (***!***).

Операторы порівняння не відрізняються від таких операторів в інших мовах програмування. До операторів порівняння належать (**==**, **<**, **<=**, **>**, **>=**, **!=**).

Оператори вибору

Оператори вибору належать до **операторів управління,** призначенням яких є зміна напрямку виконання програми. Крім операторів вибору до операторів управління належать оператори циклу та оператори маніпулювання об'єктами.

Оператори вибору призначені для виконання деяких блоків операторів в залежності від істинності певного логічного виразу. До операторів вибору належать: оператор умови `if...else` та перемикач `switch`.

Синтаксис **оператора умови**:

```
if (умова) {
група операторів 1
. . . . . .
}
[else {
група операторів 2
. . . . . .
}]
```

Перша група операторів виконується при умові істинності виразу (*умова*). Необов'язковий блок **else** визначає другу групу операторів, яка буде виконуватися у випадку хибності умови, заданої у блоці **if**. В групі операторів можуть бути використані будь-які інші оператори, в тому числі, і інші оператори умови. Це дозволяє створювати групу вкладених операторів умови **if**, якщо це передбачено алгоритмом. Однак, вкладені конструкції ускладнюють розуміння програми. Тому в таких випадках доцільно використовувати оператор **switch**. В цьому операторі обчислюється деякий вираз і його значення порівнюється із значенням, заданим в блоках **case**.

Синтаксис оператора **switch** такий:

```
switch (вираз) {
case значення1:
[оператори1]
break;
case значення2:
[оператори2]
break;
...
default:
[оператори]
}
```


Якщо значення виразу у блоці **switch** дорівнює значенню1, то виконується група операторів оператори1, якщо значення дорівнює значенню2, то виконується група операторів оператори2 і т. д. Якщо значення виразу не дорівнює жодному із значень, що задані в блоках **case**, то обчислюється група операторів блоку **default**, якщо цей блок заданий, інакше відбувається вихід із оператора switch. Необов'язковий оператор break, який можна задавати в кожному із блоків case, виконує безумовний вихід із оператора switch. Якщо він не заданий, то продовжується виконання операторів в наступних блоках case до першого оператора break або до кінця оператора switch.

Оператори циклу

Цикл – це деяка група команд, що повторюється до тих пір, поки виконується задана умова. JavaScript підтримує цикл у формі `for` та у формі `while`. Крім того, в циклах використовуються оператори `break` та `continue`.

For

Цикл `for` повторює групу команд до тих пір, поки вказана умова істинна. Синтаксис оператора `for` такий:

```
for (ініціалізація_змінної_циклу; умова;
     збільшення_або_зменшення_змінної_циклу)
{
      тіло_циклу
}
```

Виконання циклу `for` проходить в такій послідовності.

1. Вираз `ініціалізація_змінної_циклу` служить для присвоєння початкового значення змінній циклу. Цей вираз обчислюється один раз на початку виконання циклу.

2. Вираз `умова` обчислюється на кожній ітерації циклу. Якщо значення виразу `умова` дорівнює `true`, виконується група операторів, які утворюють тіло циклу. Якщо значення умови дорівнює `false`, то виконання циклу припиняється і починає виконуватися оператор, наступний за циклом `for`. Якщо вираз `умова` пропущено, то він вважається рівним `true`. В цьому випадку цикл продовжується до оператора `break`.

У JavaScript до логічного типу можуть бути зведені інші типи даних. Зокрема, будь-яке число трактується як `false`, якщо воно дорівнює 0, і `true` – в іншому випадку. Текстовий рядок розцінюється як `true`, якщо він непорожній, тобто має ненульову довжину, і `false` – в іншому випадку.

3. Вираз `збільшення_або_зменшення_змінної_циклу` використовується для зміни значення змінної циклу. Типовою зміною є збільшення або зменшення змінної циклу на 1 (наприклад, i++ або i– –).

Приклад 1. Цикл для обчислення суми цілих чисел від 1 до 100.

```
    s=0
for (i=1;i<101;i++) {
    s=s+i;
}
```

Приклад 2. Використання циклу для роботи з масивом.

```
var arr = ["червоний", "синій", "жовтий",
"зелений"]
for(var i=0; i<arr.length; i++){
   switch(arr[i]){
case "синій":
   window alert(”синій”);
   break;
default:
   window alert(”інший”);
}
}
```

В цьому прикладі цикл завершиться при досягненні кінця масиву (`i<arr.length`). При кожній ітерації циклу для обчислень береться черговий елемент масиву (`arr[i]`).

For in

Розглянутий у прикладі 2 цикл опрацювання елементів масиву добре підходить для випадку, коли індекси задані послідовно. А якщо деякі індекси пропущені?

Наприклад, є лише елементи з індексами 0, 13, 26: `var mass = ["червоний"]; mass[13] = "синій"; mass[26] = "жовтий";`

Використання циклу `for` тут буде неефективним. В таких випадках для роботи з масивами, об’єктами чи іншими послідовностями використовують цикл `for in`:

```
for(var змінна in масив) {
  команди
}
```

Цей цикл перебирає всі елементи масиву, розміщуючи кожен із них у <змінну>. При першому виконанні циклу в змінній буде розміщений перший елемент масиву (не обов’язково з індексом 0), при другому – другий і т. д.

Наприклад:

```
for(var item in mass) {
  window.alert(item);
}
```

В цьому варіанті циклу також можна використовувати ключові слова `break` і `continue`.

While

Крім циклу `for` існує ще один різновид циклу – цикл `while`. Оператор `while` повторює виконання набору команд (тіло циклу), поки вказана умова істинна. Оператор `while` має вигляд:

```
while (умова) {
  тіло_циклу
}
```

Цикл `while` виконується таким чином. Спочатку перевіряється `умова`. Умовою може бути довільний вираз логічного типу (який набуває значення `true` або `false`). Якщо умова істинна, то виконується група операторів `тіло_циклу`. Перевірка істинності виконується на кожному кроці циклу. Якщо `умова` хибна, то цикл припиняється.

Do while

Синтаксис циклу `do while`, який є різновидом `while` :

```
do {
тіло_циклу
} while (умова)
```

Інколи необхідно завершити цикл не за умовою, що задана в його заголовку, а в результаті виконання деякої умови в тілі циклу. Для цього використовуються оператори **`break`** та **`continue`**. Оператор `break` спричиняє негайне припинення виконання циклу та початок виконання оператора, записаного у коді безпосередньо після циклу.

Оператор `continue` передає управління оператору перевірки істинності умови в циклі `while` та оператору оновлення значення лічильника в циклі `for` і продовжує виконання циклу.

Питання для самоконтролю

1. Призначення мови JavaScript.
2. Які є способи використання скриптів у HTML-документі?
3. Коли виконуються скриипти JavaScript?
4. Де може розміщуватися скрипт на web-сторінці?
5. Яке програмне забезпечення використовується для розробки та тестування програм JavaScript?
6. Вирази. Синтаксис запису виразів у JavaScript.
7. Арифметичні оператори JavaScript.
8. Логічні оператори та оператори порівняння.
9. Оператори JavaScript.
10. Формат запису умовного оператора.
11. Формат запису оператора вибору.
12. Формат запису операторів циклу.
13. Як достроково завершити цикл?
14. Як достроково перейти на наступну ітерацію циклу?

3.2. Об’єктна модель документа

Об'єктна модель документа (DOM)

Незалежно від базової стратегії розробки клієнта, всі підходи базуються на об’єктній моделі документа DOM (Document Object Model) — специфікації прикладного програмного інтерфейсу для роботи із структурованими документами (HTML, XML). Специфікація такої моделі визначена консорціумом W3C (World Wide Web Consortium - об’єднання розробників технологій та організацій, відповідальних за стандарти HTTP, HTML, XML, XSL, DOM та інші важливі стандарти Web і Internet). Більшість розробників браузерів реалізували її в останніх версіях своїх продуктів. Основна ідея полягає у використанні спільного інтерфейсу API, який розробники web-сторінок можуть застосовувати для обробки вмісту документа HTML (або XML), а також ресурсів самого браузера.

При використанні об’єктної моделі документа програми та сценарії можуть динамічно отримувати доступ та оновлювати зміст документа, його структуру та стиль. Потім документ може

опрацьовуватися браузером, а результати цієї обробки можуть фіксуватися на відкритій сторінці. За такої архітектури браузер відповідає як за відображення сторінки HTML, яка може бути змінена після її отримання з сервера, так і за виконання сценаріїв та компільованих програм у документі.

Назва моделі DOM пов'язана з тим, що вона забезпечує об'єктний інтерфейс до документів HTML (та XML). Документи розглядаються як об'єкти, що мають дані та поведінку. Взаємодія цих об'єктів показує структуру документа.

JavaScript належить до об'єктно-орієнтованих мов програмування. **Об'єкт** – це цілісна конструкція, що має **властивості**, які є змінними JavaScript, та **методами** їх обробки. Властивості можуть бути іншими об'єктами. Функції, пов'язані з об'єктом, називаються **методами** об'єкта. Для звернення до властивостей об'єкта використовується наступний синтаксис:

```
objectName.propertyName
```

Ім'я об'єкта, імена властивостей та методів чутливі до регістру. Для визначення властивостей їм необхідно присвоїти значення. *Наприклад*, якщо існує об'єкт з ім'ям *myCar,* то для визначення властивості `model` необхідно записати:

```
myCar.model = "Таврія"
```

Для визначення методів необхідно спочатку задати звичайну функцію, а після цього зв'язати цю функцію з існуючим об'єктом:

object.methodname = function_name

де *object* – існуючий об'єкт; *methodname* – ім'я, що призначається методу; *function_name* – ім'я функції.

Виклик методу в контексті об'єкта реалізується так:

```
object.methodname (params);
```

Для створення примірника об'єкта необхідно:

- **написати функції**, які будуть використані як методи об'єкта;
- за допомогою звичайної функції **визначити об'єкт**;
- за допомогою оператора **new створити примірник** об'єкта.

Наприклад, необхідно створити об'єкт з ім'ям *car*, властивостями *model* і *color* та методом *go*. Для цього необхідно

написати функцію *when*, яка буде використана для визначення методу *go*:

```
function when()
{
  //код функції
}
```

Після цього треба написати функцію для визначення об'єкта:

```
function car( model, color)
{
  this.model = model;
  this.color = color;
  this.go =when;
}
```

Відмітимо, що оператор **this** використовується для того, щоб присвоїти значення властивостям об'єкта, базуючись на параметрах, які передаються функції.

Створення об'єкта з ім'ям *mycar* можна реалізувати так:

```
mycar = new car("Таврія", "Зелений")
```

В JavaScript всі елементи (теги) на HTML-сторінці розміщені в ієрархічній структурі. Причому кожен елемент поданий у вигляді об'єкта з визначеними властивостями та методами. Керування об'єктами на HTML-сторінці можна здійснити за рахунок того, що JavaScript дозволяє одержати доступ до цих властивостей та методів. При реалізації доступу необхідно враховувати ієрархію об'єктів на HTML-сторінці. Загальним об'єктом-контейнером є об'єкт `window`, який відповідає вікну браузера. В свою чергу цей об'єкт містить деякі елементи оформлення, наприклад, рядок стану. Завантаженій у вікно браузера HTML-сторінці відповідає об'єкт `document`. Всі без винятку елементи HTML-сторінки є властивостями об'єкта `document`. *Прикладами* об'єктів HTML є таблиця, гіперпосилання або форма. Для доступу до методів чи властивостей елементів на HTML-сторінці використовується наступне звертання:

```
document.ім'я_об'єкта.ім'я_методу()
```

Об'єкти JavaScript

Об'єктна модель документа, головне джерело об'єктів для JavaScript, забезпечує об'єктний інтерфейс не лише для документів HTML, але й для браузера. Сценарій JavaScript може взаємодіяти з браузером для завантаження нової сторінки, перевіряти журнал браузера (список завантажених раніше web-сторінок) або взаємодіяти з іншими сторінками у сусідніх фреймах.

Головним об'єктом при роботі з документами є `document`. Посилання на цей об'єкт можна отримати за допомогою атрибута цього об'єкта `window`, який є глобальним для будь-якої функції JavaScript. Для звернення до атрибутів у JavaScript використовується оператор "крапка" (`.`).

В об'єктній моделі документи згруповані у так звані **колекції**. Колекцію можна розглядати як проміжний об'єкт, що містить об'єкти власне документа. З іншого боку, колекція є масивом об'єктів, відсортованих у порядку слідування відповідних елементів у HTML-документі. Синтаксис звертання до елементів колекції такий же, як і для елементів масиву. Колекція має атрибут **`length`** — кількість всіх її документів. Колекція всіх елементів документа називається **`all`**; є також тематичні колекції: `images`, `forms`, `links` тощо (колекції усіх зображень, форм, посилань відповідно). Об'єкт може належати до якоїсь тематичної колекції, але обов'язково входить до колекції `all`.

Об'єкти, розташовані нижче в дереві об'єктів DOM, не можна викликати без зазначення об'єктів, які розташовані вище. Тобто, щоб звернутися до об'єкта `forms`, потрібно спочатку звернутися до об'єкта `window`, потім через крапку до об'єкта `document`, і тільки після цього - до `forms`: **`window.document.forms.`** Загальні правила звертання до атрибутів усіх об'єктів такі:

```
document.колекція.id_об'єкта
document.колекція["id_об'єкта"]
document.колекція[індекс_об'єкта]
```

3.3. Доступ до елементів web-сторінки

Модель DOM (англ. Document Object Model) являє собою web-документ у вигляді дерева тегів. Дерево елементів web-сторінки доступне з об'єкта `document.` Кореневим елементом будь-якого дерева web-сторінки є тег `html`. Однак замість вершини дерева можна звернутися відразу до тега `body`: `document.body`. Подальша навігація на web-сторінці може виконуватися різними способами.

Масив childNodes дає можливість звернутися до будь-якого дочірнього елемента: `document.body.childNodes[`*індекс*`]`. Масив `childNodes` підтримує властивість `length`, що дозволяє визначити поточну кількість елементів у масиві. Тому для перебору всіх елементів масиву можна скористатися наступним кодом (цикл буде працювати тільки в тому випадку, коли web-сторінка повністю завантажена):

```
for(var i=0;
i<document.body.childNodes.length; i++) {
var child = document.body.childNodes[i]
window.alert(child.tagName)
}
```

firstChild - властивість дозволяє отримати доступ до першого дочірнього елемента поточного елемента. Якщо таких елементів немає, властивість дорівнює `null`.

lastChild - дає можливість отримати доступ до останнього дочірнього елемента поточного елемента.

Наприклад, `document.body.lastChild.nodeType`.

previousSibling (nextSibling) - дозволяє звернутися до елемента дерева, який розташований ліворуч (праворуч) від поточного елемента.

parentNode - дозволяє звернутися до батьківського тега поточного елемента web-сторінки, тобто повернутися на один крок назад по дереву DOM.

getElementById. Перераховані вище властивості дозволяють послідовно одержати доступ до будь-якого дочірнього тега всередині web-сторінки. Але послідовний перебір вкладених тегів незручний. Простіше вказати ідентифікатор потрібного

тега з допомогою цього методу. Метод дозволяє звернутися до будь-якого тега на web-сторінці, для якого встановлений атрибут `id`:

```
document.getElementById(“ім'я").
```

getElementsByTagName. З допомогою моделі DOM можна послідовно звернутися до всіх тегів з заданим ім'ям на web-сторінці. *Наприклад*, щоб перебрати в сценарії всі теги `<li>`, які є на web-сторінці, скористаємося записом:

```
document.getElementsByTagName('li')
```

Метод `getElementsByTagName` повертає масив, що містить посилання на всі знайдені теги **getElementsByName.** Як ми вже відмічали, всім елементам форми присвоюється атрибут `name`, значення якого повинно бути унікальним у межах поточного тега `<form>`. Можливості JavaScript дозволяють отримати масив тегів, яким присвоєно атрибут `name` із вказаним значенням. Це робиться за допомогою методу `getElementsByName`. Однак слід враховувати, що даний метод буде працювати тільки з тими тегами, для яких у специфікації HTML явно зазначена підтримка атрибута name: `<form>, <input>, <a>, <select>, <textarea>`.

this в обробниках подій. При роботі з подіями часто виникає необхідність отримати доступ до тега, для якого було призначено цю подію. Найпростіше скористатися оператором `this`, який вказує на об'єкт, що викликав поточну подію.

3.4. Події і функції JavaScript

Події

У HTML існує можливість призначати код JavaScript певним подіям. У цьому випадку JavaScript-код буде виконуватися тільки при настанні визначеного події. **Подія** в HTML - це певна дія (клацання кнопкою миші, переміщеня покажчика, натискання клавіші тощо), виконана користувачем на web-сторінці або виконується з елементами web-сторінки. Події відносяться тільки до того тегу, якому вони призначені. *Наприклад*, якщо було призначено подію клацання кнопкою миші тега з ідентифікатором `header`, то вона настане тільки в

тому випадку, якщо користувач клацане на тегу з цим ідентифікатором.

Коли подія настає, браузер викликає обробник події - об'єкт JavaScript, що виконує певний код. Найчастіше подіям призначають не сам JavaScript-код, а виклик функції, яка виконувала б потрібний JavaScript-код. Саму ж функцію створюють в окремому JS-файлі або в тегу `script` HTML-документа. Існує чотири основних групи **обробників подій**:

- обробники подій вікна - виконують певні дії при відкриванні, закриванні, зміні розміру або переміщенні вікна браузера;
- обробники подій миші - виконують певні дії при звичайному або подвійному клацанні кнопкою миші, а також при переміщенні вказівника миші;
- обробники подій клавіатури - виконують певні дії при натисканні клавіш клавіатури;
- обробники подій форми - виконують певні дії при зміні стану елементів форми, а також при закриванні форми або відправленні її на сервер.

Сценарії в документі HTML призначені, зокрема, для обробки подій: натискання мишею на елементі документа, наведення стрілки миші на елемент чи забирання її з нього, натиснення клавіші тощо. Більшість дескрипторів HTML мають спеціальні атрибути, що визначають події, на які можуть реагувати відповідні елементи. Список допустимих подій наведено нижче; він досить великий і розрахований на всі випадки життя. Значенням такого атрибуту-події є рядок, що містить сценарій — код-обробник події. Зазвичай обробники подій оформляються у вигляді функцій, визначення яких розміщують у дескрипторі `<script>`. Розглянемо наступні приклади:

Приклад 1

```
<html>
<script>
function clickimage(){
```

```
alert (“Hello!”);
}
</script>
<img src=“pict.gif” onclick=“clickimage()”>
</html>
```

Приклад 2

```
<html>
<img src=“pict.gif” onclick=“alert(“Hel-
lo!”)”>
</html>
```

Є ще один метод, за допомогою якого можна визначати обробника подій. Майже для всіх дескрипторів HTML можна вказати атрибут **`ID` — ідентифікатор**. Його значенням є будь-який рядок, який грає роль індивідуального імені елемента в об’єктній моделі документа. З використанням цього атрибута для задання обробника події можна не використовувати атрибути-події; замість цього досить в контейнері `<script>` написати визначення функції-обробника події, ім'я якої створюється за шаблоном `значення_ID.подія()`. Розглянемо наступний фрагмент коду: *Приклад 3*

```
<html>
<h1 id= “Myheader”>Привіт усім!</h1>
<script>
function Myheader.onclick(){
alert(“Привіт”);
}
</script>
</html>
```

Браузери, зустрічаючи в HTML-документі дескриптор з визначеним ідентифікатором `ID`, створює в об’єктній моделі цього документа об’єкт з тим же ім’ям. Для цього об’єкта існує метод обробки події. Назва методу збігається з назвою події, однак синтаксис використання методу вимагає, щоб його ім'я було написане в нижньому регістрі. Елемент документа повинен бути завантажений раніше, ніж функція-обробник події. В таблиці 3.1 наведено основні події мови HTML.

Таблиця 3.1.

Основні події для документів HTML

Подія	**До яких елементів застосовна**	**Коли відбувається**	**Обробник події**
`abort`	Зображення	Користувач перериває завантаження, *наприклад*, натисканням на нове посилання чи кнопку зупинки	`onabort`
`blur`	Вікна, фрейми та всі елементи форм	Користувач переносить фокус введення з вікна, фрейму чи елемента форми	`onblur`
`click`	Всі кнопки, перемикачі, прапорці та посилання	Натискання мишею на елементі	`onclick`
`change`	Текстові поля та області, списки	Користувач змінює значення елемента	`onchange`
`error`	Зображення, вікна	Завантаження документа чи зображення призвело до помилки	`onerror`
`focus`	Вікна, фрейми та всі елементи форм	Користувач переносить фокус введення у вікно чи елемент форми	`onfocus`

продовження таблиці 3.1

load	Тіло документа	Користувач завантажує сторінку у браузер	onload
mouseout	Області, посилання	Користувач пересуває стрілку миші за межі області чи посилання	onmouseout
mouseover	Посилання	Користувач поміщає стрілку миші на посилання	onmouseover
reset	Форми	Користувач повертає форму у вихідний стан (натисканням кнопки Reset)	onreset
select	Текстові поля та області	Користувач виділяє поле введення елемента форми	onselect
submit	Кнопки Submit	Користувач передає форму на сервер	onsubmit
unload	Тіло документа	Користувач закриває сторінку	onunload

Приклад. Необхідно, щоб при наведені курсора на клітинку таблиці із написом "Привіт" з'являлося вікно повідомлення з фразою "Hello". Можливі способи розвязку:

Варіант 1:

```
<td onClick="alert('Hello')"> Привіт </td>
```

Варіант 2:

```
<script>
function Go() {
  alert("Hello")
}
</script>
<td onClick="Go()"> Привіт </td>
```

У варіанті 1 код JavaScript був записаний безпосередньо в тегу, а у варіанті 2 наслідком кліку став виклик функції. Варіант 2 доцільно використовувати, якщо код обробки події великий за обсягом.

Помилки в JavaScript

Будь-який сучасний браузер має інструменти для розробника, з допомогою яких можна, зокрема, побачити кількість помилок в сценарії і коротко прочитати про можливі причини виникнення цих помилок.

- *Mozilla Firefox*. Пункт меню **Інструменти, Web-розробка, Консоль помилок** або поєднання клавіш Ctrl+Shift+J.
- *Internet Explorer*. У повідомленні про помилки, що з'явилось при відкритті web-сторінки, натисніть кнопку Так.
- *Opera*. Пункт меню **Сторінка, Засоби розробки, Консоль помилок** або сполучення клавіш Ctrl+Shift+O.
- *Google Chrome*. Пункт меню **Інструменти, Консоль JavaScript** або поєднання клавіш Ctrl+Shift+J.

Використання функцій

Функція JavaScript - це іменована група команд, які розв'язують певну задачу та можуть повернути деяке значення. В JavaScript є вбудовані функції, які можна використовувати в програмі (їх код не можна змінити чи переглянути). Крім цих функцій, користувач може створювати власні. Функція користувача визначається оператором, що має синтаксис:

```
function Ім'я_функції ([параметри])
{
  [оператори]
```

```
  return [значення_що повертається]
}
```

Якщо функція повинна повернути деяке значення, то в її тілі використовується оператор `return`, справа від якого вказується результат, що повертається. Також ключове слово `return` може використовуватися для того, щоб достроково припинити виконання функції.

Визначення функції задає її ім'я і визначає, що буде робити функція при її виклику. Безпосереднє виконання функції реалізується, коли в сценарії відбувається її виклик та передаються необхідні параметри. Визначення функції необхідно реалізувати у будь-якому місці HTML-сторінки. У випадку невеликого JavaScript-коду функції визначаються вгорі сторінки, у випадку "об'ємного" коду – внизу. Це пов'язано з послідовним завантаженням web-сторінки. Такий підхід не впливає на загальний час завантаження сторінки, але дозволяє користувачу швидше побачити вміст сторінки.

Наприклад, для показу на екрані вікна повідомлення з текстом "Це виклик функції" визначимо функцію `Go` та реалізуємо її виклик:

```
<html>
<head>
<title>Використання JavaScript</title>
<script>
function Go() {
alert("Це виклик функції");
}
</script>
</head>
<body>
<script>Go();</script>
</body>
</html>
```

Ініціалізація в тілі функції змінної з ключовим словом `var` створює локальну змінну, навіть якщо вона вже була оголошена у зовнішній програмі. *Приклад:*

```
function result(a,b){
```

```
  c=a+b;
return c;
}
```

Якщо функція не має аргументів і не повертає ніякого значення, то викликати її можна командою ім'я_функції(). Якщо для роботи функції необхідні якісь аргументи, функцію викликають наступним чином:

ім'я_функції (знач_аргум1, знач_аргум2,…).

Вбудовані функції

JavaScript надає у розпорядження програміста велику кількість вбудованих (стандартних) функцій. Наведемо *приклади* декількох з них:

parseInt(рядок, основа) — перетворює вказаний рядок на ціле число у системі числення за вказаною основою (8, 10, 16); якщо основа не вказана, то за замовчуванням використовується 10.

parseFloat(рядок, основа) — перетворює вказаний рядок у число з плаваючою крапкою.

isNaN(значення) — повертає true, якщо вказане значення не є числом, і false в іншому випадку.

eval(рядок) — обчислює вираз у вказаному рядку; вираз повинен бути написаний мовою JavaScript (не містити дескрипторів HTML).

escape(рядок) — повертає рядок у вигляді %XX, де XX — ASCII-код вказаного символу.

unescape(рядок) — здійснює обернене перетворення.

typeOf(об'єкт) — повертає тип вказаного об'єкта у вигляді символьного рядка.

Питання для самоконтролю

1. Назвіть події, що можуть бути опрацьовані у скрипті.
2. Яким чином JavaScript дозволяє опрацьовувати події?
3. Що називають JavaScript-функцією?
4. Правила запису функцій у JavaScript.
5. Яким чином в JavaScript можна створювати функції?
6. Вбудовані (стандартні) функції JavaScript.

3.5. Стандартні об’єкти і об’єкти користувача

В ядрі JavaScript визначені **об’єкти та функції**, які можна застосовувати не використовуючи контекст завантаженої сторінки. До основних об’єктів належать: `Array, Date, Math, String`.

`Array` (масив) - це набір даних, до окремих елементів якого можна звернутися за ім’ям та індексом.

Для організації масиву необхідно використати одну із двох конструкцій:

```
ім’я_масиву = new Array([елемент1], [елемент2], [елемент3],...)
ім’я_масиву = new Array([довжина масиву])
```

Зауважимо, що перший елемент масиву має індекс 0. В першій конструкції параметрами служать елементи масиву, а в другій - значення довжини масиву.

Наприклад:

```
ar1 = new Array(1, 2, 3)
ar2 = new Array(3)
```

Для доступу до значень елементів масиву, після імені масиву в квадратних дужках вказують індекс (номер) елемента.

Наприклад:

```
a = ar1[2]
ar1[0] = 7
```

Змінній *a* присвоюється значення елемента масиву за номером *2*, а елемент масиву за номером *0* стає рівним *7*.

Особливістю масивів JavaScript є те, що розмір масиву може встановлюватися динамічно.

Наприклад, якщо для масиву із попереднього прикладу написати `ar1[100] = 7`, то розмір масиву буде автоматично встановлений рівним *101*.

Для визначення довжини масиву можна скористатися властивістю `length`.

Наприклад:

```
a = ar1.length
```

JavaScript дуже динамічна мова програмування. Елементами масиву можуть бути різнотипні елементи.

Наприклад: `var array = [‘вік’,5,6.6,“оцінка”];`

Масиви у JavaScript можуть використовувати своїми елементами інші масиви. Це дозволяє створення багатовимірних масивів у вигляді:

```
var matrix = [[1,2,3],[4,5,6],[7,8,9]];
```

Для доступу до елементів у масиві масивів достатньо використати квадратні дужки двічі підряд:

```
document.write(matrix[2][1]);//вибрано елемент 4
```

Отже, будь-який елемент `matrix[n]` – це масив чисел. Для доступу до певного елемента масиву потрібно записати `matrix[n][m]`, де у других квадратних дужках вказується індекс елемента внутрішнього масиву.

Слід зазначити, що внутрішні елементи можуть мати різну розмірність, а також, в свою чергу, містити ще власні внутрішні масиви. Це дає змогу у JavaScript створювати масиви довільної структури і розмірності.

Зручність використання масивів створюють методи, вміщені в таблиці 3.2.

Таблиця 3.2.

Методи об'єкта `Array`

Метод	**Призначення**
`concat`	Об'єднує два масиви в один
`join`	Об'єднує всі елементи масиву в один рядок
`pop`	Знищує останній елемент масиву і повертає його значення
`push`	Додає один або декілька елементів в кінець масиву і повертає останній доданий елемент
`reverse`	Переставляє елементи масиву в зворотному порядку
`shift`	Знищує перший елемент масиву і повертає його значення
`slice`	Створює перетин масиву в вигляді нового масиву
`splice`	Додає та/або знищує елементи масиву
`sort`	Сортує елементи масиву
`unsift`	Додає один або більше елементів на початок масиву та повертає нову довжину масиву

Об'єкт **Date** використовується для роботи з датами. *Синтаксис* оператора створення об'єкта дати:

```
ім'я_об'єкта_дати = new Date([параметри])
```

Якщо параметри відсутні, то значенням об'єкта буде поточна дата. Параметром може бути рядок: "місяць день, рік години: хвилини: секунди". *Наприклад*, для створення дати "5 лютого 2015 року 23:12:07" можна записати:

```
day = new Date("February 5, 2015 23:12:07")
```

Прочитати або змінити параметри створеного об'єкта `Date` можна за допомогою ряду методів. Методи, які найчастіше використовуються, наведені в таблиці 3.3.

Таблиця 3.3.

Методи об'єкта `Date`

Метод	Призначення
`getDate`	Повертає число місяця для вказаної дати
`getDay`	Повертає день тижня для вказаної дати
`getHours`	Повертає годину для вказаної дати
`getMinutes`	Повертає хвилини для вказаної дати
`getMonth`	Повертає місяць для вказаної дати
`getSeconds`	Повертає секунди для поточного часу
`getYear`	Повертає рік для вказаної дати
`setDate`	Встановлює число місяця для вказаної дати
`setDay`	Встановлює день тижня для вказаної дати
`setHours`	Встановлює годину для вказаної дати
`setMinutes`	Встановлює хвилини для вказаної дати
`setMonth`	Встановлює місяць для вказаної дати
`setSeconds`	Встановлює секунди для вказаної дати
`setYear`	Встановлює рік для вказаної дати

Об'єкт **Math** дозволяє використовувати вбудовані в JavaScript математичні функції та константи. При зверненні до методів та властивостей цього об'єкта створювати його не потрібно, але необхідно вказувати його ім'я.

Наприклад, присвоїти змінній *a* результат обчислення тригонометричної функції від аргумента в радіанах можна так:

```
a = Math.sin(1).
```

Методи об'єкта `Math`, що використовуються найчастіше наведені у таблиці 3.4.

Об'єкт **`String`** використовується для роботи з рядковими типами даних. Створити об'єкт `String` можна присвоюєнням змінній рядкового літералу:

```
a = "Неявний спосіб створення рядкового об'єкта"
```

Крім того, можна явно створити рядковий об'єкт, використовуючи оператор `new` та конструктор `String`:

```
ім'я_об'єкта = new String(Рядок)
```

Таблиця 3.4

Методи об'єкта `Math`

Метод	**Призначення**
`abs`	Повертає абсолютне значення змінної
`sin, cos, tan`	Повертають значення тригонометричних функцій. Аргументи задаються в радіанах
`acos, asin, atan`	Повертають значення обернених тригонометричних функцій
`exp, log`	Повертають значення експоненційної функції та функції натурального логарифма відповідно
`ceil`	Повертає найменше ціле число, більше або рівне значенню аргументу
`floor`	Повертає найбільше ціле число, менше або рівне значенню аргументу
`max, min`	Повертає найбільше/найменше значення з двох аргументів
`pow`	Повертає значення функції $pow(x,y)=x^y$
`round`	Повертає значення аргументу, заокруглене до найближчого цілого числа
`sqrt`	Повертає квадратний корінь з аргументу

Параметром конструктора може бути будь-який рядок.

Наприклад: `a = new String("Явний спосіб створення рядкового об'єкта")`

Єдиною властивістю об'єкта `String` є **`length`**, що зберігає довжину рядка.

Наприклад, для присвоєння змінній *h* значення довжини рядка `a` необхідно записати: `h=a.length`

Методи об'єкта `String`, що використовуються найчастіше, наведені в таблиці 3.5.

Запишемо *приклад* використання методу `toLowerCase` для переведення рядкової змінної *a* у нижній регістр:

```
a=a.toLowerCase()
```

Таблиця 3.5.

Методи об'єкта `String`

Метод	Призначення
`anchor`	Створює HTML якір, який використовується як гіпертекстове посилання
`link`	Створює гіпертекстове посилання, за яким можна перейти на інший URL
`fontsize`	Виводить рядок із встановленим розміром шрифту
`bold`	Виводить рядок, що відображається напівжирним шрифтом
`italics`	Виводить рядок, що відображається курсивом
`strike`	Виводить рядок, що відображається перекресленим шрифтом
`substring`	Повертає частину рядка об'єкта `string`
`sub`	Виводить рядок, що відображається як нижній індекс
`sup`	Виводить рядок, що відображається як верхній індекс
`toLowerCase, toUpperCase`	Переводить зміст рядка у нижній/верхній регістр

На додаток до стандартних об'єктів JavaScript існує декілька функцій, що дістали назву "функцій верхнього рівня", для виклику яких не потрібно створювати об'єктів. До цих функцій належать **parseFloat(параметр)** та **parseInt(параметр).**

Їх призначенням є аналіз рядкового аргументу та повернення відповідно числа з плаваючою крапкою або цілого числа. Також часто використовуються функції **Number(об'єкт)** та **String(об'єкт),** які перетворюють об'єкт, що використовується як параметр, в число або рядок.

Використання об'єктів window, document, location

Об'єкт **window** створюється автоматично при запуску браузера. Крім того, нове вікно можна створити і засобами JavaScript. Для цього необхідно використати метод **open**. Синтаксис методу такий:

```
Ім'я_змінної = window.open([Ім'я_ файлу], [Ім'я_вікна], [Параметри])
```

Ім'я_змінної – це ім'я для звернення до нового вікна в програмі JavaScript.

Ім'я_ файлу – повна або відносна URL-адреса документа, що буде відкриватись у вікні браузера.

Ім'я_вікна – це ім'я вікна, що вказане як мета гіпертекстового посилання з іншого HTML-документа.

Параметри – задають значення параметрів вікна. Основні параметри вміщені в таблиці 3.6. Якщо існують значення властивості *yes* або *no*, то в стандартних налаштуваннях використовується значення *yes*.

Таблиця 3.6.

Основні параметри вікна

Назва	Призначення	Можливі значення
directories	Наявність/відсутність панелі "Посилання"	yes/no
height	Висота вікна	кількість пікселів

продовження таблиці 3.6

`location`	Наявність/відсутність адресного рядка	yes/no
`menubar`	Наявність/відсутність рядка меню	yes/no
`resizable`	Можливість/неможливість зміни розмірів вікна користувачем	yes/no
`scrollbars`	Наявність/відсутність смуг прокрутки вікна	yes/no
`status`	Наявність/відсутність рядка стану браузера	yes/no
`toolbar`	Наявність/відсутність панелей інструментів	yes/no
`width`	Ширина вікна	кількість пікселів

Наприклад, для створення нового вікна браузера, в якому буде завантажено файл *a.html*, можна записати:

```
<script>
  myw=window.open("a.html","displayWindow",
  "width=400,height=300,status=no,toolbar=no,
  menubar=no")
</script>
```

При цьому ширина вікна дорівнює 400 пікселів, висота вікна 300 пікселів; рядок стану вікна, панель інструментів та рядок меню будуть відсутні.

Зауважимо, що код необхідно записати в одному рядку.

Об'єкт `window` перебуває на вершині ієрархії об'єктів DOM. Він відповідає за саме вікно браузера і все, що з ним пов'язано.

Для закривання вікна браузера використовується метод `close`.

Наприклад, щоб закрити вікно попереднього прикладу, необхідно задати: `myw.close()`

При звертанні до методів та властивостей вікна браузера, в якому розміщена програма JavaScript, ім’я вікна або ключове слово `window` можна не вказувати.

Наприклад, закрити поточне вікна браузера можна так: `close()`

Об'єкт **`document`** містить інформацію про завантажену сторінку. Всі елементи HTML-сторінки є властивостями цього об'єкта. Найчастіше в об'єкті document використовують метод **`write`**, за допомогою якого можна зробити запис у HTML-сторінку.

Таблиця 3.7.

Об'єкти, безпосередньо підпорядковані `window`

Назва	Звернення, призначення
`location`	`window.location`. Зберігає інформацію про поточну web-сторінку: адреса сайту, адреси сторінок тощо
`history`	`window.history`. Містить інформацію про сторінки сайту, які користувач відвідав до цього. Список цих сторінок називається **стеком історії браузера**
`navigator`	`window.navigator`. Містить інформацію про браузер і ОС користувача, який в даний момент відкрив сторінку
`screen`	`window.screen`. Містить інформацію про монітор користувача, який в даний момент відкрив web-сторінку (ширина і висота екрану в пікселях)
`document`	`window.document`. Містить інформацію про web-сторінку, яка в даний момент відкрита у вікні браузера
`forms`	`window.document.forms`. Є колекцією і містить масив форм, які є на web-сторінці, відкритої в даний момент у вікні браузера
`images`	`window.document.images`. Є колекцією і містить масив зображень, які є на web-сторінці, відкритій в даний момент у вікні браузера
`links`	`window.document.links`. Є колекцією і містить масив посилань, які є на відкритій сторінці

Наприклад, для запису рядка "Привіт JavaScript" в документ, де розміщено сценарій, необхідно ввести:

```
document.write("Привіт JavaScript")
```

Ще одним популярним методом цього об'єкта є `getElementById`, що реалізує доступ до будь-якого об'єкта з визначеним `id`. Синтаксис методу такий:

```
document.getElementById(ім'я_id)
```

Наприклад, встановлення значення стилю `display` елемента з `id = myid` рівним `block`, можна реалізувати так:

```
document.getElementById("myid").style.
display="block"
```

Властивості об'єкта **`location`** дозволяють одержати інформацію про URL-адресу завантаженої HTML-сторінки. Метод **`reload`** дозволяє перезавантажити в браузер поточну HTML-сторінку. Метод **`replace`** завантажує у вікно браузера сторінку, адреса якої використана як параметр цього методу.

Об'єкти користувача JavaScript

Мова JavaScript забезпечує механізм для інкапсуляції. **Інкапсуляція (`encapsulation`)** — це можливість приховування даних всередині об'єкта. У мові JavaScript можна створити **екземпляр родового об'єкта (`generic object`)** і призначити йому атрибути і навіть методи.

Наприклад, у наступному фрагменті коду на JavaScript створюється екземпляр родового об'єкта з іменем `cartLineItem` (один з товарів у віртуальному кошику покупця). За допомогою оператора „крапка" (`.`) задаються значення чотирьох вказаних користувачем властивостей.

Приклад.

```
LineItem= new Object();
LineItem.productID = 'MG1234';
LineItem.productName = 'MGB Roadster
(1935)';
LineItem.qty =1;
LineItem.unitPrice =36000;
```

Екземпляр `LineItem` можна використовувати в подальшому

у будь-якому сценарії JavaScript з відповідним посиланням:

```
alert('В вашому кошику є '+cartLineItem.qty
+ ' ' +  LineItem.productName);
```

Для об'єктів можна також визначати **методи**.

Наприклад, наступну функцію `total()` можна використовувати як метод. Вона звертається до свого об'єкта за допомогою оператора **`this`**:

```
function total(){
return (this.qty * this.unitPrice);
}
```

Доступ до функції здійснюється таким же чином, як і до атрибутів:

```
LineItem.total();
```

Тепер цю функцію можна викликати безпосередньо для об'єкта `LineItem`:

```
LineItem= new Object();
LineItem.productID = 'MG1234';
LineItem.productName = 'MGB Mk I Roadster';
LineItem.qty = 2;
LineItem.unitPrice = 12500;
LineItem.total = total;
document.write('<p>' + LineItem.qty + ' ' +
LineItem.productName +'коштує $' +
LineItem.total() + '</p>');
```

Прототип у JavaScript нагадує конструктор в C++.

Прототип (англ. *prototype*) — це функція, що створює визначений користувачем об'єкт та ініціалізує його атрибути. Прототип для об'єкта JavaScript повинен також містити функції об'єкта.

Прототип об'єкта `cartLineItem` має такий вигляд:

```
function cartLineItem(id, name, qty, price){
this.productID = id;
this.productName = name;
this.qty = qty;
this.unitPrice = price;
this.total = total;
}
```

За допомогою прототипів здійснюється наслідування в JavaScript.

Якщо прототип визначений, то масив екземплярів `LineItem` можна створити за допомогою наступного фрагмента коду JavaScript:

```
  var LineItem = new Array();
  LineItem[0]= new cartLineItem (‘MG123’,‘MGB
Mk I Roadster’,1,36000);
  LineItem[1]= new cartLineItem
(‘AH736’,‘Austin-Healey Sprite’,1, 9560);
  LineItem[2]= new cartLineItem
(‘TS225’,‘Triumph Spitfire Mk I’,1, 11000);
```

Питання для самоконтролю

1. Що називають об'єктом JavaScript, методами та властивостями об'єкта?
2. Як визначити об’єкт користувача в JavaScript?
3. Які стандартні об’єкти у JavaScript?
4. Опишіть ієрархію об'єктів JavaScript.
5. Як звернутися в скрипті до об'єкта, що міститься у HTML-документі?
6. Засоби для доступу до елементів web-сторінки.
7. Методи та властивості об’єкта Array.
8. Методи та властивості об’єкта Date.
9. Методи та властивості об’єкта Math.
10. Методи та властивості об’єкта String.
11. Методи та властивості об’єкта Window.
12. Методи та властивості об’єкта Document.
13. Методи та властивості об’єкта Location.

3.6. Робота з тегами і властивостями CSS у JavaScript

Робота з властивостями CSS у JavaScript. Ще однією корисною особливістю JavaScript є можливість встановлювати нові **CSS-властивості** або змінювати значення існуючих властивостей для будь-якого тега на web-сторінці. Щоб

встановити CSS-властивість, потрібно скористатися DOM-властивістю **style**, яка підтримується всіма елементами web-сторінки. Тобто нам спочатку потрібно отримати доступ до потрібного елемента web-сторінки і тільки потім викликати властивість `style` для нього.

Наприклад:

```
document.getElementById("ім'я").style
```

У будь-якому випадку після DOM-властивості `style` через крапку потрібно вказати CSS-властивість, значення якої потрібно змінити:

```
document.getElementById("ім'я").style.width
```

Це просто звертання до CSS-властивості. Для зміни її значення, слід виконати операцію присвоювання:

```
document.getElementById("ім'я").style.width="
100px";
```

Зміна вмісту тега. Можливості JavaScript дозволяють не тільки призначати тегам нові властивості CSS, а й змінювати вміст будь-якого тега.

Для цього використовується властивість **innerHTML**, яка підтримується всіма елементами web-сторінки.

Щоб **отримати поточний вміст тега**, можна використати запис:

```
var w = document.getElementById("ім'я").in
nerHTML;
```

Для **зміни вмісту тега** можна скористатися записом:

```
document.getElementById("ім'я").innerHTML="
текст";
```

Доповнити вміст тега новим значенням можна за допомогою операції конкатенації:

```
document.getElementById("ім'я").innerHTML+=
"текст";
```

Властивість **innerHTML** розміщує текст всередині даного тега. Властивість **outerHTML** дозволяє розмістити текст замість цього тега. Покажемо це на *прикладах*. Нехай маємо фрагмент коду:

```
<div id="mtext">test</div>.
```

Тоді:

```
  document.getElementById("mtext").innerHTML="
текст";
  // <div id="mtext">текст</div>
  document.getElementById("mtext").outerHTML="
текст";
  // текст (тег з ідентифікатором mtext
видалено з web-сторінки)
  document.getElementById("mtext").innerHTML+=
"текст";
  // <div id="mtext">testтекст</div>
  document.getElementById("mtext").outerHTML+=
"текст";
  // <div id="mtext">test</div>текст
```

Змінюємо атрибути тега. За допомогою JavaScript можна додавати, вилучати та змінювати атрибути тега. Для цього використовуються спеціальні методи або пряме звернення до атрибута.

setAttribute. Метод `setAttribute("ім'я", "значення")` дозволяє присвоїти `"значення"` атрибуту `"ім'я"`.

Наприклад:

```
document.getElementById("myimg").setAttribute(
"alt","лого");
```

getAttribute. Для отримання поточного значення атрибута досить скористатися методом `getAttribute("ім'я")`.

Наприклад:

```
  var myattr=document.getElementById("myimg").
getAttribute("alt");
```

hasAttribute. Перед тим, як використовувати метод `getAttribute`, бажано перевірити, чи існує взагалі потрібний атрибут у цього тега. Це робиться за допомогою методу `hasAttribute("ім'я")`:

```
  if(document.getElementById("myimg").hasAttri
bute("alt"))
```

```
{
  var myattr=document.getElementById("myimg").
getAttribute("alt");
  }else
  {document.getElementById("myimg").setAttribu
te("alt","так");
  }
```

removeAttribute. Ще один метод для роботи з атрибутами, який дозволяє вилучити атрибут даного тега. Перед цим слід впевнитися, що він існує:

```
  if(document.getElementById("myimg").hasAttri
bute("alt"))
  {
  document.getElementById("myimg").removeAttri
bute("alt");
  }
```

Пряме звернення до атрибуту. У моделі DOM для доступу до основних стандартних атрибутів можна використовувати не лише перелічені вище методи, але і пряме звернення до атрибута, тобто:

```
document.getElementById("ім'я").атрибут
```

Наприклад: `document.body.id = "mytheme";`

Створення, публікація і вилучення тегів

Для **створення** нових тегів призначений метод **createElement** об'єкта **document**. Аргументом цього методу вказують тег, який потрібно створити. Повернене значення зберігають у будь-якій змінній:

```
var newDiv = document.createElement('div');
```

Після створення тега він буде знаходитися у зазначеній змінній `newDiv`. Далі йому можна додати певні атрибути:

```
newDiv.className = 'myclass';
newDiv.id = 'myid';
```

За необхідності тегу присвоюють властивості CSS:

```
newDiv.style.backgroundColor = 'white';
```

Крім того, тег можна наповнити вмістом:

```
newDiv.innerHTML = 'Наш новий тег';
```

Це основне, що необхідно для створення нових тегів. Наступний етап - навчитися виводити створений тег на web-сторінку.

Публікація. Після того як тег створений, його потрібно додати на web-сторінку. Для цього спочатку потрібно визначити, де його розмістити, тобто після або перед яким із вже існуючих на web-сторінці тегів варто додати створений тег.

Для цього зазвичай обирають один із наступних методів:

appendChild(новий тег) - додати тег в кінець даного елемента (останнім у списку дітей цього елемента);

insertBefore(новий тег, перед яким тегом вставити) – додати новий тег перед дочірнім тегом, який був вказаний як другий аргумент методу.

Наприклад:

```
var newDiv = document.createElement('div');
document.getElementById('oldDiv').appendChil
d(newDiv);
```

або

```
var newLi = document.createElement('li');
document.getElementById('listUL').insertBefo
re(newLi,document.getElementById('listUL').get
ElementsByTagName('li')[0]);
```

Для **вилучення** тега з web-сторінки використовується метод **removeChild(елемент)**. Викликати цей метод потрібно у батьківського тега, який необхідно вилучити: `батько.removeChild(дитина)`.

Наприклад:

```
document.getElementById('listUL').removeChil
d(document.getElement-ById('listUL').get
ElementsByTagName('li')[0]);
```

Якщо посилання на батьківський тег немає, можна скористатися послугами проміжної властивості `parentNode`, яка повертає батьківський елемент цього елемента:

```
document.getElementById('oldDiv').parentNod
.removeChild(document.getElementById('oldDiv')
);
```

Питання для самоконтролю

1. Робота із властивостями CSS.
2. Зміна властивостей CSS, атрибутів тега.
3. Створення і вилучення тегів.

3.7. Використання таймерів

В JavaScript існує ще одна цікава можливість, пов'язана з функціями. Користувач може запустити функцію через зазначений час або призначити багаторазовий запуск функції через вказаний інтервал часу. Все це реалізується за допомогою **таймерів**. Таймер буває: **одноразовий** - виконує запуск функції лише один раз; **багаторазовий** - виконує запуск функції через певний інтервал часу.

Для **створення таймерів** використовують метод **`setTimeout`**, який є методом об'єкта `window`**.** Він дає можливість запрограмувати виконання деяких команд після закінчення встановленого проміжку часу.

Синтаксис методу такий:

```
setTimeout("Код_JavaScript",інтервал_часу)
```

Першим параметром функції `setTimeout`, зазвичай, вказують функцію. Цю функцію слід визначити на HTML-сторінці до використання функції `setTimeout`. Другим параметром функції `setTimeout` є інтервал часу, який задається в мілісекундах. Він визначає, через який проміжок часу після виклику методу **`setTimeout`** буде виконана зазначена функція (або набір команд).

Наприклад, виконання функції *myfunction* через *3000 мілісекунд* після завантаження HTML-сторінки можна запрограмувати так:

```
setTimeout("myfunction()",3000)
```

Призначений, але ще не виконаний таймер можна скасувати за допомогою методу **`clearTimeout`**. Однак для цього вказаному методу потрібно передати ідентифікатор таймера, який слід відмінити. Отримати ж ідентифікатор таймера можна при оголошенні методу `setTimeout`:

```
var mytimer = window.setTimeout(ім'я
функції, інтервал);
```

```
window.clearTimeout(mytimer);
```

Для створення **багаторазових таймерів** використовується метод **`setInterval`**, який має ті ж аргументи, що і метод `setTimeout`:

```
window.setInterval(ім'я функції, інтервал);
```

Метод `setInterval` повертає ідентифікатор створеного таймера.

Багаторазовий таймер безперервно запускає функцію із вказаним інтервалом часу до тих пір, поки користувач не закриє сторінку або поки багаторазовий таймер не буде скасований за допомогою методу `clearInterval(`ідентифікатор таймера`)`.

Методу `clearInterval` необхідно передати ідентифікатор таймера, який повертається методом `setInterval`. Для прикладу встановимо таймер, який би запускав функцію `myFunction` кожні п'ять секунд:

```
var myTimer=setInterval("myFunction()", 5000);
```

Надалі, якщо нам знадобиться припинити запуск функції `myFunction`, ми зможемо скористатися командою

```
window.clearInterval(myTimer).
```

Функція `setTimeout` часто використовується для створення **анімаційних ефектів**. Це може бути, *наприклад*, циклічна зміна кольору тексту, або циклічна заміна одного зображення іншим.

Таймери працюють у фоновому режимі, тобто після установки таймера робота web-сторінки і всіх JavaScript сценаріїв на ній не припиняється (як і коду, який вказаний після методу `setTimeout`).

Питання для самоконтролю

1. Призначення таймерів та їх різновиди.
2. Як за допомогою JavaScript запрограмувати виконання деяких команд після закінчення встановленого терміну часу?

3.8. Стандартні діалогові вікна

Для спілкування з відвідувачем у JavaScript використовуються діалогові вікна. Існує три **методи** для **виклику діалогових вікон**:

- **alert**(повідомлення) - відображає інформаційне діалогове вікно із вказаним повідомленням і однією кнопкою для закривання вікна;
- **confirm**(повідомлення) - виводить підтверджувальне діалогове вікно із вказаним повідомленням і двома кнопками для закривання вікна;
- **prompt**(повідомлення, за замовчуванням,заголовок) - відображає діалогове вікно із зазначеним повідомленням, двома кнопками для закривання вікна, а також полем для введення даних.

Всі ці методи належать об'єктові window. Тому викликати їх можна або із зазначенням об'єкта (window.alert), або без нього (alert). Розглянемо ці методи детальніше.

Метод **window.alert** призначений лише для інформування користувача. Діалогове вікно, яке відкривається з його допомогою, містить лише одну кнопку для закривання діалогового вікна.

Метод **window.confirm** не тільки інформує відвідувача про подію, але й надає йому вибір: клацнути кнопку **ОК** або **Скасувати**. Обидві ці кнопки закривають діалогове вікно. Якщо була натиснута кнопка ОК, повертається значення true. Якщо Скасувати - значення false:

```
  var res = window.confirm("Метод window.
confirm()");
  if(res==true){
  window.alert('Натиснута кнопка ОК');
  } else{
  window.alert('Натиснута кнопка Скасування');
  }
```

Ще більше можливостей надає метод **window.prompt**. Він не лише виводить зазначене повідомлення, але і дозволяє користувачеві відповісти на повідомлення. Метод prompt

повертає відповідь користувача. Причому відповідь залежить від кнопки, яку відвідувач натиснув для закривання діалогового вікна:

- **кнопка ОК** - повертає текст, який користувач написав у діалоговому вікні;
- **кнопка Скасувати** - повертає `null`, навіть якщо в діалоговому вікні написаний будь-який текст.

На відміну від раніше перелічених методів, метод `prompt` вимагає вказати три аргументи:

1. Містить повідомлення, яке буде відображатися в діалоговому вікні;
2. Включає в себе значення за замовчуванням, яке буде показуватися у полі для введення відповіді;
3. Містить заголовок для діалогового вікна, що виводиться.

Наприклад:

```
while(1==1){
var res = window.prompt('Скільки вам
  років?','16','Вікові обмеження');
if(res===null){
window.alert('Будь ласка, дайте відповідь на
  питання');
continue;
}
if(parseInt(res)>=18){
window.alert('Ласкаво просимо на наш сайт');
}else{
window.alert('На жаль, Ви повинні залишити
  наш сайт.
Побачимося через '+(18-parseInt(res))+'
  років');
}
break;
}
```

Питання для самоконтролю

1. Як реалізована інтерактивність у JavaScript?
2. Як у JavaScript відкрити нове вікно браузера?

4. Завдання для лабораторних робіт

Лабораторна робота № 1

***Тема**: "Робота з браузером, створення найпростішої сторінки за допомогою програми Notepad++. Виведення тексту та його редагування. Списки"*

Завдання 1: Створіть найпростішу web-сторінку

Рівень І

1. В папці із своїм прізвищем створіть папку власної web-сторінки, в папці web-сторінки створіть файл-пояснення призначення основних тегів, в ньому наберіть наступну інформацію:

Теги:

<html>
</html> <!-- початок та закінчення web-сторінки-->
<head>
</head > <!--заголовок сторінки -->
<title>
</title> <!--назва заголовку вікна -->
<body>
</body> <!-- тіло сторінки -->
<h1>
</h1> <-- вивід заголовку у вікні-->

Параметри:

Align <!--вирівнювання тексту можливі значення left, center, right->

Bgcolor <!--колір фону сторінки-- >

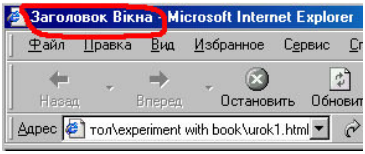
2. Відкрийте програму Notepad++ (Блокнот) і створіть свою першу html-сторінку:

```
<html>
<head>
<title> Заголовок Вікна </title>
</head>
<body>
</body>
</html>
```

3. Збережіть файл як web-сторінку в папку web-сторінки:
   - Файл / Сохранить;

- Папка *lab1*
- Имя файла: *task1a.html*;
- Тип файла *Все файлы(*.*)*;
- Сохранить.

4. Відкрийте файл у браузері, зверніть увагу на верхній лівий кут вікна:



Зауваження: для зручнішої та швидшої роботи одночасно відкрийте два вікна, перше – файл у редакторі Notepad++ ; друге - файл у браузері. Зробивши зміни у редакторі, можна їх переглянути в браузері, натиснувши клавішу **F5** або виконуючи щоразу команду **Обновить**.

5. Зробіть так, щоб на web-сторінці замість слів "*Заголовок Вікна*" було написано "*Прізвище та ім'я*".

6. У тегу `<body>` наберіть наступний рядок:

```
<h1> Hello HTML! </h1>
```

Збережіть файл та перегляньте зміни у браузері.

7. Аналогічно, використовуючи тег `<h1>`, виведіть на екран привітання: "*Привіт, мене звати (ваше ім'я)*"

8. У рядку "*Hello HTML!*" в тег `<h1>` допишіть `align=right`, збережіть і перегляньте зміни:

```
h1 align=right> Hello HTML! </h1>
```

9. Вирівняйте весь текст до середини сторінки (`align=center`).

10. В тегу `<body>` допишіть параметр `bgcolor=red`, перегляньте, що при цьому змінилося у браузері, змініть його значення на: *blue, green, yellow, silver*:

```
<body bgcolor=red>
```

Рівень II

Створіть web-сторінку з наступними параметрами:

- назва файлу *task1b.html*;
- розміщення файлу ...*мої web-сторінки\lab1*\;

- Заголовок вікна *"Мій перший сайт"*;
- На web-сторінці повинна міститися інформація:
  - Прізвище, ім'я, по-батькові вирівняне до середини;
  - Університет, спеціальність вирівняні до лівого краю;
  - Домашня адреса та телефон - до правого краю;
- Колір фону сторінки - морська хвиля (*navy*)**.**

Завдання 2: Організуйте виведення тексту та розгляньте елементи його редагування

Рівень І

1. Створіть такий HTML-документ:

```
<html>
<head>
<title> Завдання 2 </title>
</head>
<body>
<h1> Web-дизайн </h1>
</body>
</html>
```


Збережіть та перегляньте його у браузері

2. Змініть заголовок вікна на своє ім'я, а слово "*Web-дизайн"* вирівняйте до середини.
3. Запишіть тег `<h1>`, у ньому слово "*Привіт*" вирівняйте посередині.
4. Змініть цифру *1* в тегу `<h1>` на будь-яку від *2* до *6*.
5. Запишіть тег абзацу `<p>`, а в ньому слова "*Ви завітали*" і вирівняйте їх до лівого краю.
6. Запишіть тег абзацу `<p>`, а в ньому слова "*на сторінку*" і вирівняйте їх до середини.
7. Запишіть тег абзацу `<p>`, а в ньому своє прізвище та ім'я у родовому відмінку (*наприклад*, *Сидоренка Віктора*) і вирівняйте їх до правого краю.
8. Запишіть посередині таке речення "*Ось мій улюблений вірш*": та наберіть такий текст (використайте тег `<br>`):

Вечірнє сонце, дякую за день!

Вечірнє сонце, дякую за втому.
За тих лісів просвітлений Едем
І за волошку в житі золотому.
За твій світанок, і за твій зеніт,
І за мої обпечені зеніти.
За те, що завтра хоче зеленіть,
за те, що вчора встигло оддзвеніти

9. Під віршем проведіть лінії товщиною *4, 8, 16* та довжиною *80%, 50%* та *30%* відповідно.

10. Запишіть посередині сторінки наступне речення "*Ось моя улюблена проза"* та скопіюйте уривок прози з файлу; вирівняйте уривок по ширині сторінки.

Рівень II

Створіть web-сторінку з наступними атрибутами:

- назва файлу *task2b.html*;
- розміщення файлу ...*мої web-сторінки\самостійні роботи*;
- заголовок вікна *Робота з текстом;*
- колір фону сторінки *зелений*;
- колір тексту *жовтий*;
- вгорі web-сторінки повинна розміститися наступна інформація:
  - три заголовки різних рівнів: *Web дизайн, 2016 рік*, *автор - власне ім'я*
  - лінія довжиною *57%* і розміром *6*;
  - уривок довільного тексту (до 4 рядків);
  - лінія довжиною *300 пікселів* з розміром *2*;
  - копія тексту, вирівняна по ширині сторінки.

Завдання 3: Опрацюйте можливості html для редагування текстів, перегляньте варіанти оформлення списків

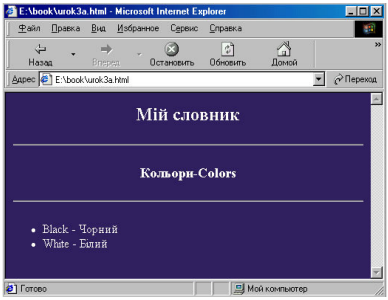
Рівень I

1. Відкривши *Notepad++*, створіть сторінку з наступними атрибутами:

- Заголовок вікна – *Повторення*;
- Колір фону сторінки - *зелений*, колір тексту – *жовтий*;
- Наберіть три тексти заголовків трьох різних рівнів:

1) *HTML*, 2) *web-дизайн*, 3) *Notepad* ;

- Проведіть лінію довжиною *400 пікселів*, шириною *6*;
- Наберіть уривок з улюбленого вірша і вирівняйте його до середини сторінки;
- Скопіюйте уривок прози у свою сторінку, вирівняйте по ширині сторінки. Наведемо *приклад* результатів роботи:



Створіть нову web-сторінку. Посередині запишіть слова *Мій словник*, проведіть горизонтальну лінію.

2. Напишіть ще один заголовок, але меншим шрифтом *Кольори – Colors*, проведіть ще одну лінію.

3. Створіть маркований список, в якому вкажіть найбільшу кількість кольорів англійською мовою та їх переклад українською:

```
<ul> <li> black – чорний </li>
.....інші кольори
</ul>
```

4. Аналогічно створіть нову сторінку з маркованим списком наступного змісту: Напис 1 – *Університет*, Напис 2 – *Рейтинг предметів*, у списку вкажіть 5 улюблених предметів у спадному порядку.

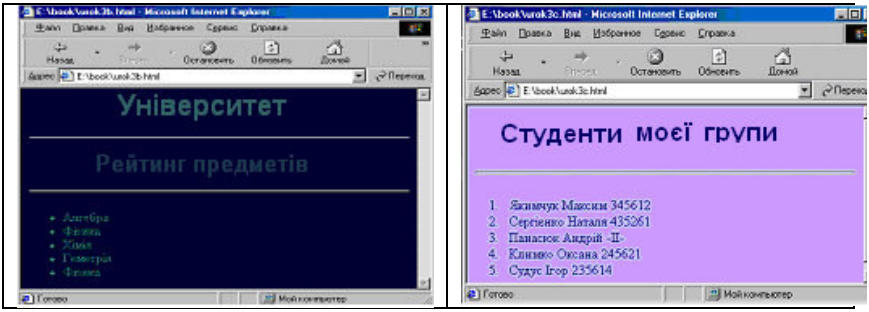
5. На новій web-сторінці наберіть заголовок *Студенти моєї групи*, проведіть горизонтальну лінію та створіть нумерований список, кожним пунктом якого буде прізвище, ім'я та телефон кожного із студентів групи:

Рівень II

6. Створіть web-сторінку з наступними атрибутами:

- назва файлу *task3a.html*;

- розміщення файлу ...*мої web-сторінки*;
- заголовок вікна *Робота із списками;*



- колір фону сторінки *сірий*;
- колір тексту *чорний*;
- на web-сторінці повинна розміститися наступна інформація:
  - два ненумерованих списки: А) 5 чоловічих імен, які починаються з перших 5 літер української абетки (Антон, Борис, Вадим і ін.); Б) 5 тропічних фруктів;
  - два нумерованих списки: А) 5 країн західної Європи; Б) основні складові частини комп'ютера;
  - кожен список назвіть за допомогою заголовка та розділіть списки лініями.

Лабораторна робота № 2

***Тема**: "Списки визначення, преформатований текст. Робота з графікою. Розміщення зображень на сторінці"*

Завдання 1: Організуйте списки визначень, запишіть преформатований текст

Рівень І

1. Створіть нову сторінку, на ній виведіть маркований список, де пунктом списку буде ім'я студента групи та нумерований список з 5-ма англійськими назвами кольорів.
2. Відкривши *Notepad++*, підготуйте наступну web-сторінку. Створіть список визначень:

HTML

це мова програмування, розроблена для опису web-сторінок, які можна створювати у програмі Notepad++.

Тег

це основне "слово" мови HTML, із сукупності тегів створюється web-сторінка.

3. Відкривши *Notepad++*, підготуйте наступну web-сторінку. Створіть список визначень:

% процент;
& амперсант;
** зірочка;*
/ слеш;
@ альфа комерційне.

4. Запишіть у HTML-сторінці такий преформатований текст:

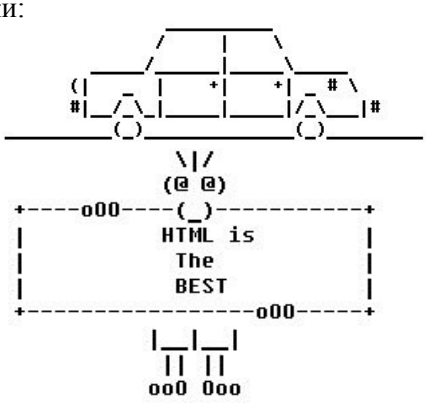
Я вмію створювати ненумеровані, нумеровані та інші списки

Рівень II

1. Складіть список визначень для наступних термінів: *браузер, гіперпосилання, параметр*.

2. Створіть компактний список визначень, де термін - це скорочена назва іноземної валюти, а визначення це її повна назва, *наприклад: дол.- долар.*

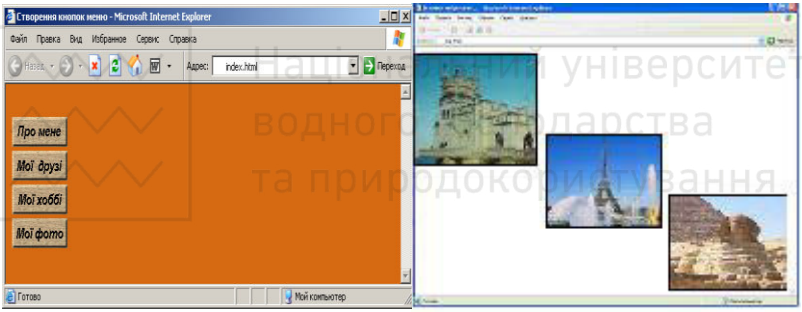
3. За допомогою преформатованого тексту створіть наступні малюнки:



Завдання 2. Розташуйте зображення на web-сторінці. Перегляньте можливості керування графічними елементами

Рівень I

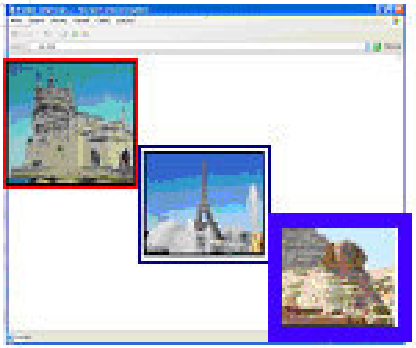
1. У власній папці *Lab2* створіть нову web-сторінку, збережіть її з назвою *task2a.html*.
2. Задайте заголовок вікна *Робота з графікою.*
3. Скопіюйте у власну папку *graphics* будь-який файл з малюнком, *наприклад fon.gif*.
4. В тегу `<body>` запишіть параметр `background=graphics/fon.gif`, збережіть та перегляньте сторінку.
5. Вставте малюнок *pic1.jpg* за допомогою тега `<img>`.
6. Аналогічно вставте 4 кнопки:

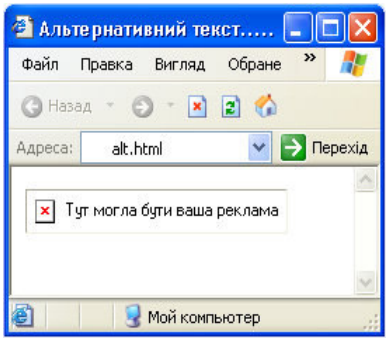


7. Вставте на сторінці *image.html* за вибором 3 малюнки.
8. Вирівняйте їх до лівого краю, до середини та до правого краю сторінки відповідно.

Рівень II

1. У першому малюнку зробіть так, щоб не було рамки навколо зображення.
2. Зробіть рамку навколо другого малюнка товщиною 4 пікселі.
3. Збільшіть розміри третього малюнка у півтора рази.
4. Вставте тег <img>, який посилається на неіснуючий файл. Створіть для нього альтернативний текст: “Тут могла бути Ваша реклама”.





Лабораторна робота № 3

***Тема**:"Шрифти, параметри шрифтів. Створення гіпертексту"*

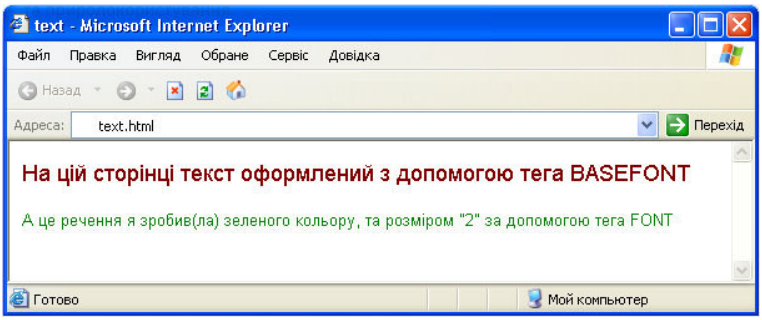
Завдання 1: Опрацюйте варіанти підбору параметрів шрифтів на web-сторінці

Рівень І

1. За допомогою тега BASEFONT задайте за замовчуванням текст такого вигляду: *шрифт-Arial, колір-Maroon, розмір-4*.

2. Створіть абзац тексту: "*На цій сторінці текст оформлений з допомогою тега BASEFONT*". Збережіть сторінку і перегляньте результат.

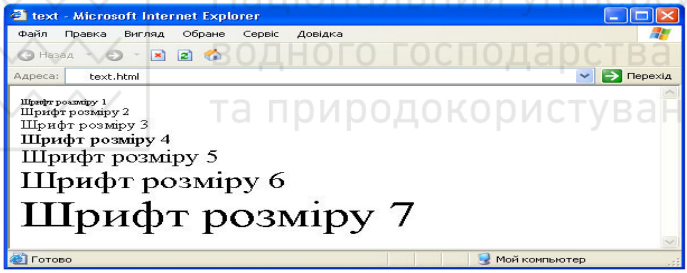
3. Використовуючи тег `<font>`, виведіть на екран наступний текст: "*А це речення я зробив(ла) зеленого кольору, та розміром „2" за допомогою тега* `font` ". Зробіть його зеленим та відповідного розміру. Після виконання перших трьох завдань сторінка повинна мати наступний вигляд:



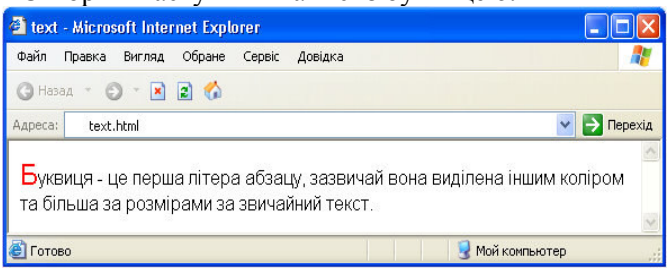
4. Створіть 6 написів за допомогою тега `<font>` різного розміру двома способами.

Наприклад:

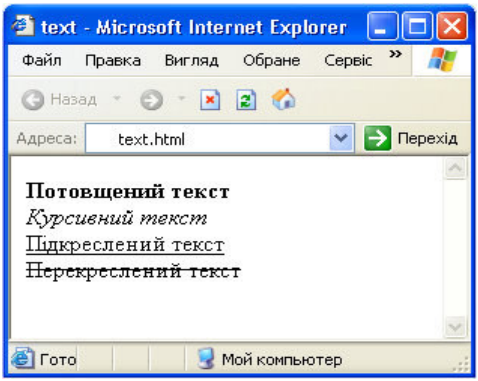
```
<font size=1> Шрифт розміру 1</font> і
<font size=-3> Шрифт розміру 1</font>
```



5. Створіть наступний напис із буквицею:



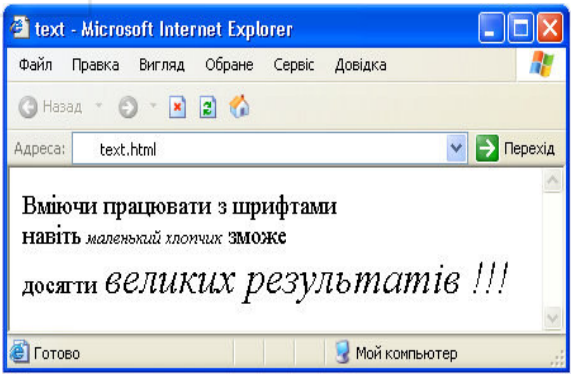
6. Зробіть чотири написи, використовуючи відповідні видозміни шрифтів:



7. Запишіть на сторінці формули води (H_2O), сірчаної кислоти (H_2SO_4), та теорему Піфагора ($AC^2=AB^2=BC^2$).

Наприклад: `<p>H<sub>2</sub>O</p>`, для зменшення індексу двійку можна взяти у середину тега `<small>`.

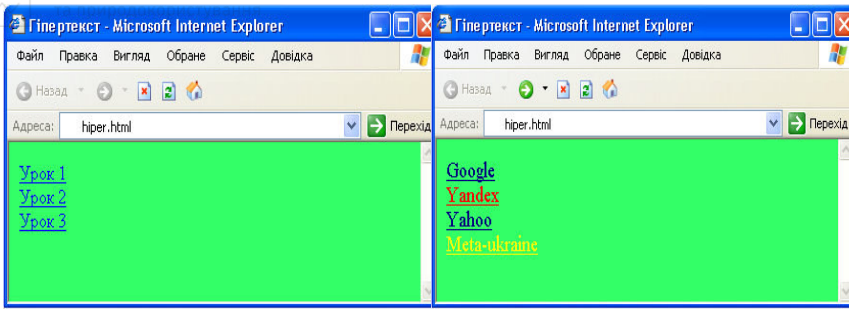
8. Змінюючи розміри тексту, створіть такий напис:



Завдання 2: Використайте гіперпосилання на web-сторінці

Рівень I

1. Створіть web-сторінку *lab3\task1.html*, де організуйте переходи до попередніх лабораторних робіт, протестуйте їх дію.

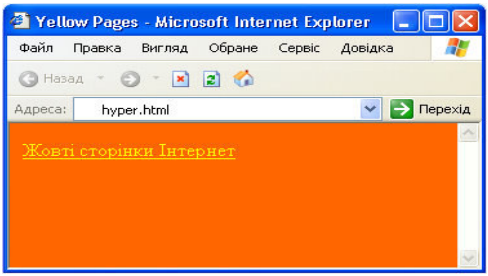


2. Під протестованими посиланнями проведіть горизонтальну лінію, зробіть напис "*Пошукові сервери*" та створіть такі посилання:

```
<a href=www.google.com> Google </a>
<a href=www.yandex.ru> Yandex </a>
<a href=www.yahoo.com> Yahoo </a>
<a  href=www.meta.ukraine.com>  Meta-ukraine
</a>
```

Зробіть звичайні посилання - червоного кольору, активні - оранжевого, а відвідані – синього кольору, протестуйте роботу збереженої сторінки.

3. Проведіть горизонтальну лінію. Створіть посилання "*Жовті сторінки Інтернет*", яке повинно переадресовувати до сайту *www.yellowpages.com* . За допомогою тега `<font>` зробіть гіпертекст *жовтим*. Нехай посилання відкривається у новому вікні:



4. Створіть посилання на адресу електронної пошти. Додайте до посилання тему повідомлення.

Лабораторна робота № 4

***Тема**: "Робота з таблицями. Форматування таблиць"*

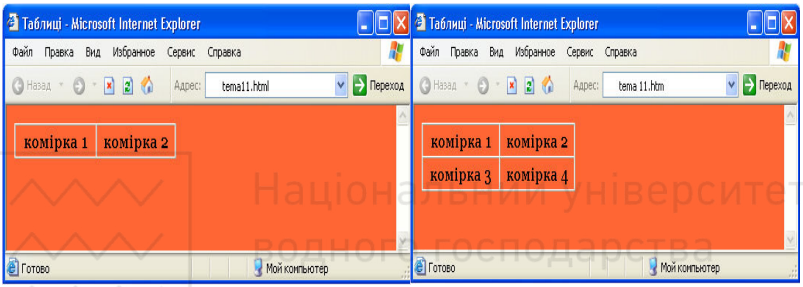
Завдання 1: Створіть таблиці на web-сторінках

Рівень I

1. Створіть найпростішу таблицю з одною коміркою:

```
<table border=1>
  <tr><td>комірка 1</td></tr>
</table>
```

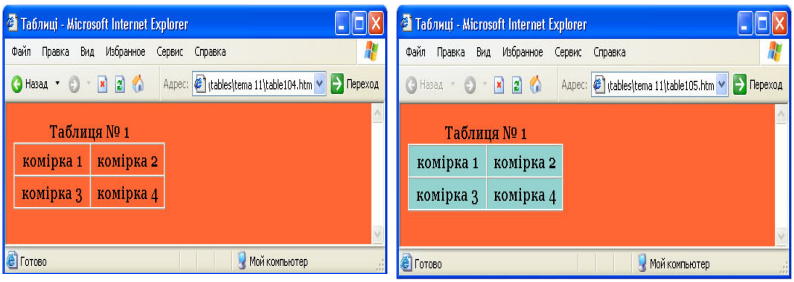
2. Додайте в таблиці ще одну комірку.



3. Допишіть другий рядок так, щоб утворилася таблиця розмірами 2 на 2.

4. За допомогою заголовка підпишіть дану таблицю:

```
<table border=1>
  <caption> Таблиця № 1 </caption>
```

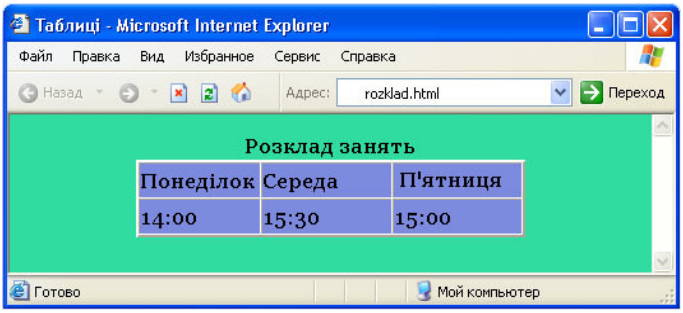


Рівень II

5. Задайте колір фону таблиці.

6. Зробіть рамку таблиці прозорою.

7. Створіть нову таблицю із заголовком *Розклад занять* та розмістіть його над таблицею. Таблиця повинна виглядати таким чином:



Завдання 2: Перегляньте варіанти об'єднування даних в таблицях

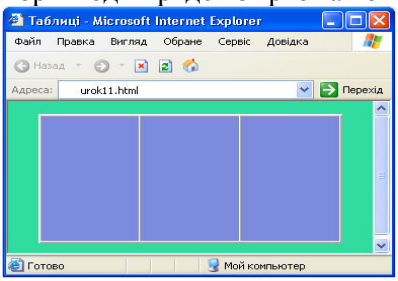
Рівень I

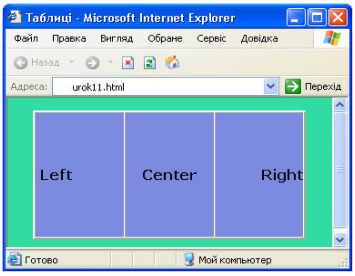
1. Створіть таблицю розмірами 200 на 200:

```
<table width=200 height=200>
...
</table>
```

Вирівняйте таблицю до середини сторінки, розмістивши її код в середині тега `<center> </center>`.

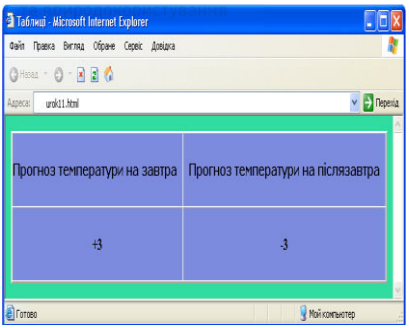
2. Створіть нову таблицю розмірами 300 на 150, у ній створіть один рядок з трьома комірками.





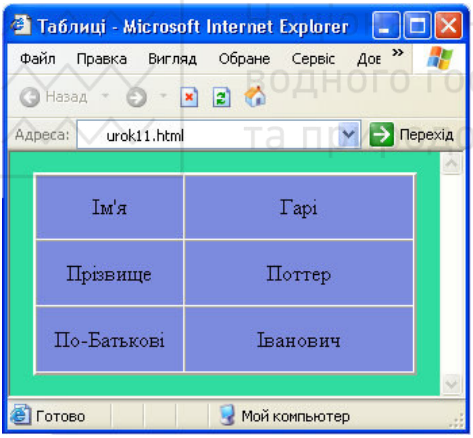
3. За допомогою параметра `align` розмістіть текст у комірках певним чином, *наприклад,* вирівняйте такст першої комірки до лівого краю: `<td align=left>...</td>`

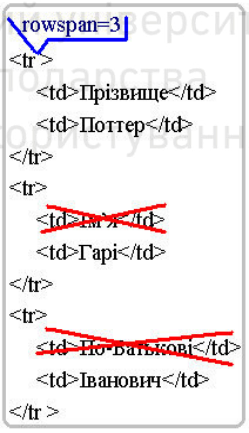
4. Створіть наступну таблицю розмірами 200 на 150:



Рівень II

5. Об'єднайте дві комірки другого рядка в одну (скористайтеся схемою).
6. Створіть наступну таблицю:



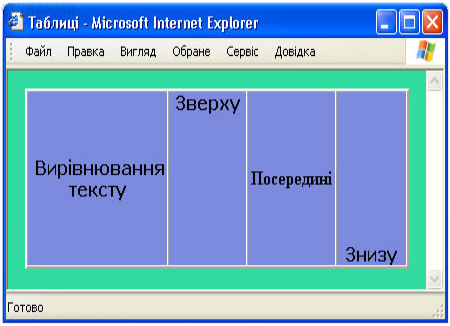


7. Об'єднайте 3 комірки першої колонки в одну комірку(скористайтеся схемою).

Завдання 3. Розгляньте варіанти форматування даних в таблицях

Рівень I

1. Використовуючи вертикальне вирівнювання, створіть наступну таблицю:



2. На новоствореній web-сторінці з фоном *ff6600* створіть нову таблицю з одною коміркою. Встановіть колір фону таблиці – *7c8bde*, відстань між комірками - 15 пікселів (скористайтеся параметром `cellspacing`), в комірку вставте малюнок *einstein.gif*.

3. В попередній таблиці зробіть внутрішній відступ (скористайтеся параметром `cellpadding`) - 15 пікселів і колір фону комірки *ff6600*.

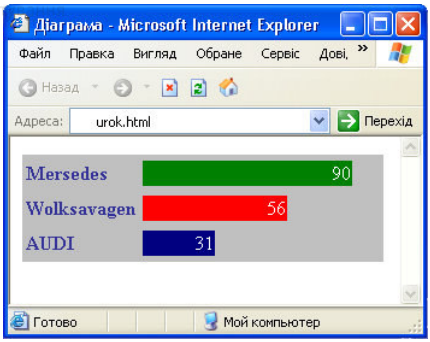
4. Створіть нову таблицю з трьома рядками та двома стовпчиками, ширина таблиці *300 пікселів*, фон таблиці – *silver*, товщина рамки –"*0*".

Рівень II

1. Створіть заголовок таблиці: "*Статистика продажу автосалону "German Auto*".

2. У першому стовпчику вставте назви трьох іномарок, *наприклад, Mersedes, Wolksvagen, Audi*.

3. У першій комірці другого стовпчика вставте внутрішню таблицю з двома комірками шириною і висотою *100%*, встановіть колір першої комірки *blue*, а ширину *90%*, другої *10%*. У створеній синій комірці зробіть напис "*90*" і вирівняйте *до правого краю*. Аналогічно створіть ще 2 комірки 2-го стовпчика. Ось який вигляд може мати результат:



Завдання 4. Застосуйте елементи оформлення таблиць

Рівень I

1. Створіть таблицю шириною *400 пікселів*, з кольором фону - *#ff0000*, товщиною рамки - *1*, внутрішнім відступом та відступом між комірками – *0*. Вирівняйте цю таблицю до середини сторінки та створіть заголовок таблиці "*Таблиця 1*".

2. В створеній таблиці додайте 4 комірки у першому рядку

3. Задайте кожній з 5-ти комірок почергово наступні кольори фону: #ffee00 #ffcc00 #ff9900 #ff6600 #ff4400

4. Скопіюйте код таблиці та вставте нижче.

5. У другій таблиці задайте коміркам наступні значення кольорів фону: #ff00ee #ff0099 #ff0066 #ff0099 #ff00ee

6. Змініть товщину рамки таблиці на *0*

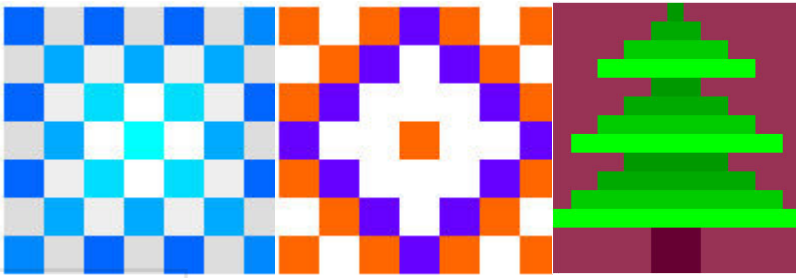
Рівень II

7. Створіть наступні таблиці, знаючи кольори окремих комірок, розміри комірки *20x20*:

А) #0066ff #00aaff #00ddff #00ffff #dddddd

#eeeeee #ffffff

Б) #ff6600 #6600ff #ffffff



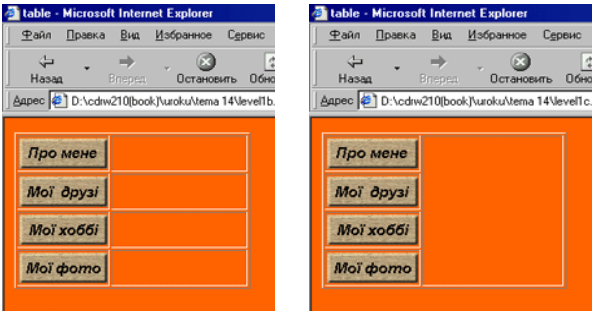
Завдання 5: Розгляньте використання таблиць для розмітки web–сторінок

Рівень І

1. Зайдіть у власну папку *Graphics*. У ній розташуйте наступні файли: *aboutme.jpg, myfoto.jpg, myhoby.jpg, myfriends.jpg*. Це створені Вами раніше кнопки.

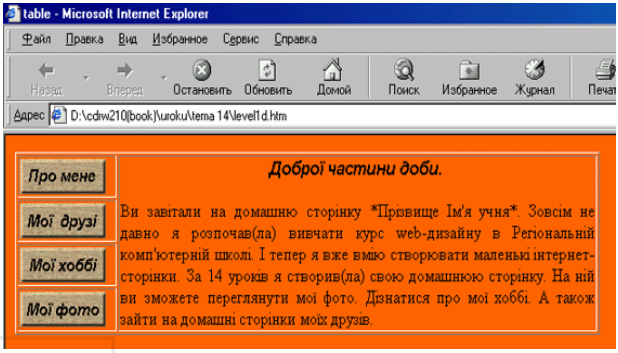
2. Створіть нову web-сторінку, призначте їй фон за смаком, створіть таблицю з 4-ма рядками та 2-ма стовпчиками.

3. В першому стовпчику розмістіть за порядком 4 зображення (згадані раніше файли):



4. Другий стовпчик об'єднайте в одну комірку.

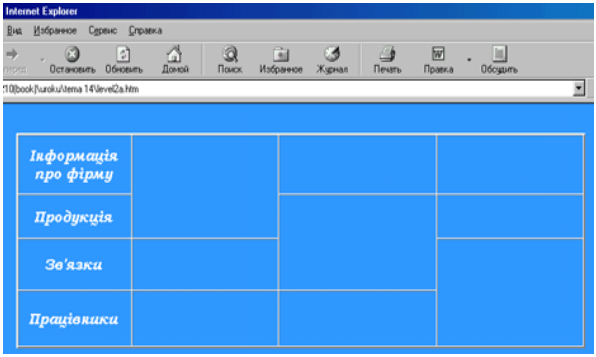
5. Скопіюйте уривок тексту та вставте в таблицю у щойно створену велику комірку, в тексті додайте власне *і м'я* та відредагуйте залежно від Вашої статі.



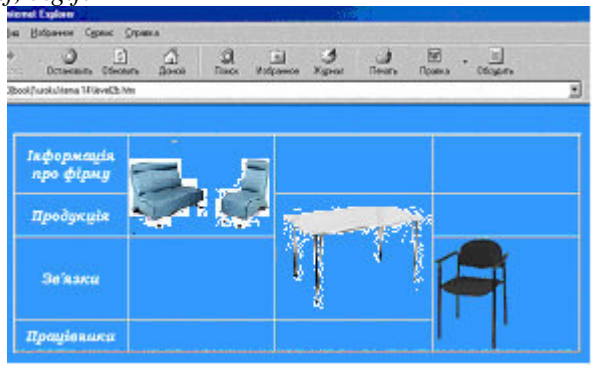
Рівень II

1. Створіть таблицю 4x4 комірки. У першому стовпчику у комірках зробіть такі написи: *Інформація про фірму, Продукція, Зв'язки, Працівники*. Скористайтеся шрифтом *Batang*, потовщеним та з нахилом.

2. Зробіть перетворення за допомогою параметра `rowspan`, щоб таблиця набула такого вигляду:



3. В новоутворених комірках вставте відповідно малюнки *1.gif, 2.gif, 3.gif*.



4. Зробіть товщину рамки *нульовою*.

Лабораторна робота № 5

***Тема**: "Створення, модифікація і використання форм"*

Завдання 1: Організуйте форми на web-сторінках

Рівень І

1. Створіть нову форму. Виділеним шрифтом (*наприклад*, потовщеним або підкресленим) наберіть текст запитання:

З якими операційними системами ви працювали?

2. Запишіть тег `<form>..</form>`, у ньому створіть перший варіант відповіді за допомогою елемента форми `checkbox`:

```
<input type="checkbox" name="system"
value="os1"> DOS
```

3. Аналогічно запишіть наступні варіанти відповіді змінюючи лише параметр `value`, де його значення будуть *os2, os3, os4* і далі:

```
<input type="checkbox" name="system" value =
    "os2"> Windows 3.1
<input type="checkbox" name="system" value =
    "os3"> Windows 95
```

```
<input type="checkbox" name="system" value =
    "os4"> Windows 98
<input type="checkbox" name="system" value =
    "os5"> Windows NT
<input type="checkbox" name="system" value =
    "os6"> Windows 2000
<input      type="checkbox"      name="system"
    value="os8"> Windows XP
```

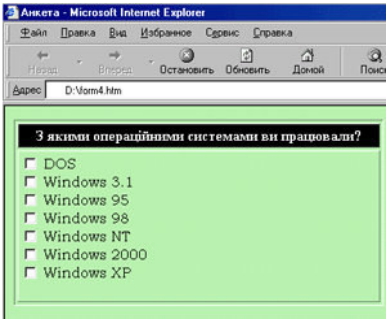
4. Перегляньте сторінку за допомогою браузера.

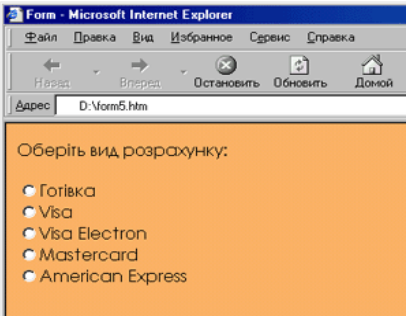
5. Форма працює, але не досить гарно оформлена. Розділіть варіанти відповідей тегом `<br>`:

6. Додайте сторінці фон. Наведемо *приклади* оформлення окремих сторінок різними кольрами фону. Спробуйте прикрасити власну форму, орієнтуючись на один із зразків. Створіть таблицю шириною *50%* з двома рядками, в кожному з них - по одній комірці.

7. У верхній комірці розмістіть текст запитання, а у нижній - власне форму з варіантами відповіді.

8. Фон верхньої комірки зробіть чорним, а колір тексту - білим. В тегу `<table>` допишіть параметри `bordercolorlight = "#738278" bordercolordark = "#ffffff".` Результат буде мати вигляд:





Рівень II

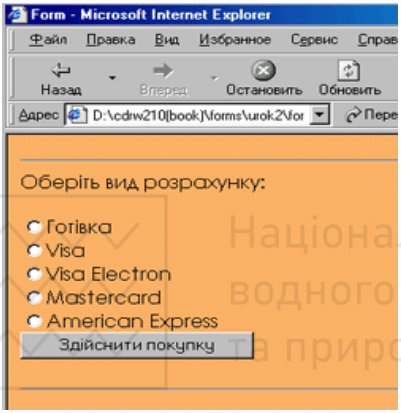
1. Створіть нову форму, яка повинна бути сторінкою інтернет-магазину. Надрукуйте заголовок "*Оберіть вид*

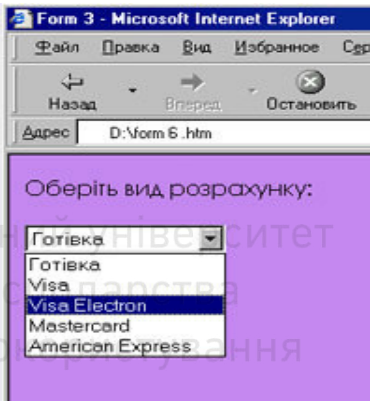
розрахунку". У новому рядку створіть форму, де за допомогою об'єкта `radio` створіть варіант відповіді:

```
<input type="radio" name="moneytype" value =
   "cash"> Готівка
```

2. Аналогічно створіть решту об'єктів форми, розділяючи їх тегом `<br>`.

3. Створіть кнопку з написом "*Здійснити покупку*". Додайте згори та знизу сторінки тег `<hr>`.





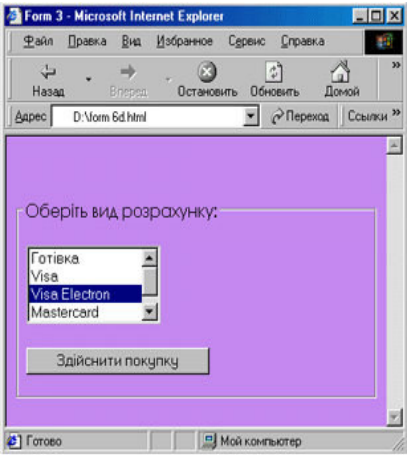
Рівень III

1. Перероблюйте форму 2 так, щоб замість об'єкта `radio` використовувався об'єкт `select`:

```
<form action=money.php>
  <select >
  <option selected value="cash"> готівка
</option>
  <option value="visa"> visa </option>
  ...і так далі...
</form>
```

2. Допишіть у тегу `<select>` параметр `multiple`, збережіть та перегляньте зміни на сторінці. За допомогою тега `<fieldset>` логічно згрупуйте всі елементи форми, текст заголовка помістіть в тег `<legend>`:

```
 <fieldset>
 <legend >Оберіть
вид розрахунку:
</legend>
 <br> <select
multiple>
 <option selected
value="cash">
готівка </option>
 <option
value="visa"> visa
</option>
 …і так далі…
 </fieldset>
```

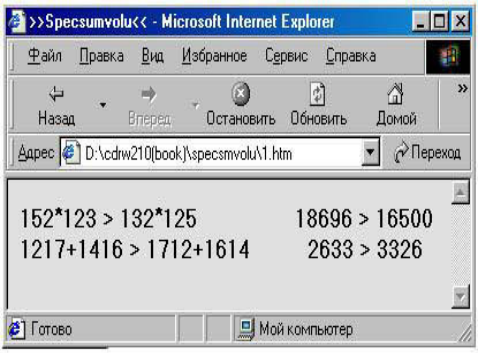


Лабораторна робота № 6

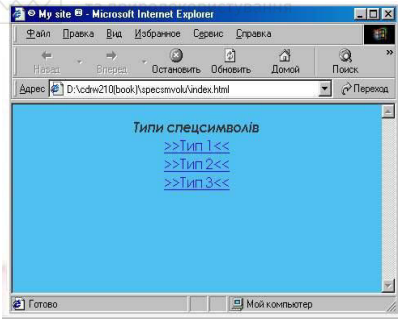
***Тема**: "Використання спеціальних символів"*

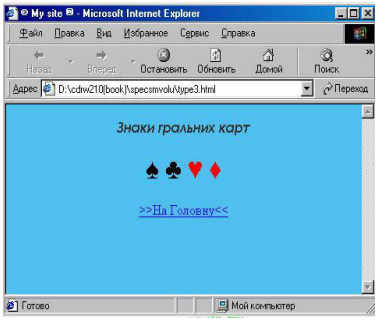
Завдання 1: Ознайомтесь із спеціальними символами

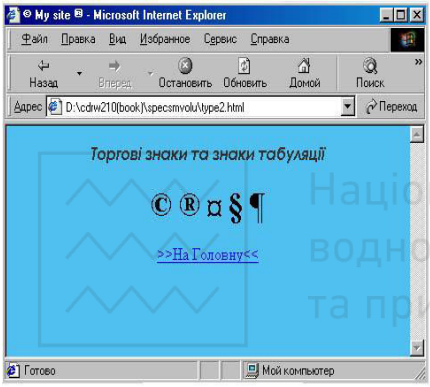
Рівень I

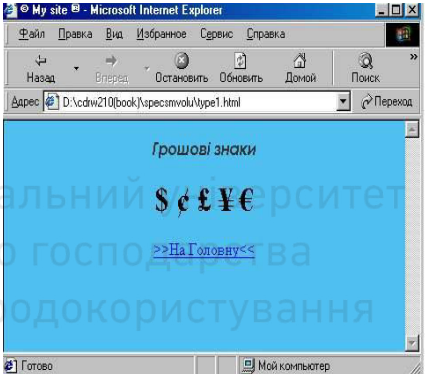


Створіть сторінку *lab6\task1.html* з посиланнями на інші:









Лабораторна робота № 7

***Тема*:** "*Основи об'єктно-орієнтованого програмування (ООП) в JavaScript . Методи об'єкта window. Обробка подій*"

***Мета*:** *Опрацювати методи alert, confirm, close, prompt. Розглянути варіанти використання в сценаріях подій onBlur, onClick, onFocus*

Приклад 1. Написати сценарій реалізації наступного діалогу з користувачем: перебуваючи на сторінці, ввести у текстове поле свій вік. При втраті фокусу текстовим полем, програма проаналізує введене значення та виведе на екран результат перевірки віку (чи вік перевищує 18 років?). У скрипті використати метод **`alert`**.

<html>

```
<head>
<script >
<!--
function Age()
{
    if (document.form1.a.value <= 18)
        { alert ("Вибачте! Вік повинен бути більше 18 років!"); }
    if (document.form1.a.value > 18)
        { alert ("Вітаємо! Вам більше 18 років."); }
}
<!---->
</script>
</head>
<body>
<form name="form1">
Вік: <input type="text" name="a" size=5 onBlur="Age()">
</form>
</body>
</html>
```


Приклад 2. Розробити сценарій для обчислення площі квадрата при введенні значення довжини його сторони. Програма повинна спрацювати при отриманні фокусу текстовим полем.

```
<html>
<head>
<script >
<!--//
function loan(obj)
{
    var a=obj.a1.value;
    var result=a*a;
    obj.sr.value=result;
}
//-->
</script>
</head>
```

```
<body>
<form name="form1">
<font size=3>
Для сторони квадрата: <input type="text" size=6 name="a1"><br>
Його площа буде дорівнювати: <input type="text" size=6
name="sr" onFocus="loan(form1)">
</form>
</body>
</html>
```

Приклад 3. Реалізувати програму діалогу з користувачем. Програма повинна запитати, чи бажає клієнт закрити вікно. У скрипті використати методи **`confirm`** і **`close`**.

```
<html>
<head>
<script >
var newWin;
function wikno()
{
  var newWin = window.open("","about:blank","width=200,
                                                height=200");
  if (confirm("Бажаєте закрити вікно?")){
    newWin.close();}
}
</script>
</head>
<body onLoad="wikno()" >
</body>
</html>
```

Приклад 4. Скласти програму реалізації введення реєстраційних даних користувачем. Використати метод **`prompt`**.

```
<html>
<head>
```

```
<script >
<!--//
function registration(obj)
{
    var a=window.prompt("Введіть Ваше ім’я"," ")
    obj.regname.value=a;
}
//-->
</script>
</head>
<body>
<form name="form1">
<input type="button" value="Реєстрація" onClick= "registration
(form1)">
<input type=text name=regname size=50>
</form>
</body>
</html>
```


Завдання:

1. Створити сценарій, у якому є функція, що запускається методом **`alert`** з текстом “Було клацання!”, якщо користувач клацне кнопкою миші на сторінці.
2. Написати сценарій, у якому при зміні тексту в текстовому полі з’явиться нове вікно з текстом “Текст було змінено!”.
3. Підготувати сценарій, у якому функція обчислює середню зарплату за 3 місяці. Подія настає при втраті фокуса.
4. Розробити програму, яка визначає зарплату працівнику, що працює на умовах погодинної оплати. Подія настає при отриманні фокуса. Введення даних здійснити за допомогою вікна **`prompt`**. Необхідно ввести кількість годин і оплату за годину.
5. Реалізувати програму для обчислення коренів квадратного рівняння. Для виведення даних використати метод **`alert`**, а для обробки подій - **`onBlur`**.
6. Написати програму “Обмін валют”.

Лабораторна робота № 8

***Тема*:** "*Робота з графічними зображеннями"*

***Мета*:** *Отримати навички управління властивостями графічних зображень засобами JavaScript . Навчитися готувати діаграми на основі числових даних*

Приклад 1. Підготувати програму, яка реалізуватиме діалог з користувачем і виведення зображення на екран із дотриманням уведених параметрів. На сторінці дозволяється змінювати значення наступних параметрів зображення: висота, ширина, товщина границі, підпис.

```
<html>
<head>
<title> Дослідження властивостей зображень </title>
<script >
function chipct(obj)
{
   var w=obj.wd.value;
   var h=obj.hg.value;
   if (w!=0) document.mypict.width=w;
   if (h!=0) document.mypict.height=h;
   document.mypict.border=obj.br.value;
   document.mypict.alt=obj.al.value;
}
</script>
</head>
<body bgcolor="F8F8FF">
<center>
<h3> Вбудовувані зображення </h3>
<img src="ved.jpg" name=mypict>
<form name="form_name" Для зміни параметрів зображення введіть значення
одного або двох параметрів і натисніть кнопку <b> Перегляд </b><br>
Ширина: <input type="text" name="wd" size=8><br>
```

Висота: <input type="text" name="hg" size=8>
Для задання рамки введіть число, яке визначає товщину рамки в пікселях,
і натисніть кнопку Перегляд
Товщина рамки: <input type="text" name="br" size=8 value=0>
Альтернативний текст: <input type="text" name="al" size=40 value=0>
<input type="button" value="Перегляд" onClick="chipct(form_name)">
<input type="reset" value="Очистити" onClick="close">
</form>

</body>
</html>

Приклад 2. Розробити сценарій, який реалізує обмін рисунків місцями у документі.


```
<html>
<head>
<title> Перестановка зображень </title>
<script >
function chan(obj)
{
var r1=Number(obj.a1.value);
var r2=Number(obj.a2.value);
if((r1<1)||(r1>4)||(r2<1)||(r2>4))
alert("Неправильно задані номери рисунків!");
var z=document.images[r1-1].src;
document.images[r1-1].src=document.images[r2-1].src;
document.images[r2-1].src=z;
}
</script>
</head>
<body bgcolor="F8F8FF">
<center>
```

```
<h4> Галерея рисунків </h4><br>
<img src="le.jpg" width="90" name="pic1">
<img src="lg.jpg" width="90" name="pic2">
<img src="sg.jpg" width="90" name="pic3">
<img src="so.jpg" width="90" name="pic4">
<form name="form_name">Рисунки з номерами <br>
<input type="text" name="a1" size=1>
<input type="text" name="a2" size=1>
<input type="button" value="Поміняти місцями" size=1
onClick="chan(form_name)">
<br><br><input type="reset" value="Очистити форму">
</form>
</center>
</body>
</html>
```

Приклад 3. Реалізувати сценарій виведення вертикального графічного меню. При наведенні курсора миші на пункт меню повинно змінюватися зображення, яка відповідає виокремленому пункту меню. Кожному пункту меню поставити у відповідність два зображення: перше зображення - коли пункт меню не обрано, друге - при обраному пункті меню. Графічні зображення, які відповідають ситуації, коли пункти меню не обрані, зберігати у файлах з іменами: file1.jpg, file2.jpg, file3.jpg, file4.jpg, file5.jpg. Відповідні їм графічні зображення обраних пунктів - obr1.jpg, obr2.jpg, obr3.jpg, obr4.jpg, obr5.jpg.

```
<html>
<head>
<title> Просте вертикальне меню </title>
<script >
function im(n, action)
{
if (action)
    document.images[n-1].src="obr"+n+".jpg"
    else
    document.images[n-1].src="file"+n+".jpg"
}
```

```
</script>
</head>
<body bgcolor="white" link="#0000EE" vlink="#551A8B"
alink="#FF0000">
<h2><font color="#0000FF" > Зміст </h2>
<a href="ch1.htm" onMouseOver="im(1,1)"
onMouseOut="im(1,0)">
<img src="ch1.htm" alt="Форматування тексту" border="0"
width="180" height="35"></a><br>
<a href="ch2.htm" onMouseOver="im(2,1)"
onMouseOut="im(2,0)">
<img src="" alt="Створення списків" border="0" width="180"
height="35"></a><br>
<a href="ch3.htm" onMouseOver="im(3,1)"
onMouseOut="im(3,0)">
<img src="" alt="Побудова таблиць" border="0" width="180"
height="35"></a><br>
<a href="ch4.htm" onMouseOver="im(4,1)"
onMouseOut="im(4,0)">
<img src="" alt="Використання графіків" border="0" width="180"
height="35"></a><br>
<a href="ch5.htm" onMouseOver="im(5,1)"
onMouseOut="im(5,0)">
<img src="" alt="Імпорт/ Експорт" border="0" width="180"
height="35"></a><br>
</font>
</body>
</html>
```

Приклад 4. Написати сценарій обробки анкети викладача. В анкеті передбачити наступні поля для введення даних: кількість годин, запланованих на читання лекцій; кількість годин на проведення практичних занять, а також кількість студентів. Якщо з предмету читаються лекції, додатково автоматично планувати наступну частину навантаження: 10% від кількості лекційних годин відводити на консультації, для прийому екзамену планувати по 30 хвилин на 1-го студента. Якщо з

предмету проводяться практичні заняття, слід автоматично передбачити наявність: контрольних робіт з розрахунку 15 хвилин на 1-го студента, заліку з розрахунку 20 хвилин на 1-го студента.

```
<html>
<head>
<title> Навантаження викладача </title>
<script >
function graph(obj)
{
var k1=0;
var l=obj.lec.value
var p=obj.pract.value
var s=obj.stud.value
var tableHeight=150
var knTime=0 //консультації
var eTime=0  //екзамен
var krTime=0 //контрольна
var zTime=0  //залік
d=document
if (s=="") alert("Вкажіть кількість студентів")
if (l!="" && l!="0")
knTime=Math.round(l*10/100); //консультації
eTime=Math.round(s*30/60)    //екзамен
if (p!="" && p!="0")
krTime=Math.round(s*15/60);  //контрольна
zTime=Math.round(s*20/60)    //залік
sTime=knTime+eTime+zTime+krTime   // разом
k1=tableHeight/sTime
d.images[0].height=Math.round(krTime*k1)
d.images[0].alt=krTime+" год."
d.images[1].height=Math.round(zTime*k1)
d.images[1].alt=zTime+" год."
d.images[2].height=Math.round(knTime*k1)
d.images[2].alt=knTime+" год."
d.images[3].height=Math.round(eTime*k1)
```

```
d.images[3].alt=eTime+" год."
d.images[4].height=Math.round(sTime*k1)
d.images[4].alt=sTime+" год."
}
function CheckValHours(hourse)
{ if (hourse<0) alert("Неможливе значення") }
function CheckValNum(num)
{ if (num<0)     alert("Неможливе значення ") }
</script>
</head>
<body bgcolor="F8F8FF">
<center>
<h4> Навантаження викладача </h4><br>
<form name="data">
<table bgcolor=#cccccc width=300 cellspacing=2 cellpadding=2
border=1>
<tr><td> Лекції </td><td><input type="text" name="lec" size=5
value="" onchange="CheckValHours(this.value)"></td></tr>
<tr><td> Практичні заняття </td><td><input type="text"
name="pract"                size=5                value=""
onchange="CheckValHours(this.value)"></td></tr>
<tr><td> Кількість студентів</td><td><input type="text"
name="stud"                size=5                value=""
onchange="CheckValNum(this.value)"></td></tr>
<tr align=center><td colspan=2 bgcolor=#cccccc>
<input type="button" value=" Обчислити" onclick=graph(data)>
</td></tr>
</table>
</form>
<h4> Діаграма </h4><br>
<table width=212 height=190 cellspacing=1 cellpadding=0
border=0>
<tr valign=bottom width=100 align=center>
<td bgcolor=#aaaaaa height=150>
<img src="" width=75 height=0 border=0 ></td>
<td bgcolor=#aaaaaa height=150>
<img src="" width=75 height=0 border=0 ></td>
```

```
<td bgcolor=#aaaaaa height=150>
<img src="" width=75 height=0 border=0 ></td>
<td bgcolor=#aaaaaa height=150>
<img src="" width=75 height=0 border=0 ></td>
<td bgcolor=#aaaaaa height=150>
<img src="" width=75 height=0 border=0 ></td>
</tr>
<tr valign=top width=100 align=center>
<td height=40><small> Контрольні роботи </small></td>
<td height=40><small> Залік </small></td>
<td height=40><small> Консультації </small></td>
<td height=40><small> Екзамен </small></td>
<td height=40><small> Всього </small></td>
</tr></table>
</body>
</html>
```


Завдання:

1. В **прикладі 1** організувати введення даних за допомогою вікна **prompt**.
2. В **прикладі 2** добавити два графічних об’єкти.
3. В **прикладі 3** зробити горизонтальне меню з 6-ти графічних об’єктів з відповідними посиланнями (посилання розмістити в таблиці).
4. Підготувати скрипт, який виконуватиме наступне. Для кожного з N (наперед не відоме!) студентів вводяться такі дані: прізвище і дві оцінки за контрольні роботи. Студентів потрібно класифікувати за категоріями. Категорія “відмінники” складатиметься із студентів, у яких обидві контрольні виконані на оцінку 5. До категорії “хорошисти” включати студентів, у яких кожна з оцінок – не нижче 4, але при цьому вони не відмінники. Категорію “встигаючі” складатимуть студенти, у яких хоча б одна контрольна виконана на 3 і немає жодної двійки. Нарешті, “невстигаючі” - ті студенти, які мають оцінку 2 хоча б за одну контрольну. Побудувати діаграму розподілу студентів за категоріями.

5. Написати програму, призначену для обробки анкет співробітників. В анкеті для кожного з N (наперед не відоме!) співробітників вводити наступні дані: прізвище, рік прийому на роботу. Обчислити стаж роботи кожного співробітника. Визначити максимальне число співробітників з однаковим стажем. Побудувати діаграму, яка відображає стаж роботи співробітників.

Лабораторна робота № 9

***Тема*:** "*Оператори JavaScript . Умовний оператор. Циклічні конструкції. Переривання циклів. Об'єкт radio"*

Мета: *Навчитися розробляти сценарії JavaScript з використанням управляючих операторів (умовного оператора, оператора циклу), ознайомитися з варіантами застосування об'єкта radio*

Приклад 1. Підготувати програму обчислення площі фігури за однією з формул у залежності від обраного типу фігури (квадрат, круг чи рівнобедрений прямокутний трикутник) з використанням заданого початкового значення сторони, радіуса або катета.

```
<html>
<head>
<title>!!!</title>
<script >
<!--
function ds()
{
var a1=form1.a.value;
if(document.form1.elements[1].checked)
    k=(document.form1.elements[1].value);
if(document.form1.elements[2].checked)
    k=(document.form1.elements[2].value);
```

```
if(document.form1.elements[3].checked)
    k=(document.form1.elements[3].value);
if(a1!="") form1.res.value=k*Math.pow(a1,2);
else alert("Введіть значення!");
}
//-->
</script>
</head>
<body>
<form name="form1">
<h3>Введіть значення (сторони квадрата, радіуса, катета): </h3>
<input type="text" name="a" size=10>
<hr>
<input type="radio" name="fv" value="1" onClick="ds()"> квадрат
    <br>
<input type="radio" name="fv" value="3.14" onClick="ds()"> круг
    <br>
<input type="radio" name="fv" value="0.5" onClick="ds()">
    прямокутний трикутник <br>
<input type="reset" name="reset" value="Очистити"><hr>
Площа: <input type="text" name="res" size=10>
</form>
</body>
</html>
```

Приклад 2. Реалізувати найпростішу тестову програму. Передбачити виведення 10 запитань. На кожне із запитань запропонувати 3 можливих варіанти відповідей. Оцінити кожен із допустимих варіантів відповідей балами: 0, 1 або 2. При оцінюванні загального результату тестування використати наступну шкалу: якщо кількість балів менше 6 – результат №1; 6 - 12 – результат №2; більша 12 - результат №3.

```
<html>
<head>
<title> Тест </title>
<script >
```

```
<!--
function see()
{
var k=0
for (var i=0;i<30;i++)
{
   if(document.test.elements[i].checked)
   {
   k=k+Number(document.test.elements[i].value)
   }
}
if(k<=5) document.test.rez.value="Ви ніколи не вдастеся до
   ризикованих авантюр з грошима, навіть якщо вони обіцяють
   прибуток. Швидше за все, Ви 'консервативний вкладник',
   який тихо зберігає свої заощадження. Спробуйте іноді
   ризикнути і обрати форму активних дій з капіталом.
   Ймовірно, Ви від цього тільки виграєте "

if (k>5&&k<=12) document.test.rez.value="Ви непогано
   орієнтуєтеся на ринку капіталів, але часом Вам не завадить
   трохи схаменутися і, притримуючи любов до ризику,
   звернутися до більш консервативних способів вкладення
   капіталу. Зрештою, все чудово "

if (k>12) document.test.rez.value="Природа подарувала Вам
   чудовий нюх на гроші! Але будьте обережним і час від часу
   намагайтеся зупинитися і задати собі питання: чи не занадто
   Вас заносить? Навіть найздібніша людина не застрахована
   від помилок. Якщо бути надто самовпевненим, можна
   пропустити момент небезпеки "
}
//-->
</script>
</head>
<body >
<font color=brown>
<font size=4>
```

```
<i>
<h1><center> Чи вмієте Ви поводитися з грошима?</center>
   </h1>
<form name="test">
1.<u>Чи можна збагатитися чесною працею? </u><br>
<input type="radio" name="a" value=2> Так <br>
<input type="radio" name="a" value=1> Не знаю <br>
<input type="radio" name="a" value=0> Ні <br><br>
2. <u> Ви регулярно читаєте матеріали на економічні теми?
   </u><br>
<input type="radio" name="b" value=2> Так <br>
<input type="radio" name="b" value=1> Часом <br>
<input type="radio" name="b" value=0> Ні <br><br>
3. <u>Чи готові Ви піти на великий ризик заради отримання
   прибутку? </u><br>
<input type="radio" name="c" value=1> Так <br>
<input type="radio" name="c" value=2> Можливо <br>
<input type="radio" name="c" value=0> Ні <br><br>
4. <u> Ви згодні з висловлюванням "не в грошах щастя"?
   </u><br>
<input type="radio" name="d" value=0> Так <br>
<input type="radio" name="d" value=1> Не знаю <br>
<input type="radio" name="d" value=2> Ні <br><br>
5. <u>Фінансовим магнатам було вигідним об'єднання
   Німеччини? </u><br>
<input type="radio" name="e" value=1> Так <br>
<input type="radio" name="e" value=2> Не відомо <br>
<input type="radio" name="e" value=0> Ні <br><br>
6. <u>Чи можна позичити у Вас гроші після нетривалої розмови
   за чашкою кави? </u><br>
<input type="radio" name="f" value=0> Так <br>
<input type="radio" name="f" value=1> Навряд чи <br>
<input type="radio" name="f" value=2> Ні <br><br>
7. <u> У Вас залишаються гроші перед зарплатою? </u><br>
<input type="radio" name="g" value=2> Так <br>
<input type="radio" name="g" value=1> Рідко <br>
<input type="radio" name="g" value=0> Ні <br><br>
```

```
8. <u>Чи можна нажитися на перепродажі товару? </u><br>
<input type="radio" name="h" value=0> Так <br>
<input type="radio" name="h" value=1> Не знаю <br>
<input type="radio" name="h" value=2> Ні <br><br>
9.<u> Пристойно давати чайові дріб'язком? </u><br>
<input type="radio" name="i" value=0> Так <br>
<input type="radio" name="i" value=1> Можливо <br>
<input type="radio" name="i" value=2> Ні <br><br>
10. <u>Чи псують гроші характер людини? </u><br>
<input type="radio" name="j" value=2> Так <br>
<input type="radio" name="j" value=1> Не завжди <br>
<input type="radio" name="j" value=0> Ні <br><br>

<textarea name="rez" cols=35 rows=7></textarea><br>
<input type="reset" name="Очистити" value="Очистити">
<input type="button" value="Результат" OnClick="see(test)">

</form>
</font>
</body>
</html>
```


Завдання:

1. Написати програму обчислення довжини кола, площі круга, об'єму кулі за заданим радіусом. Радіус вводиться користувачем. У залежності від вибраної радіо-кнопки, виконується обчислення довжини кола (L=2*Pi*r), площі круга ($S=Pi*r^2$) або об'єму кулі ($V=(4/3)*Pi* r^3$) і відповідно виводиться результат у текстовому вікні.
2. Написати програму для обчислення об'ємів тіл (конус, циліндр). Радіус і висота вводяться користувачем. У залежності від вибраної радіо-кнопки, виконується розрахунок об'єму конуса ($V= (1/3)*Pi * r^2 * h$) або циліндра ($V= Pi * r^2 * h$) і відповідно виводиться результат у текстовому вікні.

3. Написати програму, яка дозволяє протестувати студентів з предмету "Web-орієнтоване програмування". Передбачити виведення 10 запитань. На кожне із запитань запропонувати 3 можливих варіанти відповіді. Правильним може бути тільки один із можливих варіантів відповіді. За кожну правильну відповідь студент отримує 1 бал. Вивести кількість набраних балів і оцінку. При оцінюванні результату тесту використати наступну шкалу: якщо кількість балів більша 13 – відмінно; 10 - 13 – добре; 7 - 9 - задовільно; менше 7 - незадовільно.

Лабораторна робота № 10

***Тема*:** *"Оператори JavaScript . Умовний оператор. Циклічні конструкції. Переривання циклів. Об'єкт checkbox"*

Мета: *Навчитися розробляти сценарії JavaScript з використанням операторів управління (умовного оператора, оператора циклу), ознайомитися з варіантами застосування об'єкта checkbox*

Приклад 1. Написати сценарій **"Анкета курортника"**. Нехай потрібно провести опитування відпочивальників щодо того, які компоненти пансіонату приваблювали їх як клієнтів найбільше? Справа від тексту запитання слід розмістити поле **`textarea`** для виведення варіантів назв компонент і встановлених біля них прапорців. Обирати можна кілька елементів із списку.

- Харчування
- Котеджі
- Пляж
- Дискотека
- Морські атракціони
- Якість обслуговування

```
<html>
<head>
<title> Анкета курортника </title>
<script >
<!--//
var s="Вас приваблює: \n" ;
```

```
function set (vch)
{
s=s+vch+"\n";
document.form1.area.value=s;
}
//-->
</script>
</head>
<body bgcolor="f8f8ff">
<center>
<h3 align="center"> Анкета курортника </h3>
<form name="form0">
<h4> Введіть назву улюбленого пансіонату (будинку
   відпочинку, санаторію...)</h4>
<input type="text" name="n1" size=45> <br>
</form>
<form name="form1"
<h4> Що Вам найбільше імпонує у цьому пансіонаті? </h4>
<table border=5 align=center>
<td><img  src="harch.jpg"  align="center"  width=200  height=160
   ></></td>
<td>
<input type="checkbox" name="m1" value="Харчування "
   onClick="set(form1.elements[0].value)"> Харчування <br>
<input type="checkbox" name="m2" value="Котеджі "
   onClick="set(form1.elements[1].value)"> Котеджі <br>
<input type="checkbox" name="m3" value="Пляж "
   onClick="set(form1.elements[2].value)">Пляж <br>
<input type="checkbox" name="m4" value="Дискотека "
   onClick="set(form1.elements[3].value)">Дискотека <br>
<input type="checkbox" name="m5" value="Морські атракціони "
   onClick="set(form1.elements[4].value)"> Морські атракціони
   <br>
<input type="checkbox" name="m6" value="Якість обслу-
   говування "
   onClick="set(form1.elements[5].value)"> Якість обслуго-
   вування
```

```
</td></table><br>
<textarea name="area" cols=35 rows=7></textarea><br>
<input type="reset" value="Відміна" onClick="s='Вам
подобається: \n' ">
</form>
</body>
</html>
```

Завдання:

1. Напишіть сценарій **“Розділи студентської газети”**.
   Нехай потрібно провести опитування студентів щодо того, які розділи повинні бути у студентській газеті? Справа від тексту запитання розмістимо поле **`textarea`** для виведення варіантів назв розділів і встановлених біля них прапорців. Обирати можна кілька елементів.
   - Новини
   - Гості
   - Нагороди
   - Освіта
   - Дозвілля
   - Розваги
   - Кар’єра

2. Напишіть сценарій для обробки **“Анкети перекладача”**. Прапорці у цій анкеті повинні бути об’єднані в групу. В анкеті перекладач повинен вказати мови, якими він володіє. В результаті сценарій сформує суму винагороди (за знання кожної мови призначається певна сума).
3. Напишіть сценарій **“Обробка анкети слухача курсів”**. При заповненні анкети користувачем, він може обрати курс, його тривалість, мову викладання і форму звітності. У залежності від цих параметрів встановити вартість навчання.

Назви курсів	Тривалість	Мова	Звіт	Вартість
Інформатика	О 36	□ Українська	О Екзамен	□ 250
	О 64	□ Англійська	О Залік	□ 500
	О 128			□ 1000
Бази даних	О 36	□ Українська	О Екзамен	□ 400
	О 64	□ Англійська	О Залік	□ 800
	О 128			□ 1600
Ресурси Інтернет	О 36	□ Українська	О Екзамен	□ 300
	О 64	□ Англійська	О Залік	□ 600
	О 128			□ 1200
Аналіз алгоритмів	О 36	□ Українська	О Екзамен	□ 600
	О 64	□ Англійська	О Залік	□ 1200
	О 128			□ 2400

Тест з розділу “Web-орієнтоване програмування”

1. В якому році була розроблена мова сценаріїв JavaScript?
- 1998
- 1995
- 2001
2. Якою фірмою була розроблена мова сценаріїв JavaScript?
- Microsoft
- Netscape
- Sun Microsystems
3. Що повертає функція typeof, якщо змінна не існує або не визначена?
- Undefined
- Boolean
- String
4. За допомогою якого оператора можна реалізувати повторення виконання групи операторів до тих пір, поки не буде виконано певну умову?
- for
- break
- while
5. Який об’єкт містить математичні константи і функції?
- math
- mathematics
- matrix
6. Який об’єкт використовується для роботи з графічними зображеннями?
- img
- image
- images
7. За допомогою якого методу можна створити вікно попередження?
- alert()
- prompt()
- confirm()

8. Яким атрибутом обробки події (зв'язаним з даним полем) задається дія, яка виконується в тому випадку, коли користувач обере текст, що міститься в даному полі?
- onFocus
- onChange
- onSelect

9. Яка властивість об'єкта image задає порожній простір згори і знизу?
- vspace
- hspace
- wspace

10. Який обробник події використовується в об'єктах checkbox і radio button?
- onLoad
- onUnload
- onClick

11. Який варіант є правильно організованим гіперпосиланням?
- IMG HREF="simple.gif"
- A HREF="simple.html"
- A SRC="simple.html"

12. За допомогою якого оператора можна перервати виконання циклу?
- </B>
- close
- break

13. За допомогою якого об'єкту можна отримати доступ до рядка стану?
- Image
- Status
- Button

14. Який тег HTML не є тегом організації списку?
- UL
- MAP
- OL

ДОДАТКИ

Додаток А.

Нові HTML5-теги. Модифікація тегів HTML4

В таблиці 1 вміщені теги, які були добавлені в HTML5. В полі Підтримка показано стан підтримки тега в сучасних браузерах. Можливі значення:

- *Повна* - тег підтримується всіма сучасними браузерами;
- *Часткова* - тег частково підтримується сучасними браузерами.

Таблиця 1

Назва тега	Опис	Підтримка
<article>	Визначає незалежний вміст сторінки.	**Повна**
<aside>	Визначає опосередковано зв'язані із вмістом елементи.	**Повна**
<audio>	Вставляє аудіо файл.	**Повна**
<canvas>	Дозволяє малювати довільні об'єкти з допомогою скриптів.	**Повна**
<datalist>	Визначає список з даними для елемента введення.	**Часткова**
<details>	Визначає поле з додатковою інформацією, яке користувач може приховувати чи відображати.	**Часткова**
<embed>	Дозволяє вставити зовнішній вміст.	**Повна**
<figcaption>	Визначає підпис для зображення.	**Повна**
<figure>	Використовується для створення підписів до зображень.	**Повна**
<footer>	Визначає футер для документа.	**Повна**
<header>	Визначає блок заголовка для документа.	**Повна**
<hggroup>	Дозволяє згрупувати заголовки.	**Повна**
<keygen>	Генерує відкритий і закритий ключ	**Часткова**

	для безпечного зв'язку із сервером.	
<mark>	Підсвічує (*виділяє*) текст.	**Часткова**
<meter>	Визначає смугу вимірювання.	**Часткова**
<menu>	Дозволяє визначати контекстні меню і панелі інструментів.	**Часткова**
<nav>	Визначає навігаційний блок сайту.	**Повна**
<output>	Використовується для виведення даних.	**Часткова**
<progress>	Відображає стан виконання якого-небудь процесу.	**Часткова**
<rp>	Вказує, що повинно відображатися в браузерах, які не підтримують агати.	**Повна**
<rt>	Дозволяє задати агати (*використовуються для уточнення читання ієрогліфів китайської і японської мови*).	**Повна**
<ruby>	Дозволяє задати ієрогліф, прочитання якого необхідно уточнити.	**Повна**
<section>	Дозволяє згрупувати логічно зв'язаний вміст.	**Повна**
<source>	Дозволяє задати декілька джерел для відтворення елементів <audio> і <video>.	**Повна**
<summary>	Визначає видимий заголовок для елемента <details>.	**Часткова**
<video>	Дозволяє доповнювати веб-сторінки відео файлами.	**Повна**
<wbr>	Відмічає відповідне місце в тексті для ймовірного перенесення.	**Часткова**

Змінені теги HTML4

В таблиці 2 вміщені теги, які вже були присутні в HTML4. В полі Статус вказано, які зміни відбулися з цим елементом в HTML5. Можливі значення:

- *Змінено* - цей елемент був змінений в HTML5 (*добавлено/вилучено атрибути чи змінилась поведінка*);
- цей елемент не змінено.

Таблиця 2

Назва тега	Опис	Статус
<!--....-->	Визначає коментар.	-
<!DOCTYPE>	Визначає тип документа.	**Змінено**
<a>	Визначає посилання.	**Змінено**
<abbr>	Визначає абревіатуру.	-
<address>	Визначає контактну інформацію автора документа.	**Змінено**
<area />	Визначає область-посилання на зображення.	**Змінено**
<b>	Визначає жирный текст.	-
<base />	Визначає адресу чи спосіб відкривання всіх посилань за замовчуванням.	-
<bdo>	Визначає напрямок тексту.	-
<blockquote>	Визначає довгу цитату.	-
<body>	Визначає тіло документа.	**Змінено**
** **	Визначає перенесеннння на новий рядок.	-
<button>	Визначає кнопку.	**Змінено**
<caption>	Визначає заголовок таблиці.	**Змінено**
<cite>	Оформляє текст як цитату.	-
<code>	Оформляє текст як комп’ютерний код.	-
<dd>	Визначає значення терміну в списку визначень.	-

<del>	Визначає закреслений текст.	-
** **	Визначає секцію в документі.	-
<dl>	Визначає список визначень.	-
<dt>	Визначає терміни в списку визначень.	-
****	Визначає акцентований текст.	-
<fieldset>	Визначає границю навколо елементів форми.	**Змінено**
<form>	Визначає форму для введення даних користувача.	**Змінено**
<h1> - <h6>	Визначає заголовки.	-
<head>	Визначає довідкову інформацію про документ.	-
<hr />	Визначає горизонтальную лінію.	**Змінено**
<html>	Визначає кореневий тег HTML5-документа.	**Змінено**
<i>	Визначає курсивний текст.	-
<iframe>	Визначає рядковий фрейм.	**Змінено**
<img />	Визначає малюнок (зображення).	-
<input />	Визначає поля введення даних у формі.	**Змінено**
<ins>	Визначає вкладений текст.	-
<label>	Визначає мітку для елемента input.	**Змінено**
<legend>	Визначає заголовок для елементів fieldset, figure, і details.	**Змінено**
** **	Визначає список.	**-**
<link />	Визначає взаємозв’язок між поточним документом і зовнішнім файлом.	**-**
<map>	Визначає карту зображень.	**Змінено**
<meta />	Визначає метадані HTML5-документа.	**Змінено**
<noscript>	Визначає альтернативний вміст для користувачів, браузер яких не підтримує клієнтські скрипти.	-

<object>	Визначає вбудований об’єкт.	**Змінено**
** **	Визначає впорядкований список.	**Змінено**
<optgroup>	Визначає групу схожих елементів у списку вибору.	-
<option>	Визначає варіант у списку вибору.	-
<p>	Визначає абзац.	-
<param />	Визначає параметр для вбудованого об’єкта.	**Змінено**
<pre>	Визначає відформатований текст.	-
<script>	Визначає клієнтський скрипт.	**Змінено**
<select>	Визначає список, що розгортається.	**Змінено**
<small>	Визначає маленький текст.	-
****	Визначає секцію в документі.	-
****	Визначає важливий текст.	**Змінено**
<style>	Визначає інформацію про стиль документа.	**Змінено**
<sub>	Визначає нижній індекс елемента.	-
<sup>	Визначає верхній індекс елемента.	-
<table>	Визначає таблицю.	**Змінено**
<tbody>	Групує вміст тіла таблиці.	**Змінено**
<td>	Визначає клітинку в таблиці.	**Змінено**
<textarea>	Визначає багаторядкове поле для введення тексту.	**Змінено**
<tfoot>	Групує нижній вміст таблиці.	**Змінено**
<th>	Визначає клітинку для заголовка таблиці.	**Змінено**
<thead>	Групує головний вміст таблиці.	**Змінено**
<title>	Визначає заголовок документа.	-
<tr>	Визначає рядок в таблиці.	**Змінено**
** **	Визначає невпорядкований список.	-

Вилучені теги

В таблиці 3 наведені теги, які використовувалися в HTML4, а в HTML5 були вилучені.

Таблиця 3

Назва тега	Опис
<acronym>	Визначає акронім.
<big>	Визначає текст великим шрифтом.
** **	Вирівнює текст по центру.
<dir>	Дозволяє задати список каталогів.
<font>	Дозволяє задати тип, розмір і колір шрифту.
<frame />	Визначає фрейм (вікно) в наборі фреймів.
<frameset>	Визначає набір фреймів.
<noframes>	Визначає альтернативний вміст для користувачів, браузер яких не підтримує фрейми.

Додаток Б.

Властивості і селектори CSS

Таблиця 1

Властивості CSS
Властивості тексту

Властивість	Опис	Введено в
@font-face	Дозволяє підключити до веб-сторінок довільні шрифти.	CSS3
font	Дозволяє встановити всі властивості шрифту (font-family, font-size, font-style, font-variant, font-weight) одним визначенням.	CSS1
font-family	Дозволяє встановити шрифт тексту.	CSS1
font-size	Дозволяє встановити розмір тексту.	CSS1
color	Дозволяє встановити колір тексту.	CSS1
text-shadow	Дозволяє прив’язати тінь (або	CSS3

	декілька тіней) до тексту елемента.	
text-decoration	Дозволяє оформити текст елемента.	CSS1
text-align	Дозволяє визначити горизонтальне вирівнювання тексту.	CSS1
letter-spacing	Дозволяє визначити відстань між символами тексту.	CSS1
word-spacing	Дозволяє визначити відстань між словами тексту.	CSS1
line-height	Дозволяє встановити висоту рядків.	CSS1
font-style	Встановлює стиль шрифту елемента.	CSS1
font-variant	Дозволяє відобразити текст елемента капітелем.	CSS1
font-weight	Дозволяє встановити товщину шрифту.	CSS1
text-overflow	Дозволяє вказати, як повинен відображатися текст, що вийшов за межі границь елемента.	CSS3
vertical-align	Дозволяє встановити вертикальне вирівнювання тексту.	CSS1
text-transform	Дозволяє управління регістром символів у тексті.	CSS1
text-indent	Дозволяє встановити величину відступу першого символа тексту.	CSS1
text-justify	Дозволяє встановити алгоритм вирівнювання для властивості "text-align:justify".	CSS3
word-wrap	Дозволяє вказати, чи повинні довгі слова, що виходять за межі батьківського елемента, разбиватися і переноситися на новий рядок.	CSS3
white-space	Дозволяє встановити, як повинні оформлятися пробіли в тексті елемента.	CSS1

quotes	Дозволяє встановити, як повинні оформлятися лапки, що вставлені тегом <q>.	CSS1
direction	Дозволяє встановити напрямок написання тексту.	CSS1

Властивості вирівнювання

Властивість	Опис	Введено в
height	Дозволяє встановити висоту елемента.	CSS1
width	Дозволяє встановити ширину елемента.	CSS1
margin	Дозволяє встановити величину всіх зовнішніх відступів в одному визначенні.	CSS1
margin-top	Дозволяє встановити величину зовнішнього відступу від верхнього краю елемента.	CSS1
margin-right	Дозволяє встановити величину зовнішнього відступу від правого краю елемента.	CSS1
margin-bottom	Дозволяє встановити величину зовнішнього відступу від нижнього краю елемента.	CSS1
margin-left	Дозволяє встановити величину зовнішнього відступу від лівого краю елемента.	CSS1
padding	Дозволяє встановити величину всіх внутрішніх відступів в одному визначенні.	CSS1
padding-top	Дозволяє встановити величину внутрішнього відступу від верхнього краю елемента.	CSS1

padding-right	Дозволяє встановити величину внутрішнього відступу від правого краю елемента.	CSS1
padding-bottom	Дозволяє встановити величину внутрішнього відступу від нижнього краю елемента.	CSS1
padding-left	Дозволяє встановити величину внутрішнього відступу від лівого краю елемента.	CSS1
float	Притискує елемент до вказаного боку батьківського елемента і змушує наступні елементи сторінки "обгортати" його.	CSS1
position	Дозволяє встановити метод розміщення елемента.	CSS2
top	Дозволяє встановити величину відступу від верхнього краю елемента.	CSS2
right	Дозволяє встановити величину відступу від правого краю елемента.	CSS2
bottom	Дозволяє встановити величину відступу від нижнього краю елемента.	CSS2
left	Дозволяє встановити величину відступу від лівого краю елемента.	CSS2
clear	Дозволяє "очистити" елемент від дії float.	CSS2
max-height	Дозволяє встановити максимальну висоту елемента.	CSS2
max-width	Дозволяє встановити максимальну ширину елемента.	CSS2
min-height	Дозволяє встановити мінімальну висоту елемента.	CSS2
min-width	Дозволяє встановити мінімальну	CSS2

	ширину елемента.	
z-index	Дозволяє встановити порядок відображення елементів при накладанні їх один на одного.	CSS2
clip	Дозволяє обрізувати елемент, який розміщений абсолютно.	CSS2
column-count	Дозволяє розділити текст елемента на вказану кількість колонок.	CSS3
column-gap	Дозволяє встановити величину відступу між колонками тексту.	CSS3
column-width	Дозволяє встановити ширину колонок.	CSS3
column-rule	Дозволяє встановити всі властивості оформлення розділювача колонок тексту в одному визначенні.	CSS3
column-rule-color	Дозволяє встановити колір розділювача.	CSS3
column-rule-style	Дозволяє встановити стиль розділювача.	CSS3
column-rule-width	Дозволяє встановити ширину розділювача.	CSS3
column-span	Дозволяє встановити, на скільки колонок повинен розтягуватися елемент.	CSS3
box-align	Дозволяє встановити спосіб вирівнювання елементів-нащадків.	CSS3
box-direction	Дозволяє встановити напрямок відображення елементів-нащадків.	CSS3
box-flex	Дозволяє встановити, чи повинні елементи- нащадки бути гнучкими в розмірах.	CSS3
box-ordinal-group	Дозволяє встановити порядок відображення елементов-нащадків.	CSS3

box-orient	Дозволяє встановити, як повинен бути орієнтований елемент.	CSS3
box-pack	Дозволяє встановити позицію елементів- нащадків по горизонталі в горизонтально орієнтованих елементах і позицію по вертикалі в вертикально орієнтованих елементах.	CSS3

Властивості фону

Властивість	Опис	Введено в
background	Дозволяє встановити всі властивості фону в одному визначенні.	CSS1
background-size	Дозволяє встановити розмір фонового зображення.	CSS3
background-attachment	Дозволяє визначити, чи буде фоновий малюнок закріпленим, чи буде прокручуватися із вмістом сторінки.	CSS1
background-color	Дозволяє встановити колір фону елемента.	CSS1
background-image	Дозволяє встановити фоновий малюнок для елемента.	CSS1
background-position	Дозволяє встановити координати розташування фонового малюнка.	CSS1
background-repeat	Дозволяє встановити, як має повторюватися фоновий малюнок, щоб заповнити фон елемента.	CSS1
background-clip	Дозволяє вказувати, як повинен обрізуватися фон елемента.	CSS3
background-origin	Дозволяє задати позицію, від якої буде відраховуватися місце розташування фонового зображення.	CSS3

Властивості границь

Властивість	Опис	Введено в
border	Дозволяє встановити всі властивості границь в одному визначенні.	CSS1
border-color	Дозволяє встановити колір для всіх границь елемента в одному визначенні.	CSS1
border-style	Дозволяє встановити стиль для всех границ элемента в одному визначенні.	CSS1
border-width	Дозволяє встановити ширину всіх границь елемента в одному визначенні.	CSS1
border-radius	Дозволяє встановити форму всіх кутів елемента в одному визначенні.	CSS3
border-bottom-left-radius	Дозволяє визначити форму нижнього лівого кута елемента.	CSS3
border-bottom-right-radius	Дозволяє визначити форму нижнього правого кута елемента.	CSS3
border-top-left-radius	Дозволяє визначити форму верхнього лівого кута елемента.	CSS3
border-top-right-radius	Дозволяє визначити форму верхнього правого кута елемента.	CSS3
border-image	Дозволяє використовувати зображення як границю елемента.	CSS3
border-top	Дозволяє встановити всі властивості для верхньої границі елемента в одному визначенні.	CSS1
border-top-color	Дозволяє визначити колір верхньої границі елемента.	CSS1
border-top-style	Дозволяє визначити стиль верхньої	CSS1

	границі елемента.	
border-top-width	Дозволяє визначити ширину верхньої границі елемента.	CSS1
border-left	Дозволяє встановити всі властивості лівої границі елемента в одному визначенні.	CSS1
border-left-color	Дозволяє визначити колір лівої границі елемента.	CSS1
border-left-style	Дозволяє визначити стиль лівої границі елемента.	CSS1
border-left-width	Дозволяє визначити ширину лівої границі елемента.	CSS1
border-bottom	Дозволяє встановити всі властивості нижньої границі в одному визначенні.	CSS1
border-bottom-color	Дозволяє визначити колір нижньої границі елемента.	CSS1
border-bottom-style	Дозволяє визначити стиль нижньої границі елемента.	
border-bottom-width	Дозволяє визначити ширину нижньої границі елемента.	CSS1
border-right	Дозволяє встановити всі властивості для правої границі елемента в одному визначенні.	CSS1
border-right-color	Дозволяє визначити колір правої границі елемента.	CSS1
border-right-style	Дозволяє визначити стиль правої границі елемента.	CSS1
border-rigth-width	Дозволяє визначити ь ширину правої границі елемента.	CSS1
outline	Дозволяє встановити всі властивості зовнішньої границі в одному визначенні.	CSS2
outline-color	Дозволяє визначити колір зовнішньої	CSS2

	границі елемента.	
outline-style	Дозволяє визначити стиль зовнішньої границі елемента.	CSS2
outline-width	Дозволяє визначити ширину зовнішньої границі елемента.	CSS2
outline-offset	Дозволяє встановити величину зміщення зовнішньої границі від границі елемента.	CSS3

Властивості переходів і анімації

Властивість	Опис	Введено в
@keyframes	Блок для визначення анімації.	CSS3
animation	Дозволяє встановити всі властивості анімації (*крім animation-play-state*) в одному визначенні.	CSS3
animation-name	Дозволяє встановити ім'я анімації.	CSS3
animation-duration	Дозволяє вказати час виконання анімації в секундах.	CSS3
animation-timing-function	Дозволяє вказати функцію пом'якшення, яка забезпечує плавність виконання анімації.	CSS3
animation-delay	Дозволяє встановити величину затримки початку виконання анімації.	CSS3
animation-iteration-count	Дозволяє встановити кількість повторень анімації.	CSS3
animation-direction	Дозволяє вказати, чи повинна анімація виконуватися в оберненому порядку.	CSS3
animation-play-state	Дозволяє вказати, чи повинна анімація виконуватися, чи має бути зупинена.	CSS3
transition	Дозволяє задати всі властивості	CSS3

	переходів в одному визначенні.	
transition-property	Дозволяє вказати CSS- властивості, які будуть використані для створення переходу.	CSS3
transition-duration	Дозволяє вказати час, за який буде здійснено перехід.	CSS3
transition-timing-function	Дозволяє вказати функцію пом'якшення, яка забезпечує плавність виконання переходу.	CSS3
transition-delay	Дозволяє встановити величину затримки початку виконання переходу.	CSS3

Властивості таблиць і списків

Властивість	Опис	Введено в
list-style	Дозволяє встановити всі властивості списків в одному визначенні.	CSS1
list-style-image	Дозволяє встановити довільний малюнок замість маркера списку.	CSS1
list-style-position	Дозволяє встановити, де повинен відображатися маркер списку.	CSS1
list-style-type	Дозволяє визначити вид маркера списку.	CSS1
border-collapse	Дозволяє встановити, чи повинні границі таблиці и клітинок об'єднуватися.	CSS2
border-spacing	Дозволяє встановити відстань між границями суміжних клітинок.	CSS2
caption-side	Дозволяє встановити місце розташування табличного заголовка.	CSS2
empty-cells	Дозволяє встановити, як повинні відображатися порожні клітинки таблиці.	CSS2

Інші властивості CSS

Властивість	Опис	Введено в
display	Дозволяє встановити спосіб відображення елемента.	CSS1
opacity	Дозволяє встановити рівень прозорості елемента.	CSS3
appearance	Дозволяє перетворити тег у вказаний елемент користувацького інтерфейсу.	CSS3
resize	Дозволяє розтягування елемента.	CSS3
overflow	Дозволяє вказати, що повинно відбутися, якщо вміст елемента вийде за його межі.	CSS2
visibility	Дозволяє встановити, чи повинен елемент бути видимим.	CSS2
cursor	Дозволяє встановити вид курсора миші.	CSS2
transform	Дозволяє трансформувати елемент.	CSS3
transform-origin	Дозволяє змінити місце розташування трансформованого элемента.	CSS3
content	Дозволяє вставляти довільний вміст.	CSS2

Таблиця 2

Селектори CSS

В полі статус вказано: використовувався цей селектор у попередніх версіях CSS (-) чи був добавлений у CSS3 (новий).

Селектор	Приклад	Опис	Статус
#ідентифікатор	#el1	Вибере всі елементи на сторінці, які мають ідентифікатор el1 (або які мають атрибут id='el1').	-
.клас	.group1	Вибере групу елементів на сторінці, які мають клас group1 (які мають атрибут class='group1').	-
елемент	p	Вибере всі абзаци на сторінці.	-
*****	*	Вибере всі елементи на сторінці.	-
:not(x)	*:not(div)*	Вибере всі елементи на сторінці, крім елементів div.	новий
:link	a:link	Вибере всі не відвідані посилання на	-

		сторінці.	
:visited	a:visited	Вибере всі відвідані посилання на сторінці.	-
:hover	a:hover	Вибере все ссылки, на которые наведен курсор мыши.	-
:active	a:active	Вибере всі активні в даний момент посилання (на які клацнули мишкою).	-
[атрибут]	p[id]	Вибере всі абзаци на сторінці, які мають атрибут id.	-
::selection	*::selection*	Оформляє виділений користувачем текст.	новий
[атрибут=значення]	p[id="el1"]	Вибере всі абзаци на сторінці, які мають атрибут id із значенням el1.	-
[атрибут~=значення]	a[href~="wisdomweb]"	Вибере всі посилання з атрибутом href, який містить в	-

		значенні підрядок "wisdomweb", відокремлений пробілами від іншого вмісту.	
[атрибут^=значення]	*[src^="http://"]*	Вибере всі елементи, що мають атрибут src із значенням, яке починається з "http://".	новий
[атрибут$=значення]	*[src$=".gif"]*	Вибере всі елементи, що мають атрибут src із значенням, яке закінчується на ".gif".	новий
[атрибут=*значення]	*[src*="picture"]*	Дозволяє вибрати всі елементи, що мають атрибут src із значенням, яке містить підрядок "picture".	новий
:first-child	p:first-child	Вибере всі перші абзаци в батьківському елементі.	-
:last-child	*div:last-child*	Дозволяє вибрати всі елементи div,	новий

		які є останніми елементами-нащадками в батьків- ському елементі.	
:first-line	p:first-line	Оформляє перший рядок усіх абзаців на сторінці.	-
:first-letter	p:first-letter	Оформляє першу букву всіх абзаців на сторінці.	-
ел1 ел2	div p	Вибере всі абзаци, які є нащадками елемента div.	-
ел1 > ел2	div > p	Вибере всі абзаци, які є нащадками елемента div.	-
ел1 + ел2	div + p	Вибере всі абзаци, які слідують за елементом div.	-
елемент1~елемент2	*div~p*	Вибере всі елементи div, які розміщені перед елементом p.	новий
:before	p:before	Вставить довільний вміст перед елементом p.	-

:after	p:after	Вставить довільний вміст після елемента p.	-
:focus	input:focus	Вибере всі активні елементи введення на сторінці.	-
:enabled	*:enabled*	Дозволяє вибрати всі працездатні елементи введення.	новий
:disabled	*:disabled*	Дозволяє вибрати всі непрацездатні елементи введення.	новий
:first-of-type	*div:first-of-type*	Дозволяє вибрати всі елементи div, які перші в батьківському елементі.	новий
:last-of-type	*div:last-of-type*	Дозволяє вибрати всі елементи div, які останні в батьківському елементі.	новий
:only-of-type	*div:only-of-type*	Дозволяє вибрати всі	новий

		елементи div, які є унікальними в батьківському елемен.	
:nth-child(x)	*div:nth-child(3)*	Дозволяє вибрати всі елементи div, які є третіми в батьківському елементі.	новий
:nth-last-child(x)	*div:nth-last-child(2)*	Дозволяє вибрати всі елементи div, які є другими елементами-нащадками в батьківському з кінця.	новий
:root	*:root*	Дозволяє вибрати кореневий елемент документа.	новий
:empty	*p:empty*	Дозволяє вибрати порожні абзаци.	новий

Додаток В.

Опис повідомлень валідаторів на основі HTML Tidy

У цьому додатку наведено опис основних помилок і попереджень, які відображають валідатори, що працюють на основі HTML Tidy, а також способи розв'язання виявлених проблем.

- `entity "..." doesn't end in ";"`

В кінці спецсимвола немає крапки з комою (наприклад, ` ` замість ` `).

- `numeric character reference "..." doesn't end in ';'`

В кінці числового спецсимвола немає крапки з комою (наприклад, `&#8212` замість `&#8212;`).

- `unescaped & or unknown entity "&..."`

У HTML-коді виявлений символ &, зарезервований для створення спецсимволів. Для розв'язання проблеми замініть символ & спецсимвол `&`.

- `missing </...>` Для парного тега не виявлено закриваючий тег.
- `missing </aaa> before <bbb>`

Блочний тег розміщений всередині вбудованого, що неприпустимо правилами HTML.

- `discarding unexpected <...>`

Для парного тега не виявлено або відкриваючий тег, або закриваючий тег.

- `nested emphasis ...`

Помилка говорить про повторне застосуванні одного і того ж тега фізичного форматування. Наприклад, `<b><b>текст</b></b>` потрібно замінити на `<b>текст</b>`.

- `replacing unexpected ... by </...>`

Всередині парного тега виявлені один незакритий парний

тег і один парний тег без відкриваючої пари. Наприклад, `<div><em>текст</strong></div>`.

- `isn't allowed in <...> elements`

Всередині парного тега виявлений тег, який за правилами не може перебувати в даному парному тезі.

- `missing <...>` Не виявлено тег, який обов'язково повинен розташовуватися усередині даної структури. Наприклад, відсутність тега `<tr>` при створенні таблиці.

- `inserting implicit <...>`

Виявлена помилка, у виникненні якої винні раніше виявлені валідатором проблеми.

- `Insert missing <title> element`

В HTML-документі не виявлено тег `<title>`.

- `Multiple <frameset> elements`

В HTML-документі виявлено кілька тегів `<frameset>`, не вкладених один в одного. Для вирішення проблеми необхідно вкласти теги `<frameset>` один в одного.

- `<...> is not approved by W3C`

В HTML-документі виявлений тег, який не входить в специфікацію HTML. Замініть його аналогічним за функціоналом.

- `Error: <...> is not recognized!`

В HTML-документі виявлений тег, невідомий валідатору.

- `Trimming Empty Tag` Всередині парного тега немає ніякого тексту або стоїть пробіл. Для вирішення проблеми замініть звичайний пробіл спецсимволом ` `; або введіть в тег який-небудь текст.

- `<тег> is probably intended as </тег>`

Для зазначеного в повідомленні тега виявлений імовірно закриваючий тег, але без слеша. Наприклад:
`<a href="">текст<a>.`

- `... shouldn't be nested` Виявлено кілька однакових тегів, вкладених один в одного, а за правилами для цих тегів вкладення один в одного заборонено.
- `Text found after closing </body>-tag`

В HTML-документі після закриваючого тега `</body>` виявлений якийсь текст.

- `Adjacent hyphens within comment`

У тілі коментаря виявлені два і більше дефіси підряд. Такі конструкції в коментарях заборонені. Наприклад: `<!-- текст -- текст -->.`

- `SYSTEM, PUBLIC, W3C, DTD, EN must be upper case`

Слова `SYSTEM, PUBLIC, W3C, DTD, EN` в `DOCTYPE` обов'язково повинні бути
написані у верхньому регістрі. Інакше виникне така помилка.

- `missing <!DOCTYPE> declaration`

В HTML-документі не виявлено `DOCTYPE`.

- `Too much <...>-elements`

В HTML-документі виявлено кілька примірників тега, який повинен зустрічатися тільки один раз.

- `<...> inserting "..." attribute`
  `<...> lacks "..." attribute`

Для тега не вказано обов'язковий атрибут.

- `... attribute ... lacks value`

Для атрибуту тега не вказано обов'язкове значення.

- `... attribute "..." has invalid value "..."`

Для атрибуту тега вказано некоректне значення.

- `<...> missing > for end of tag`

У HTML-коді виявлений символ > замість використання спецсимвола `>`:

`<p>текст > текст</p>`. Ще однією причиною виникнення такої помилки є відсутність символа > в кінці тега: `<p замість <p>`.

- `<...> proprietary attribute "..."`

Для тега виявлений атрибут, який не входить в специфікацію HTML цього тега. Замініть даний атрибут аналогічним за функціональністю кодом.

- `... proprietary attribute value "..."` Для атрибута виявлено значення, яке не входить в специфікацію HTML для цього атрибута.

- `... dropping value "..." for repeated attribute "..."`

В тегу виявлено кілька однакових атрибутів.

- `... unexpected or duplicate quote mark` При написанні значення тега пропущені відкриваючі або закриваючі лапки. Наприклад: `<a href=http://test.uk">`.

- `attribute with missing trailing quote mark` Для значення тега кількість відкриваючих і закриваючих лапок не збігається. Наприклад: `<a href="http://test.ru"">`.

- `id and name attribute value mismatch`

Для тега одночасно вказаний атрибут `id` і атрибут `name`. Причому значення для них відрізняються.

- `replacing <...> by <...>`

Ця помилка виникає в наступних випадках: неправильний порядок тегів; є зайвий закриваючий тег; є парний відкриваючий тег, але для нього відсутній закриваючий.

- `... anchor "..." already defined`

У документі виявлено кілька тегів з однаковим значенням атрибута `name`.

Література та електронні джерела

1. Гаевский, А. Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов [Текст] / А. Ю. Гаевский, В. А.Романовский. – Технолоджи-3000, Триумф. - 2008. – 464 с.
2. Дронов В. PHP, My SQL Dreamweaver MX 2004. Разработка интерактивных Web-сайтов [Текст] / В. Дронов. – СПб. : БХВ-Петербург, 2005. – 448 с.
3. Зудилова, Т. В. Web–программирование JavaScript [Текст] / Т. В. Зудилова, М. Л. Буркова. – СПб : НИУ ИТМО, 2012. -68 с.
4. Клименко, Р. А. Веб-мастеринг на 100% [Текст] / Р. А. Клименко. – СПб. : Питер, 2014. – 512 с.
5. Пасічник, О. Г. Основи веб-дизайну [Текст] / О. Г. Пасічник, О. В. Пасічник, І. В. Стеценко. – К. : Вид. BHV. - 2009. – 336 с.
6. Прохоренок, Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера [Текст] / Н. А. Прохоренок. – СПб. : БХВ-Петербург, 2010. - 912 с.
7. Росс, В. С. Создание сайтов : HTML, CSS, PHP, MySQL. Учебное пособие, ч. 1 [Текст] / В. С. Росс. – М. : МГДД(Ю)Т, 2010. – 107 с.
8. Стейнмец, У. PHP : 75 готовых решений для вашего сайта [Текст] / У. Стейнмец , Б. Вард. – СПб. : Наука и Техника, 2009. – 256 с.
9. Сырых, Ю. А. Современный веб-дизайн. Эпоха Веб 3.0 [Текст] / Ю. А. Сырых. – М.: “И. Д. Вильямс”, 2013. – 368 с.
10. Ташков, П. А. Веб-мастеринг на 100% : HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка [Текст] / П. А. Ташков.- СПб. : Питер, 2010. – 512 с.
11. Флэнаган, Д. JavaScript. Подробное руководство [Текст] / Д. Флэнаган. – СПб. : Символ-Плюс, 2008. – 992 с.
12. [Електронний ресурс] / Режим доступу : http://htmlbook.ru/
13. [Електронний ресурс] / Режим доступу : http://stepbystep.htmlbook.ru/
14. [Електронний ресурс] / Режим доступу : http://www.intuit.ru/department/internet/operawebst/

Навчальне видання

Зубик Людмила Володимирівна
Карпович Іван Миколайович
Степанченко Ольга Миколаївна

ОСНОВИ СУЧАСНИХ WEB-ТЕХНОЛОГІЙ. ЧАСТИНА 1

Навчальний посібник

Друкується в авторській редакції

Підписано до друку _____ Формат $60 \times 84^{1}/16$
Папір друкарський №1. Гарнітура Times. Друк різографічний.
Ум.–друк. арк. 16,9. Обл.–вид. арк. 17,7.
Тираж 100 прим. Зам № ____

Видавець і виготовлювач
Редакційно-видавничий центр
Національного університету
водного господарства та природокористування
33028, Рівне, вул .Соборна, 11

Свідоцтво про внесення суб'єкта виданичої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції РВ №31 від 26.04.2005