Міністерство освіти і науки України
Сумський державний університет

5052 Методичні вказівки

до практичних і лабораторних занять
із курсу «**Інформаційні та вебтехнології**»
для студентів спеціальності 171 «*Електроніка*»
освітнього ступеня «бакалавр»
денної форми навчання

У двох частинах

ЧАСТИНА 1

Статичні вебтехнології

Суми
Сумський державний університет
2021

Методичні вказівки до практичних і лабораторних занять із курсу «Інформаційні та вебтехнології» : у 2 ч. Ч 1 : Статичні вебтехнології / укладач Ю. М. Шабельник. – Суми : Сумський державний університет, 2021. – 64 с.

Кафедра електроніки, загальної та прикладної фізики

ЗМІСТ

ВСТУП

Інформаційні технології – невід'ємна частина сучасного світу. Їх використовують в усіх сферах діяльності людини – від космічної галузі до повсякденного життя; вони розвиваються й удосконалюються разом із прогресом науки та техніки.

Розроблення нових комп'ютерних технологій дозволило суспільству наблизитися до вирішення глобальної проблематики інформатизації, пов'язаної зі швидким поширенням інтеграційних процесів,насамперед, їх проникненням в усі сфери нашої діяльності: науку, культуру, освіту, виробництво, управління тощо.

Інформатизація громадськості – це глобальний соціальний процес, особливість якого в переважанні таких видів діяльності у сфері суспільного виробництва, як збирання, накопичення, оброблення, зберігання, передавання, використання, продукування інформації, здійснювані за допомогою сучасних засобів мікропроцесорної та обчислювальної техніки, а також різноманітних засобів інформаційної взаємодії та обміну.

Інформаційні технології, що базуються на використанні Інтернету, телекомунікаційних мереж та інтелектуальних комп'ютерних систем, відкривають перед майбутнім поколінням можливості вільного розповсюдження знань. Єдині світові інформаційні системи забезпечують уведення інформаційних технологій в освіту: формування єдиного освітнього простору, актуалізацію потреби людини в суспільстві й одержання доступу до загальних нематеріальних ресурсів, зокрема осмислення та оброблення великого об'єму інформації.

Інтернет-технології майбутнього

1. Найшвидша у світі бездротова технологія передавання даних – за допомогою світлових вихрів. Її винайшли й уперше застосували у 2011–2012 рр. учені з університету Південної Каліфорнії, Тель-Авівського університету та Лабораторії НАСА з вивчення реактивного руху. Зазначена технологія дозволяє прискорити бездротове передавання інформації до 2,5 Тбіт/с (приблизно 320 Гбайт/с). Зазначена технологія поки що перебуває на початковому етапі розвитку, тому передавати дані за допомогою світлових вихрів можна лише на дуже невелику відстань. Учені змогли стабільно передавати інформацію лише на відстань 1 м.

2. Найпотужніша у світі бездротова технологія передавання даних – за допомогою нейтрино-променів. Її можна застосовувати для передавання сигналу крізь будь-які предмети. Частинки нейтрино здатні проходити через будь-які перешкоди, не взаємодіючи з матеріалом. Зокрема, ученим з університету Рочестера вдалося передати повідомлення через 240-метрову кам'яну брилу, чого не можна зробити за допомогою жодної з нині доступних бездротових технологій. Якщо нейтрино-промені почнуть використовувати на практиці, то сигналу буде не потрібно огинати Землю: він просто проходитиме крізь неї. Це значно спростило б інтернет-з'єднання між материками й іншими віддаленими один від одного об'єктами. Для цієї технології передавання інформації необхідний потужний прискорювач частинок, яких у світі всього кілька. Учені, які вивчають передавання даних через нейтрино-промені, використовують прискорювач частинок «Fermilab» (діаметром 4 км) та детектор частинок «MINERvA» (вагою 5 т).

3. Технологія RedTacton, застосовувана для передавання даних через біологічний канал – шкіру людини. Чи замислювалися ви тоді, коли дивилися фільм про шпигунів і теж хотіли одним дотиком руки одержувати інформацію на свій телефон, про можливість обмінюватися електронними візитками й будь-якими іншими даними за допомогою рукостискання або роздруковувати документи, просто провівши рукою по принтеру? Усе це й ще багато іншого може стати реальністю, якщо зазгачену технологію продовжать розробляти.

RedTacton базується на принципі, що кожна людина має електромагнітне поле, а її шкіра може служити каналом передавання сигналу між декількома електронними пристроями. В основі технології лежить використання електрооптичних кристалів, властивості яких змінюються в результаті дії електромагнітного поля людини. Потім із кристалів за допомогою лазера зчитуються зміни та переводять у більш прийнятний формат. Водночас система RedTacton може функціонувати не лише у звичайних умовах, а й під водою, у вакуумі, космосі.

Інтернет – усесвітня мережа, яка об’єднує безліч комп’ютерних мереж різного рівня та окремих комп’ютерів, що обмінюються з іншими своєю інформацією за каналами громадських телекомунікацій на базі протоколів зв’язку TCP / IP. Інформація в Інтернеті зберігається на серверах (сайтах). Сервери, об’єднані високошвидкісними магістралями, становлять базу мереж Інтернету. Доступ до інформаційних ресурсів Інтернету користувачі зазвичай одержують через провайдерів або корпоративну мережу.

В Інтернеті функціонують кілька сервісів, або служб (E-mail, USENET, TELNET, WWW, FTP та ін.), але найбільш

популярний із них – WWW. Він побудований за принципом клієнт – сервер. Служба складається з серверів, доступ до яких здійснюють через клієнтські додатки або браузери. Основний обсяг інформаційних ресурсів (вебсторінок або файлів у форматі html) знаходиться на вебсайтах, розміщених на вебсерверах (хостингах) мережі Інтернет.

Сайт є набором вебсторінок, об'єднаних загальною тематикою й пов'язаних між собою гіперпосиланнями, єдиною системою навігації. Прикладним протоколом для передавання гіпертексту (вебсторінок) є http (https), зазначений у URL або адресі будь-якого ресурсу (документа чи файлу) в Інтернеті.

Залежно від технології створення можна виділити такі типи сайтів:

1) **статичні сайти**, що містять статичні HTML- або XHTML-сторінки. Статичні вебсторінки – це статичні файли (набори тексту, таблиць, рисунків та ін.), створювані за допомогою мови розмітки HTML (мають розширення .html або .htm), які зберігають у готовому вигляді у файловій системі сервера;

2) **динамічні сайти**, на яких вебсторінки генеруються або формуються (динамічно створюються) у процесі виконання запиту користувача. Динамічні сайти бувають двох типів: на яких вебсторінки генеруються або формуються з даних, збережених на сервері в базі даних; на яких вебсторінки генеруються на стороні клієнтського додатка (у браузері);

3) **flash-сайти** – це інтерактивні програми, розроблені в середовищі Macromedia Flash. Основним інструментом розроблення flash-програм є векторна графіка (інтерактивна векторна анімація для web). Flash додає сайтам динамічності й інтерактивності. Його доцільно використовувати, якщо мало

тексту, але потрібні звукові або анімаційні ефекти, тобто для створення векторних анімаційних файлів із невеликим часом завантаження. Основні недоліки цієї технології створення повноцінних flash-сайтів – великий об'єм вебсторінок і висока вартість розроблення сайтів.

Крім того, сайти, повністю створені на основі Flash, погано індексують пошукові системи. Flash-технології здебільшого застосовують для розроблення престижних сайтів. Для створення flash-анімацій доречна технологія Adobe Flash, що дає можливість працювати з мовами ActionScript та ActionScript 2.0;

4) **комбіновані сайти**, для створення яких застосовують комбінації вищезазначених технологій.

За взаємодією користувача з ресурсами вебсторінки сайти можна поділити на пасивні й активні, або інтерактивні.

Пасивні сайти – це сайти з пасивними вебсторінками. На пасивних сайтах користувач має змогу лише переглядати інформацію на вебсторінках.

Здебільшого статичні сайти з пасивними вебсторінками використовують для створення невеликих і середніх сайтів із постійною структурою та виглядом сторінок (але кожна сторінка може мати свій шаблон оформлення), які можна розміщувати на будь-яких хостингах, зокрема, безкоштовних, що не підтримують роботу скриптів. Навчання школярів і студентів основ створення сайтів доцільно починати з розроблення статичних сайтів із пасивними сторінками, тобто з вивчення мови розмітки HTML і каскадних таблиць стилів CSS.

Інтерактивні сайти – це сайти з активними вебсторінками, під час роботи з якими користувач має змогу

обмінюватися даними із сервером, брати участь в інтерактивному діалозі.

Для додавання статичним вебсторінкам інтерактивності й динамічності в них можна вставляти скрипти на мовах сценаріїв JavaScript і VBScript, виконуваних на стороні клієнта. Скрипти на JavaScript та VBScript можуть виконуватися або за будь-яких дій користувача, або автоматично під час завантаження вебсторінки.

Крім того, у HTML-документ можна додавати елементи DHTML (динамічного HTML) – способу створення інтерактивного вебсайту. Динамічний HTML побудований на мові програмування JavaScript, каскадних таблицях стилів CSS і DOM (об'єктній моделі документа).

Останніми роками широкого розповсюдження набули різноманітні інтернет-додатки, зокрема пошукові інформаційні системи, засоби електронної комерції, соціальні мережі, інформаційні, наукові та освітні портали, вебсайти органів державного управління тощо. Крім того, велику кількість програмного забезпечення, для функціонування якого використовують так званий віконний інтерфейс користувача, поступово переводять у веборієнтований спосіб взаємодії з користувачем для підвищення ефективності розроблення й супроводу.

Усі ці завдання потребують ґрунтовних професійних знань і навичок. З огляду на це основна увага в навчальному посібнику спрямована на ознайомлення з принципами й прийомами створення клієнтської частини вебсайту засобами html та CSS.

Видання також містить матеріали для самостійної роботи студентів, орієнтованої на закріплення знань, здобутих у

результаті аудиторних занять. Для оцінювання та практичної апробації одержаних знань у кінці кожного розділу розміщені контрольні запитання. Поглибленому вивченню матеріалу сприятимуть літературні й електронні джерела, перелік яких наведено наприкінці методичних вказівок.

Тема 1. Мова розмітки гіпертексту HTML

Гіпертекстовий документ – це текстовий файл, що має спеціальні мітки – теги, які «розпізнаються» браузером і використовуються ним для відображення вмісту файлу на екрані комп'ютера. За допомогою цих міток можна виділяти заголовки документа, змінювати колір, розмір літер, додавати графічні зображення й таблиці, але основною перевагою гіпертексту порівняно зі звичайним є можливість додавання до вмісту документа гіперпосилань – спеціальних конструкцій мови HTML.

Гіпертекст – це текст, що містить зв'язки з іншими документами («гіперзв'язки», або «гіперпосилання»), тобто читач має змогу переходити до пов'язаних документів безпосередньо з вихідного (первинного) тексту, активізувавши посилання. Останні створюють за допомогою спеціальних кодів, що задають форматування тексту. Ці коди визначені в мові розмітки HTML.

Виділяють два способи створення гіпертекстових документів. Можна скористатися одним із WYSIWYG HTML-редакторів (наприклад, «Netscape Composer», «Microsoft FrontPage» та ін.), для роботи з якими не потрібно спеціальних знань про внутрішню структуру створюваного документа. Цей спосіб дозволяє розробляти документи для WWW без знання мови HTML. HTML-редактори автоматизують створення гіпертекстових документів, позбавляють від рутинної роботи, але їх можливості обмежені, вони значно збільшують розмір файлу і не завжди одержаний за допомогою них результат відповідає очікуванням розробника.

Альтернативою служить створення й розмітка документа за допомогою звичайних редакторів plain-тексту (таких, як «Emacs» або «NotePad++»). У разі застосування цього способу в текст вручну вставляють команди мови HTML.

HTML (англ. HyperText Markup Language – мова розмітки

гіпертекстових документів, що базується на SGML) – текстова мова, призначена для розмітки документів, які містять текст, зображення, гіперпосилання тощо. Процес розміщення в тексті кодів HTML називають ***розміткою***.

HTML 5 – це остання версія HTML. У ній ураховані зміни, що відбулися в Інтернеті з часу створення попередньої версії у 1999 році. У HTML 5 було додано багато нового, тому певні можливості, доступні в HTML 4 лише з використанням зовнішніх плагінів (зокрема, «Adobe Flash») або клієнтських скриптів, тепер стали доступними для звичайних тегів. HTML 5 підтримують майже всі сучасні браузери, що дозволяє використовувати нові можливості без зайвих побоювань.

Найважливіші нововведення HTML 5:

– підтримка відео й аудіо (елементи video та audio);

– реалізація можливості малювання на вебсторінках довільних об'єктів (елемент canvas);

– підтримка форм більш складної структури (нові значення type в елементі `<input>` і значна кількість нових елементів та атрибутів);

– додавання семантичних тегів, що дозволяють робити вебсторінки більш зрозумілими для пошукових систем, браузерів та інших програм чи пристроїв, які читають вебсторінки (елементи footer, header, nav, article);

– організація сховищ DOM (Document Object Model) – більш функціональної альтернативи cookie.

Якщо HTML-документ відкрити в редакторі текстів, він буде відображеним як звичайний текстовий файл, а якщо в браузері – відповідно до тегів розмітки. В останньому разі він називатиметься **вебсторінкою**.

Для звернення до сайта використовують доменне ім'я, яке складається з кількох доменів, відокремлених крапкою. Наприклад, sumdu.edu.ua означає, що домен третього рівня sumdu входить у домен другого рівня edu, який відповідно входить у домен першого (верхнього) рівня ua. Домени

верхнього рівня поділяють на загальні й географічні. Перші призначені для певного класу організацій, наприклад com – для комерційних сайтів; org – для некомерційних організацій; net – для сайтів, діяльність яких пов'язана з наданням інтернет-послуг. Географічні домени розроблені для конкретної країни, наприклад ua – України, us – Сполучених Штатів Америки, eu – Європейського Союзу, de – Німеччини.

Документи створені мовою HTML, називають HTML-документами. Вони нагадують звичайні текстові файли за винятком інтерпретування певних символів у них (так званих тегів) як міток.

Тег (від англ. tag – ярлик, етикетка, бірка; розмічувати, маркувати) – це символ розмітки мови HTML, який наводять у кутових дужках (< >). Тег ще називають дескриптором. Використання дескрипторів дає можливість форматувати HTML-документи.

За допомогою дескрипторів у HTML-документі можна форматувати такі елементи: параграфи, розділи, абзаци, списки, малюнки, таблиці, колонтитули, індекси, зміст тощо. Кожний елемент має свою унікальну назву, записувану латинськими літерами та нечутливу до їх регістру. Загалом елемент має три складові:

– ***теги*** (початковий (opening tag) і кінцевий (closing tag)) – це назва елемента. Початковий тег вказує, де елемент починається або починає діяти, зокрема де починається абзац; кінцевий тег відрізняється від початкового лише наявністю косої лінії й свідчить про кінець елемента;

– ***зміст*** (контент (content)) – це текстова та графічна інформація документа, відтворювана браузером на екрані;

– ***елемент*** (element) – початковий і кінцевий теги й контент разом становлять елемент;

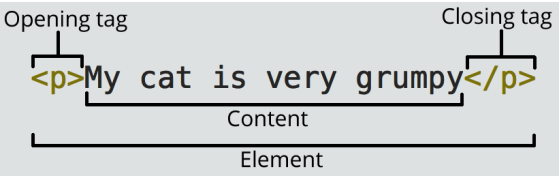

Рисунок 1.1 – Структура елемента абзацу

– ***атрибути*** містять додаткову інформацію про елемент, яку ви не хочете показувати у фактичному контенті. Зокрема, class – ім'я атрибута, а editor-note – його значення. Клас дозволяє дати елементу ідентифікаційне ім'я, що пізніше можна використовувати для звернення до нього з інформацією про стилі та ін.

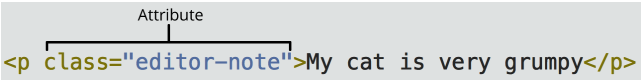

Рисунок 1.2 – Приклад атрибута class

Атрибут завжди повинен мати:

– пробіл між ним та ім'ям елемента (або попереднім атрибутом, якщо елемент уже має один або кілька атрибутів);

– ім'я атрибута, після якого розміщений знак «дорівнює»;

– значення атрибута, узяте в лапки з двох сторін.

Теги поділяють на одиничні та парні. Перші не можуть зберігати в собі інформації безпосередньо, вона прописується як значення атрибута, наприклад тег `<input type="button" value="Кнопка">` створить кнопку з текстом «**Кнопка**» всередині.

Теги можна вкладати один в один, наприклад `<p><i>Текст</i></p>`. У такому разі варто додержуватися порядку їх закриття (принципу «матрьошки»), наприклад запис `<p><i>Текст</p></i>` буде неправильним.

HTML-елементи можуть мати **глобальні** атрибути, застосовувані для всіх HTML-елементів, і **власні**. Атрибути прописуються у початковому тегу елемента, що містять ім'я та значення, які зазначають у форматі `ім'я атрибута="значення"`. Атрибути дозволяють змінювати властивості й поведінку елемента, для якого вони задані.

Кожному елементу можна присвоїти кілька значень `class` і лише одне значення `id`. Множинні значення `class` записують через пробіл: `<div class="nav top">`. Значення `class` та `id` повинні складатися лише з букв, цифр, дефісів та нижніх підкреслень і починатися виняткого з букв або цифр.

Браузер переглядає (інтерпретує) HTML-документ, вибудовуючи його структуру (DOM) та відображаючи її відповідно до інструкцій, унесених у цей файл (таблиць стилів, скриптів). Якщо розмітка правильна, у вікні браузера буде відображеною HTML-сторінка, що містить HTML-елементи: заголовки, таблиці, зображення тощо.

Процес інтерпретації (парсингу) починається раніше, ніж вебсторінка повністю завантажується в браузер. Браузери обробляють HTML-документи послідовно, спочатку, водночас обробляючи CSS і порявнюючи таблиці стилів з елементами сторінки.

HTML-документ складається з двох розділів – заголовка (між тегами `<head>` … `</head>`)та змістової частини (між тегами `<body>` … `</body>`).

Під час планування і створення будь-якого вебресурсу варто пам'ятати, що основний критерій розроблення сторінок – зручність користувача, тобто майбутнього відвідувача сайта. Зважаючи на це виникає необхідність певної стандартизації підходів до вебдизайну, вироблення алгоритмів, що могли б задовольнити всю потенційну аудиторію.

Загальноприйнята структура HTML-документа

Якщо відкрити будь-яку вебсторінку, вона міститиме в собі типові елементи, що не змінюватимуться залежно від типу й тематики сайта. На рисунку 1.3 наведений код найпростішого документа, що містить основні теги.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Web-page Template</title>
</head>
<body>
<h1>Title</h1>
<p>First paragraph</p>
<p>Second paragraph</p>
</body>
</html>
```

Рисунок 1.3 – Структура HTML-документа

Для перегляду результату виконання зазначеного коду, потрібно зберегти його у файлі з іменем sample.html і запустити для виконання через будь-який відомий браузер.

Атрибути тегів

Щоб розширити можливості окремих тегів та більш гнучко керувати вмістом контейнерів, застосовують ***атрибути тегів***.

Для атрибутів тегів використовують стандартні значення. Якщо тегу не буде додано певного допустимого атрибута, браузер надаватиме йому стандартного значення. Якщо ви очікуєте на інший результат на вебсторінках, необхідно точно задати значення певних атрибутів.

Атрибути без значень

Деякі атрибути можна використовувати в тегах, не надаючи їм значень, як зображено на рисунку 1.4.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Web-page Template</title>
</head>
<body>
<form action="self.php">
<p><input type="text"></p>
<p><input type="submit" disabled></p>
</form>
</body>
</html>
```

Рисунок 1.4 – Використання атрибутів тегів

У наведеному прикладі використовують вимкнений атрибут без значення. Такий запис називають «скороченим атрибутом тегу».

Порядок атрибутів у тегах

Порядок атрибутів у будь-якому тегу не має значень і не впливає на результат відображення елементів. Отже, теги `<img src="/images/title.gif" wigth="438" height="118">` та `<img height="118" wigth="438" src="/images/title.gif">` виконують однакові функції.

Формат атрибутів

Кожний атрибут тегу належить до певного типу (наприклад, текст, число, шлях до файлу та ін.), який обов’язково потрібно врахувати під час написання атрибута.

Зокрема, тег `<img>` додає на вебсторінку, а його атрибут `width` задає ширину зображення в пікселях. Якщо поставити не число, а щось інше, то значення буде проігнорованим і виникне помилка під час валідації документа.

Таблиця 1.1 – Список стандартних HTML-атрибутів

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 стить елемента
`title`	Містить додаткову інформацію про елемент (значення цього атрибута відображається на екрані як підказка під час наведення курсора мишки на елемент)

Довідник по HTML-тегам: https://css.in.ua/html/tags

Контрольні питання до теми 1

1. Основні можливості мови HTML.
2. Визначення HTML-документа.
3. Правила запису тегів у HTML-документі.
4. Чим відрізняються парні і непарні теги?
5. Чим відрізняються теги рівня блоку та послідовні теги?
6. Технологія створення HTML-документів.
7. Атрибути тегів.

Тема 2. Правила побудови HTML-документів. Макет вебсторінки

Структура HTML-документа

Мова HTML додержується правил, що містяться у файлі оголошення типу документа (Document Type Definition, або DTD). Document Type Definition є XML-документом, що визначає, які теги, атрибути та їх значення дійсні для конкретного типу HTML. Для кожної версії HTML є свій DTD, що відповідає за коректне відображення вебсторінки браузером, і визначає не лише версію HTML (наприклад, html), а й відповідний DTD-файл в Інтернеті.

```
<!DOCTYPE html> <!-- Оголошує формат документу -->
<html>
<head> <!-- Технічна інформація про документ -->
<meta charset="UTF-8"> <!-- Визначаємо кодування символів документу -->
<title>...</title> <!-- Задаємо заголовок документу -->
<link rel="stylesheet" type="text/css" href="style.css"> <!- - Підключаємо зовнішні таблиці стилів -->
<script src="script.js"></script> <!-- Підключаємо сценарії -->
</head>
<body><!-- Основна частина документу -->
</body>
</html>
```

Рисунок 2.1 – Структура HTML-документа

Елементи всередині тегу `<html>` утворюють дерево документа, так звану об’єктну модель – DOM (document object model). Водночас елемент `<html>` є кореневим елементом.

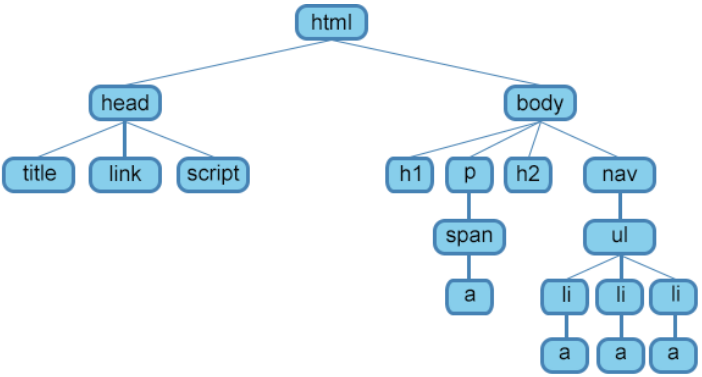

Рисунок 2.2 – Найпростіша структура вебсторінки

Щоб пояснити взаємодію елементів вебсторінки, необхідно розглянути так звані «родинні відносини» між множинними вкладеними елементами. Їх поділяють на батьківські, дочірні й сестринські.

Предок – елемент, що містить у собі інші елементи. На рисунку 2.2 предком для всіх елементів є `<html>`. Водночас елемент `<body>` є предком для всіх розміщених у ньому ключових слів: `<h1>`, `<p>`, `<span>`, `<nav>` та ін.

Нащадок – елемент, розміщений усередині одного або більше типів елементів. Наприклад, `<body>` є нащадком `<html>`, а елемент `<p>` – одночасно `<body>` і `<html>`.

Батьківський елемент – елемент, пов’язаний з іншими елементами нижчого рівня, що знаходиться на дереві вище за них. На рисунку 2.2 `<html>` є батьківським лише для `<head>` та `<body>`. Тег `<p>` батьківський лише для `<span>`.

Дочірній елемент – елемент, безпосередньо підлеглий іншому елементу вищого рівня. На рисунку 2.2 дочірніми щодо `<body>` є лише елементи `<h1>`, `<h2>`, `<p>` і `<nav>`.

Сестринський елемент – елемент, що має загальний батьківський елемент з іншим, тобто елемент того самого рівня.

На рисунку 2.2 `<head>` і `<body>` – елементи одного рівня, так само як елементи `<h1>`, `<h2>`, `<p>` є сестринськими між собою.

Основні теги мови HTML

Елемент <html>

Є кореневим елементом документа. Інші елементи розміщені всередині тегів `<html>` ... `</html>`. Усе, що знаходиться за межами тегів, браузер не сприймає як код HTML, тому не обробляє. Для елемента доступні атрибути manifest і xmlns, а також глобальні атрибути.

Атрибут	Опис, значення
`manifest`	За допомогою значення атрибута вказується шлях до документа кешу маніфесту, наприклад `<html manifest="about_company.appcache">`

Елемент <head>

Розділ `<head>` ... `</head>` містить технічну інформацію про сторінку: заголовок, опис, ключові слова для пошукових систем, кодування та ін. Уведена в ньому інформація не відображається у вікні браузера, але містить дані, що повідомляють браузеру, як необхідно обробляти сторінку.

Елемент <title>

Обов'язковим тегом розділу `<head>` є `<title>`. Текст, розміщений усередині цього тегу, відображається в рядку заголовка веббраузера. Довжина заголовка повинна не перевищувати 60 символів, щоб повністю поміститися в ньому. У тексті заголовка потрібно наводити максимально повний опис вмісту вебсторінки.

Елемент <meta>

Необов'язковим тегом розділу `<head>` є одинарний тег `<meta>`. За допомогою нього можна задати опис вмісту сторінки й ключові слова для пошукових систем, автора HTML-документа та інші властивості метаданих. Елемент `<head>` може містити кілька елементів `<meta>`, тому що залежно від використовуваних атрибутів вони несуть різну інформацію.

За допомогою тегу `<meta>` можна заборонити або дозволити індексацію вебсторінки пошуковими системами.

Індексація й перехід за посиланнями дозволені:

```
<meta name="robots" content="index, follow">
```

Індексація дозволена, перехід за посиланнями заборонений:

```
<meta name="robots" content="index, nofollow">
```

Індексація та перехід за посиланнями заборонені:

```
<meta name="robots" content="noindex, nofollow">
```

Для автоматичного перезавантаження сторінки через заданий проміжок часу потрібно скористатися значенням refresh:

```
<meta http-equiv="refresh" content="30">
```

Сторінка буде перезавантаженою через 30 секунд. Щоб спрямувати відвідувача на іншу сторінку, потрібно зазначити URL-адресу в параметрі url:

```
<meta http-equiv="refresh" content="0;
url=https://www.google.com/">
```

Для елемента `<meta>` доступні атрибути `charset`, `content`, `http-equiv`, `name`, а також **глобальні атрибути**.

Елемент <style>

Усередині цього елемента задають стилі, використовувані на сторінці. Для задання стилів в HTML-документі передбачена мова CSS. Таких елементів на сторінці може бути кілька.

Для елемента доступні атрибути `media`, `scored`, `type`, а також **глобальні атрибути**.

Усередину цього елементу можна записувати код форматування як елементів, так і вебсторінки повністю.

```
<style type="text/css">
.paper {
width: 200px;
height: 300px;
background-color: #ef4444;
color: #666666;
}
```

Щоб підключити до елемента заданий стиль, необхідно через атрибут `class` (або `id`) присвоїти йому відповідну назву:

```
<div class="paper">
...
</div>
```

CSS-код можна вбудовувати безпосередньо в елемент розмітки як значення атрибута `style`, наприклад

```
<p style="color: #666666; background-color: #ef4444; padding:
20px;">
```

Елемент <link>

Задати стилі для документа можна також за допомогою іншого способу – записати їх в окремий файл із розширенням .css, наприклад style.css.

Підключити файл зі стилями до вебсторінки можна двома способами:

– через директиву **@import url**:

```
<!DOCTYPE html>
<html>
<head>
<style>
@import url(style.css);
</style>
<meta>
<title> </title>
</head>
```

– з використанням елемента `<link>`, що не потребує закриття тегу. Зазначений елемент визначає відношення між поточною сторінкою та іншими документами. Таких елементів на сторінці може бути кілька. Запис буде таким

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

Для елемента доступні атрибути `href`, `hreflang`, `media`, `rel`, `type`, а також **глобальні атрибути**.

Елемент <script>

Елемент `<script>` дозволяє приєднувати до документа різні сценарії. Водночас текст сценарію може бути розміщеним або всередині цього елемента, або в зовнішньому файлі. Якщо текст сценарію розміщений у зовнішньому файлі, його підключають за допомогою атрибутів елемента. Для елемента доступні атрибути `async`, `charset`, `defer`, `src`, `type`, а також **глобальні атрибути** (див. додаток А).

Визначення вмісту вебсторінки

Елемент <body>

У цьому розділі знаходиться весь вміст документа. Для елемента доступні глобальні атрибути.

Таблиця 2.1 – Атрибути елементу <body> та його значення

Атрибут	Опис
`onafterprint`	Подія, що спрацьовує після відправки сторінки на друк або закриття вікна друку
`onbeforeprint`	Подія, що спрацьовує перед відправкою сторінки на друк
`onbeforeunload`	Подія, що спрацьовує, якщо відвідувач ініціював перехід на іншу сторінку або натиснув «закрити вікно». Дозволяє відображати повідомлення в діалоговому вікні підтвердження, щоб повідомити користувачеві, хоче він залишитися чи залишити поточну сторінку.
`onhashchange`	Подія спрацьовує, якщо змінюється hash-частина URL, наприклад користувач переходить із адреси `example.domain/test.aspx#page1` на `example.domain/test.aspx#page2`
`onmessage`	Подія відбувається, якщо повідомлення одержано через джерело події
`onoffline`	Подія викликається браузером тоді, коли він визначає, відсутність з'єднання з Інтернетом.
`ononline`	Подія викликається браузером тоді, коли з'єднання з Інтернетом відновлюється.
`onpagehide`	Подія відбувається, якщо користувач залишає сторінку за допомогою навігації, наприклад натиснувши на посилання,

	оновивши сторінку, заповнивши форму тощо
`onpageshow`	Подія відбувається, якщо користувач переходить на вебсторінку, тобто після події onload
`onunload`	Подія спрацьовує якщо сторінка не завантажилася внаслідок певних причин або під час закриття вікна браузера

Контрольні питання до теми 2

1. Правила побудови HTML-документів.
2. Макет вебсторінки.
3. Основні теги мови HTML.
4. Тег <body>.
5. Глобальні атрибути.

Тема 3. Робота з текстом

Заголовки

Пошукові системи використовують заголовки для індексації структури й вмісту вебсторінок. З огляду на це, щоб сторінка виводилася в пошуковій системі за відповідним запитом, потрібно використовувати змістові заголовки.

Теги заголовків (`<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`) призначені для виділення на екрані користувача певних фрагментів тексту HTML-сторінки. Текст виділяють за допомогою зміни розмірів та «жирності» тексту (рис. 3.1).

Парний тег `<hgroup>` дозволяє групувати теги заголовків (від `<h1>` до `<h6>`) вебсторінки або розділу.

Теги заголовків також є парними. Їх можна використовувати з необов'язковим параметром `align`, що служить для горизонтального вирівнювання тексту. Можливі значення параметра `align` наведені в таблиці 3.1.

Таблиця 3.1 – Значення параметра `align`

Значення параметра	**Результат використання**
`left`	Текст вирівнюється по лівому краю вікна браузера
`center`	Текст вирівнюється по центру вікна браузера
`right`	Текст вирівнюється по правому краю вікна браузера
`justify`	Текст вирівнюється по ширині вікна браузера

Якщо параметр, `align` не заданий, застосовують вирівнювання заголовка по центру вікна браузера.

```
<!DOCTYPE html>
<html>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
</body>
</html>
```

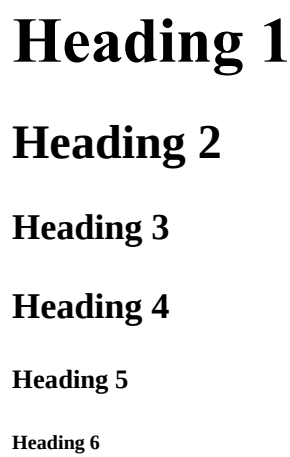

Рисунок 3.1 – Приклад відображення заголовків від 1-го рівня до 6-го рівня

Списки

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

Марковані списки

Для створення маркованого списку використовують тег-контейнер `<ul>`, у якому розміщують усі його елементи. Водночас кожний пункт списку повинен починатися тегом `<li>`.

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

– `disc` – маркери відображаються заповненими колами;

– `circle` – маркери відображаються незаповненими колами;

– `square` – маркери відображаються заповненими квадратами.

Варіант 1

```
<ul>
<li> Перший пункт маркованого списку </li>
<li> Другий пункт маркованого списку </li>
<li> Третій пункт маркованого списку </li>
…
<li> і-й пункт маркованого списку </li>
</ul>
```

Варіант 2

```
<ul style="list-style-type:circle;">
<li> Перший пункт маркованого списку </li>
<li> Другий пункт маркованого списку </li>
<li> Третій пункт маркованого списку </li>
…
<li> і-й пункт маркованого списку </li>
</ul>
```

Варіант 3

```
 <ul style="list-style-type:square;">
 <li> Перший пункт маркованого списку </li>
 <li> Другий пункт маркованого списку </li>
 <li> Третій пункт маркованого списку </li>
 …
 <li> і-й пункт маркованого списку </li>
 </ul>
```

Якщо в тегах `<ul>` і `<li>` не використовують параметра `type`, то стандартним для всього списку буде маркер disc.

Нумеровані списки

Для створення нумерованого списку використовують тег-контейнер `<ol>`, усередині якого розміщують усі його елементи списку. Водночас кожний пункт списку повинен починатися тегом `<li>`.

Варіант 1

```
<ol>
<li> Перший пункт маркованого списку </li>
<li> Другий пункт маркованого списку </li>
<li> Третій пункт маркованого списку </li>
…
<li> і-й пункт маркованого списку </li>
</ol>
```

Часто тег `<ol>` використовують з необов'язковими параметрами `type` та `start`. Водночас параметр `type` використовують також у тегу `<li>`. Його завданням є визначення одного зі стандартних типів маркера. Для нумерованого списку є п'ять стандартних типів маркерів. Використання цього параметра в тегу `<ol>` поширюється на весь список, а в тегу `<li>` – лише на поточний пункт. Можливі значення параметра `type` і відповідні їм стандартні типи маркерів наведені в таблиці 3.2.

Таблиця 3.2 – Значення параметра `type` і типи маркерів

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

Варіант 2

```
<ol type="A;">
<li> Перший пункт маркованого списку </li>
<li> Другий пункт маркованого списку </li>
<li> Третій пункт маркованого списку </li>
<li> і-й пункт маркованого списку </li>
</ol>
```

Параметр `start` призначений для зміни початку нумерації пунктів списку і може бути використовуваним лише в тегу `<ol>`.

Списки визначень

Список визначень використовують для розміщення на HTML-сторінці тексту, подібного до енциклопедії, чи словника. Кожний пункт такого списку складається з двох частин: у першій записують термін, що потребує визначення, а в другій – текст, що пояснює його зміст.

```
<dl>
<dt> HTML </dt>
<dd> текстова мова розмітки, призначена для розмітки документів, що містять текст, зображення, гіперпосилання тощо. </dd>
<dt> CSS </dt>
<dd> мова таблиць стилів, що дозволяє прикріплювати стиль (наприклад, шрифти й колір) до структурованих документів (наприклад, документів HTML і додатків XML). </dd>
</dl>
```

Список визначень задають за допомогою тегу-контейнера `<dl>`. Для визначення терміна використовують тег `<dt>`, а для визначення пояснення – `<dd>`.

Форматування основного тексту HTML-сторінки

Безпосередньо всередині тегу `<body>` не повинно бути ні тексту, ні зображень. За правилами будь-який текст або зображення потрібно розміщувати всередині блокових чи рядкових тегів, списків або таблиць.

Таблиця 3.3 – Основні теги форматування тексту:

Тег	Призначення
`<strong>`	Відображення тексту напівжирним шрифтом
`<em>`	Відображення тексту курсивом

`<u>`	Відображення тексту підкресленим
`<s>`	Відображення тексту перекресленими символами
`<sub>`	Відображення тексту у вигляді верхнього індекса
`<sup>`	Відображення тексту у вигляді нижнього індекса
`<font>`	Визначення параметрів шрифту: кольору, розміру тощо

Контрольні питання до теми 3

1. Блокові теги HTML.
2. Рядкові теги HTML.
3. Параметри тегу `<font>`.
4. Перенесення тексту на вебсторінці на наступний рядок.
5. Вирівнювання тексту.
6. Організація списків HTML.

Тема 4. Каскадні таблиці стилів CSS

CSS (Cascading Style Sheets) – мова таблиць стилів, що дозволяє прикріплювати стиль (наприклад, шрифти й колір) до структурованих документів (наприклад, документів HTML і додатків XML). Зазвичай CSS-стилі використовують для створення та зміни стилю елементів вебсторінок і призначених для користувача інтерфейсів, написаних на мовах HTML та XHTML, але вони також можуть бути застосованими до будь-якого виду XML-документа, зокрема XML, SVG і XUL. Відокремлюючи стиль подання документів від їх вмісту, CSS спрощує створення вебсторінок та обслуговування сайтів.

Cascading Style Sheets підтримує таблиці стилів для конкретних носіїв, тому автори можуть адаптувати подання своїх документів до браузерів, слухових пристроїв, принтерів та ін.

Каскадні таблиці стилів описують правила форматування елементів за допомогою властивостей і допустимих значень цих властивостей. Для кожного елемента можна використовувати обмежений набір властивостей, а інші на нього не впливатимуть.

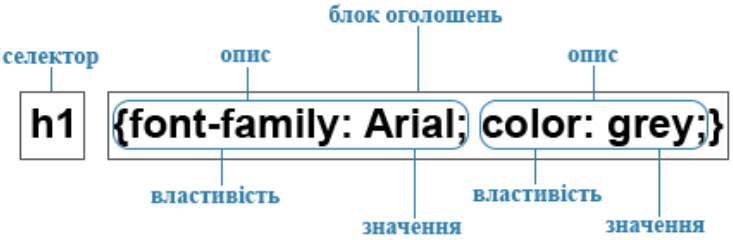

Рисунок 4.1 – Структура оголошення

Оголошення стилю складається з двох частин: **селектора** та **оголошення**. У HTML імена елементів не чутливі до регістру, тому «h1» працює так само, як і «H1». Оголошення складається з двох частин: імені властивості (наприклад, `color`) і

її значення (`grey`). Селектор повідомляє браузеру, який саме елемент форматувати, а в блоці оголошення (коді у фігурних дужках) зазначені команди форматування – властивості та їх значення.

CSS3 – це новий стандарт оформлення HTML-документів, що значно розширює можливості попереднього стандарту CSS2.1. Багато можливостей, що потребували залучення додаткових зовнішніх програм (наприклад, «Adobe Photoshop» або «JavaScript») можуть достатньо просто бути реалізованими у CSS3 завдяки новим властивостям оформлення.

CSS3 дозволяє:

– створювати елементи зі згладженими кутами;
– створювати лінійні й сферичні градієнти;
– додавати до елементів і тексту різноманітні тіні;
– використовувати не лише «безпечні» шрифти;
– створювати анімацію та різні ефекти переходів.

Види таблиць стилів

Зовнішня таблиця стилів

Зовнішня таблиця стилів – це текстовий файл із розширенням `.css`, у якому знаходиться набір CSS-стилів елементів. Файл створюють у редакторі коду, так само як HTML-сторінку. Усередині файл може містити лише стилі без HTML-розмітки. Зовнішню таблицю стилів підключають до вебсторінки за допомогою тегу `<link>`, розміщеного всередині розділу `<head> </head>`. Такі стилі функціонують для всіх сторінок сайта.

До кожної вебсторінки можна приєднати кілька таблиць стилів. Для цього потрібно послідовно додати кілька тегів `<link>`, зазначивши в атрибуті тегу `media` призначення кожної таблиці стилів. Запис `rel="stylesheet"` свідчить про тип посилання (посилання на таблицю стилів).

```
<head>
<link rel="stylesheet" href="css/style.css">
<link rel="stylesheet" href="css/assets.css" media="all">
</head>
```

Атрибут `type="text/css"` не є обов'язковим за стандартом HTML5, тому його можна не зазначати. Якщо атрибута немає, стандартно використовуваним є значення `type="text/css"`.

Внутрішні стилі

Внутрішні стилі вбудовують у розділ `<head> </head>` HTML-документа, зазначаючи всередині тегу `<style></style>`. Вони мають переваги порівняно із зовнішніми, але поступаються вбудованим стилям (задають через атрибут `style`).

```
<head>
<style>
h1,
h2 {
color: red;
font-family: "Times New Roman", Georgia, Serif;
line-height: 1.3em;
}
</style>
</head>
<body>
...
</body>
```

Вбудовані стилі

Для застосування вбудованих стилів необхідно написати CSS-код у HTML-файлі безпосередньо всередині тегу елемента за допомогою атрибута `style`:

```
<p style="font-weight: bold; color: red;">Зверніть увагу на цей текст.</p>
```

Такі стилі поширюються лише на той елемент, для якого вони задані.

Правило @import

Правило `@import` дозволяє завантажувати зовнішні таблиці стилів. Щоб директива `@import` функціонувала, її потрібно розмістити в таблиці стилів (зовнішній чи внутрішній) перед усіма іншими правилами:

```
<style>
@import url(mobile.css);
p {
font-size: 0.9em;
color: grey;
}
</style>
```

Правила `@import` також додержуються для підключення вебшрифтів:

```
@import
url(https://fonts.googleapis.com/css?family=Open+Sans&subset=latin,
cyrillic);
```

Типи селекторів

Селектори репрезентують структуру вебсторінки. За допомогою них створюють правила для форматування елементів вебсторінки. Селекторами можуть бути елементи, їх класи й ідентифікатори, а також псевдокласи та псевдоелементи.

Універсальний селектор

Відповідає будь-якому HTML-елементу. Наприклад, `* {margin: 0;}` обнулить зовнішні відступи для всіх елементів сайта. Також селектор можуть використовувати в комбінації з псевдокласом або псевдоелементами: `*:after {CSS-стилі}`, `*:checked {CSS-стилі}`.

Селектор елемента

Селектори елементів дозволяють форматувати всі елементи певного типу на всіх сторінках сайта. Наприклад, `h1 {font-family: Lobster, cursive;}` задасть загальний стиль форматування всіх заголовків `h1`.

Селектор класу

Селектори класу дозволяють задавати стилі для одного й більше елементів з однаковим ім'ям класу, розміщених у різних місцях сторінки або на різних сторінках сайта. Наприклад, для створення заголовка з класом `headline` необхідно додати атрибут `class` зі значенням `headline` у тег `<h1>` і задати стиль для зазначеного класу. Стилі, створені за допомогою класу, можна застосовувати до інших елементів, не обов'язково такого типу.

```
<h1 class="headline">Інструкція користування персональним комп'ютером</h1>

.headline {
text-transform: uppercase;
color: lightblue;
}
```

Якщо елемент має декілька атрибутів класу, їх значення об'єднують через пробіл:

```
<h1 class="headline post-title">Інструкція користування персональним комп’ютером</h1>
```

Селектор ідентифікатора

Селектор ідентифікатора дозволяє форматувати **один** конкретний елемент. Значення `id` повинне бути унікальним, зустрічатися на одній сторінці лише один раз і містити щонайменше один символ. Значення не може містити пробілів.

Немає ніяких інших обмежень щодо форми `id`. Зокрема, ідентифікатори можуть складатися лише з цифр чи розділових знаків, починатися з цифри або підкреслення тощо.

Унікальний ідентифікатор елемента може служити для різних цілей, зокрема посилання на конкретні частини документа з використанням ідентифікаторів фрагментів, націлювання на елемент під час створення сценаріїв, стилізації конкретного елемента із CSS.

```
<div id="sidebar"></div>
```

```
#sidebar {
width: 300px;
float: left;
}
```

Селектор атрибута

Селектори атрибутів вибирають елементи на основі імені атрибута або його значення:

1) `[атрибут]` –усі елементи, що містять зазначений атрибут, `[alt]` –усі елементи, для яких заданий атрибут `alt`;

2) `селектор[атрибут]` – елементи певного типу, що містять атрибут, `img[alt]` –лише рисунки, для яких заданий атрибут `alt`;

3) `селектор[атрибут="значення"]` – елементи певного типу, що містять зазначений атрибут із конкретним значенням, `img[title="flower"]` – усі картинки, назва яких містить слово `flower`;

4) `селектор[атрибут~="значення"]` – елементи частково містять певне значення, наприклад якщо для елемента задано кілька класів через пробіл, `p[class~="feature"]` – абзаци, ім'я класу яких містить `feature`;

5) `селектор[атрибут|="значення"]` – елементи, список значень атрибута яких починається із зазначеного слова, `p[class|="feature"]` – абзаци, ім'я класу яких `feature` або починається на `feature`;

6) `селектор[атрибут^="значення"]` – елементи, значення атрибута яких починається із зазначеного значення, `a[href^="http://"]` – усі посилання, що починаються на `http://`;

7) `селектор[атрибут$="значення"]` – елементи, значення атрибута яких закінчується зазначеним значенням, `img[src$=".png"]` – усі картинки у форматі `png`;

8) `селектор[атрибут*="значення"]` – елементи, значення атрибута яких у будь-якому місці містить зазначене слово, `a[href*="book"]` – усі посилання, назва яких містить `book`.

Селектор псевдокласу

Псевдокласи – це класи, майже не прикріплені до HTML-документа тегами. Вони дозволяють застосувати CSS-правила до елементів під час здійснення події або підпорядковується певним правилом. Псевдокласи характеризують елементи з такими властивостями:

1) `:link` – невідвідуване посилання;

2) `:visited` – відвідуване посилання;

3) `:hover` – будь-який елемент, по якому проводять курсором мишки;

4) `:focus` – інтерактивний елемент, до якого перейшли за допомогою клавіатури або активували мишкою;

5) `:active` – елемент, активізований користувачем;

6) `:valid` – поля форми, вміст яких перевірений у браузері на відповідність зазначеному типу даних;

7) `:invalid` – поля форми, вміст яких не відповідає зазначеним типам даних;

8) `:enabled` – усі активні поля форм;

9) `:disabled` – заблоковані, тобто неактивні, поля форм;

10) `:in-range` – поля форми, значення яких перебувають у заданому діапазоні;

11) `:out-of-range` – поля форми, значення яких не входять у встановлений діапазон;

12) `:lang()` – елементи з текстом на зазначеній мові;

13) `:not(селектор)` – елементи, що не містять зазначеного селектора (класу, ідентифікатора, назви або типу поля форми) – `:not([type="submit"])`;

14) `:target` – елемент із символом `#`, на який посилаються в документі;

15) `:checked` – виділені (вибрані користувачем) елементи форми.

Селектор структурних псевдокласів

Структурні псевдокласи відбирають дочірні елементи відповідно до параметра, зазначеного в круглих дужках:

1) `:nth-child(odd)` – непарні дочірні елементи;

2) `:nth-child(even)` – парні дочірні елементи;

3) `:nth-child(3n)` – кожний третій дочірній елемент;

4) `:nth-child(3n+2)` – вибирає кожний третій елемент, починаючи з другого дочірнього елемента `(+2)`;

5) `:nth-child(n+2)` – вибирає всі елементи, починаючи з другого;

6) `:nth-child(3)` – вибирає третій дочірній елемент;

7) `:nth-last-child()` – у списку дочірніх елементів вибирає елемент із зазначеним розміщенням, аналогічно `:nth-child()`, але, починаючи з останнього, у зворотний бік;

8) `:first-child` – дозволяє оформити лише найперший дочірній елемент тегу;

9) `:last-child` – дозволяє форматувати останній дочірній елемент тегу;

10) `:only-child` – вибирає елемент, що є єдиним дочірнім;

11) `:empty` – вибирає елементи, у яких немає дочірніх елементів;

12) `:root` – вибирає елемент, що є кореневим у документі – `html`.

Селектор структурних псевдокласів типу

Вказують на конкретний тип дочірнього тегу:

1) `:nth-of-type()` – вибирає елементи за аналогією з `:nth-child()`, водночас бере до уваги лише тип елемента;

2) `:first-of-type` – вибирає перший дочірній елемент певного типу;

3) `:last-of-type` – вибирає останній елемент певного типу;

4) `:nth-last-of-type()` – вибирає елемент заданого типу в списку елементів відповідно до зазначеного розміщення, починаючи з кінця;

5) `:only-of-type` – вибирає єдиний елемент зазначеного типу серед дочірніх елементів батьківського елемента.

Селектор псевдоелемента

Псевдоелементи використовують для додавання вмісту генерованого за допомогою властивості `content`:

1) `:first-letter` – вибирає першу букву кожного абзацу, застосовують лише до блокових елементів;

2) `:first-line` – вибирає перший рядок тексту елемента, застосовують лише до блокових елементів;

3) `:before` – додає інформацію, генеровану перед елементом;

4) `:after` – додає інформацію, генеровану після елемента.

Використання селекторів

Комбінування

Щоб більш точно відбирати елементи для форматування, можна використовувати комбінації селекторів:

1) `a[href][title]` – вибере всі посилання, для яких задані атрибути `href` та `title`;

2) `img[alt*="css"]:nth-of-type(even)` – вибере всі парні картинки, альтернативний текст яких містить слово `css`.

Групування

Один і той самий стиль можна одночасно застосовувати до кількох елементів. Для цього необхідно в лівій частині оголошення зазначити через кому потрібні селектори:

```
h1,
h2,
p,
span {
color: tomato;
background: white;
}
```

Спадкування

Спадкування є механізмом, за допомогою якого певні властивості передаються від предка до його нащадків. Специфікацією CSS передбачено спадкування властивостей, що стосуються текстового вмісту сторінки, таких як `color`, `font`, `letter-spacing`, `line-height`, `list-style`, `text-align`, `text-indent`, `text-transform`, `visibility`, `white-space` і `word-spacing`. Здебільшого це зручно, тому що не потрібно задавати розміру шрифту й сімейства шрифтів для кожного елемента вебсторінки.

Властивості щодо форматування блоків не успадковуються. Це `background`, `border`, `display`, `float`, `clear`, `height`, `width`, `margin`, `min-max-height` та `-width`, `outline`, `overflow`, `padding`, `position`, `text-decoration`, `vertical-align` і `z-index`.

Примусове спадкування

За допомогою ключового слова `inherit` можна примусити елемент успадковувати будь-яке значення властивості батьківського елемента. Це працює навіть для тих властивостей, що стандартно не успадковуються.

Особливості CSS-стилів:

1) можуть успадковуватися від батьківського елемента (успадковані властивості або за допомогою значення `inherit`);

2) стилі, розташовані в таблиці стилів нижче, скасовують стилі, розміщені в таблиці вище;

3) до одного елемента можуть бути застосовуваними стилі з різних джерел. Перевірити, які стилі застосовані, можна в режимі розробника браузера. Для цього над елементом потрібно натиснути правою кнопкою мишки й вибрати пункт «Переглянути код» (або щось аналогічне). У правій колонці будуть зазначеними всі властивості, задані для цього елемента або успадковані від батьківського, а також файли стилів, у яких вони наведені, та порядковий номер рядка коду;

4) для визначення стилю можна використовувати будь-яку комбінацію селекторів – селектор елемента, псевдокласу елемента, класу або ідентифікатора елемента.

```
<div id="wrap" class="box clear"></div>

div {border: 1px solid #eee;}
#wrap {width: 500px;}
.box {float: left;}
.clear {clear: both;}
```

Каскадування

Каскадування – це механізм, що керує кінцевим результатом, якщо до одного елемента застосовують різні CSS-правила. Виділяють три критерії, що визначають порядок застосування властивостей: правило `!important`, специфічність і порядок, у якому підключені таблиці стилів.

Правило `!important`

Вагу правила можна задати за допомогою ключового слова `!important`, що додають відразу після значення властивості, наприклад `span {font-weight: bold!important;}`. Правило необхідно розміщувати в кінці оголошення перед закриваючою дужкою, без пробілу. Таке оголошення буде мати пріоритет над усіма іншими правилами. Це правило дозволяє скасовувати значення властивості й встановлювати нове для елемента групи, якщо немає прямого доступу до файлу зі стилями.

Порядок підключених таблиць

Ви можете створити кілька зовнішніх таблиць стилів і підключити їх до однієї вебсторінки. Якщо в різних таблицях будуть наведеними різні значення властивостей одного елемента, то застосується правило з таблиці стилів, розміщеної в списку нижче.

Шрифт. Властивості шрифту

CSS3 надає розробникам повну свободу під час вибору шрифту. У попередніх версіях CSS розробники були змушеними використовувати лише шрифти, гарантовано встановлені на комп'ютері користувача. У CSS3 дозволені будь-які шрифти. Знайдений потрібний шрифт необхідно розмістити на вебсервері й підключити його за допомогою нового CSS3-правила @font-face. Шрифт буде завантаженим та автоматично відображатиметься під час відвідування сторінки користувачем.

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

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

```
<!DOCTYPE html>
<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>
```

Інші властивості шрифтів у CSS:

1) `font-family` – визначає сімейства шрифтів, яких через кому може бути зазначено декілька;

2) `font-style` – визначає тип шрифту. Значення normal – звичайний шрифт (стандартний); italic – курсив; oblique – похилий екранний шрифт з меншим нахилом, ніж курсив, значення застосовують до шрифтів без зарубок;

3) `font-variant` – визначає, як шрифт буде виведеним на екран: у вигляді малих чи великих літер;

4) `font-weight` – визначає товщину виведеного шрифту;

5) `font-size` – визначає кегель (висоту символів) шрифту. Можна задати через абсолютний або відносний розмір.

Безпечні шрифти

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

У CSS шрифти поділяють на такі «сімейства»:

1) **Serif** – мають невеликі зарубки на кінцях символів (*Times New Roman, Georgia*);

2) **Sans-Serif** – не мають зарубок (*Arial, Arial Black, Trebuchet MS, Verdana*);

3) **Monospace** – усі символи шрифтів цього сімейства займають однакову ширину (наприклад, *Courier New*);

4) **Fantasy** – сімейство «химерних» шрифтів, використовуваних здебільшого для створення кольорових не шаблонних заголовків (*Impact*);

5) **Cursive** – сімейство шрифтів рукописного стилю (*Comic Sans MS*).

Контрольні питання до теми 4

1. Що таке CSS?
2. Що таке властивість стилю? Як визначити цей параметр?
3. Як змінити стиль елемента? Наведіть приклад.
4. Коли потрібна конструкція з використанням тегу `<span>` `</span>`. Наведіть приклад.
5. Як установити стиль для блоку, щоб властивості стилю застосовувалися до всіх його елементів? Наведіть приклад.
6. Як використовують контейнер `<style>` `</style>` для зміни стилів елементів? Синтаксис. Приклад використання.
7. Як реалізувати основну концепцію CSS?
8. Які загальні правила визначення й присвоєння стильових властивостей, передбачені CSS? Наведіть приклади.
9. Синтаксис і типи селекторів.

10. Як використати стилі селекторів і їх властивості, розміщені в окремому файлі?

11. Як задають коментарі?

12. До чого не чутливий CSS?

Тема 5 Блочна модель CSS

Кожний блок має так звану область вмісту, у якій знаходиться текст, дочірні елементи, зображення та ін., і необов'язкові зовнішні елементи: `padding`, `border` та `margin`. Розмір кожної області обумовлений відповідними властивостями й може бути нульовим або в разі `margin` негативним.

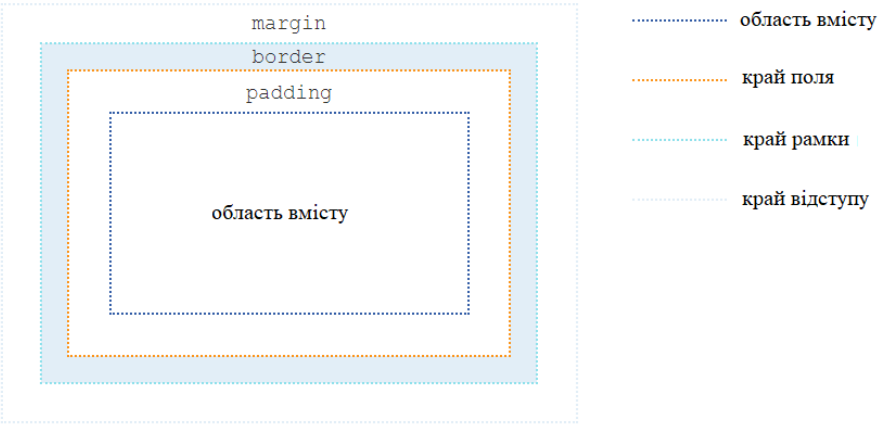

Рисунок 5.1 – Області та краї блоків

Поля, рамка й відступи можуть бути розбитими на верхній, правий, нижній і лівий сегменти.

Фон області вмісту, полів та рамки блоку обумовлений властивостями фону. Область рамки може бути додатково пофарбованою за допомогою властивості `border`. Відступи елемента завжди прозорі, що дозволяє показувати фон батьківського елемента.

Оскільки поля й відступи елемента не є обов'язковими, їх стандартне значення дорівнює нулю. Крім того, певні браузери

додають цим властивостям позитивні стандартні значення на основі своїх таблиць стилів. Очистити стилі браузерів для всіх елементів можна за допомогою універсального селектора

```
* {
margin: 0;
padding: 0;
}
```

Властивості встановлюють верхню, правий, нижній і лівий відступи блоку елемента відповідно. Від'ємне значення дозволене, але можуть бути обмеження під час конкретної реалізації. **Властивості не успадковуються.**

```
margin-top: 20px;
margin-right: 1em;
margin-bottom: 5%;
margin-left: auth;
margin-top: inherit;
margin-right: initial;
```

`margin` є скороченою властивістю для установки `margin-top`, `margin-right`, `margin-bottom` та `margin-left` в одному оголошенні.

Якщо задане лише одне значення, воно застосовується до всіх сторін. Якщо два, верхній і нижній відступи визначає перше значення, а правий та лівий установлюються відповідно до другого. Якщо є три значення, верхній відступ установлюється згідно з першим, лівий і правий – другим, а нижній – третім. Якщо задані всі чотири значення, вони застосовуються до верхнього краю, справа, нижнього краю та зліва відповідно.

Область полів – простір між краєм області вмісту й рамкою елемента. Властивості полів обумовлюють товщину їх області. Властивості застосовують до всіх елементів, крім внутрішніх елементів таблиці (за винятком її комірок). Скорочена властивість `padding` задає поля для всіх чотирьох сторін, а додаткові властивості – лише їх відповідних сторін. Правила задання розмірів полів такі самі, як відступів.

```
paddin-top: 0.5em;
paddin-right: 0;
paddin-bottom: 2cm;
paddin-left: 10%;
paddin-top: inherit;
paddin-right: initial;
```

Рамки елемента заповнюють область рамок, візуально визначаючи краї блоку. Властивості рамок обумовлюють товщину області межі блоку, а також її стиль і колір.

Основні правила написання ефективного коду

У процесі написання CSS варто додержуватися певних принципів, що дозволять скоротити код, зробити його більш зручним, наочним та читабельним. Це означає, по-перше, що розробник через деякий час зможе легко зрозуміти й модифікувати код, а також те, що код може зрозуміти непрофесіонал.

1. ***Розміщуйте каскадні таблиці стилів в окремому файлі.*** Це дозволяє пришвидшити завантаження вебсторінки для зменшення її коду, а також кешування файлів з описом стилю.

2. ***Видаляйте невикористовувані селектори.*** Велика кількість селекторів спричиняє незручності. Насамперед слдадно розібратися, який із з них за що відповідає. Крім того, це збільшує обсяг документа. Щоб цього уникнути, видаляйте

невикористовувані селектори. Часом точно визначити, який селектор використано, а який ні, досить складно, тому рекомендовано додавати коментар у код.

3. ***Застосовуйте групування.*** Перевага, а також зручність застосування групування полягає в описі однакових властивостей в одному місці. Завдяки цьому значення властивості можна прописувати лише один раз, а не повторювати багато разів.

4. ***Використовуйте універсальні властивості.*** Замість задавання значення відступу на кожній стороні елемента через властивості `margin-left`, `margin-right`, `margin-top` і `margin-bottom`, це можна зробити одночасно через універсальну властивість `margin`. Наведення значень через пробіл дозволяє встановити індивідуальні відступи для кожної сторони. Крім `margin`, до універсальних властивостей належать `background`, `border`, `font`, `padding`. Їх застосування скорочує обсяг коду й підвищує його читабельність.

5. ***Форматувати код.*** Виділяють безліч різних підходів до написання CSS-коду. Можна впорядковувати селектори за блоками, відповідно до структури документа, за алфавітом тощо. Крім того, є онлайн-інструмент, що форматує CSS-код відразу чотирма способами (https://www.cssportal.com/format-css/).

6. Мінімізувати код. Редагування CSS-файлу – суперечливе завдання. З одного боку, код повинен бути зручним для сприйняття й унесення правок, швидкого пошуку потрібного селектора, для чого активно використовують коментарі, пробіли та символи табуляції, а з іншого – компактним і не містити в собі нічого зайвого. Компактність дозволяє дещо прискорити завантаження сайта й підвищити його продуктивність.

Контрольні питання до теми 5

1. У чому полягає відмінність між блочними та рядковими елементами?
2. Способи позиціонування елементів у CSS.
3. Для чого використовують float і clear?
4. Зовнішні елементи `content`, `padding`, `border` та `margin`.
5. Основні правила написання коду.

Тема 6. Адаптивний вебдизайн

Адаптивний вебдизайн – це дизайн сторінок в Інтернеті, що дозволяє одержати правильне відображення вебресурсу на будь-яких пристроях, підключених до мережі (смартфонах, планшетах, ноутбуках, ПК). Сторінки з таким дизайном підлаштовуються під параметри вікна браузера, задані користувачем або пристроєм.

Адаптивний вебдизайн і зростання мобільного трафіку

Сьогодні подібний дизайн сторінок сайта є не просто популярним, а необхідним у веброзробленні. Причина цього – велика кількість різних пристроїв і платформ, що відрізняються налаштуваннями. Адаптивна версія сайта буде функціонувати на всіх таких платформах. Це дозволяє витрачати менше зусиль для реалізації ідеї, а також коштує замовникові значно дешевше, ніж розроблення окремої мобільної версії та версії для ПК.

Згідно з результатами досліджень TNS Web Index в останні кілька років із комп'ютерів на сайти заходять лише 29 % користувачів, більшість використовує смартфони або планшети. Якщо брати до уваги вікову категорію «молодші за 35 років», то результати ще цікавіші: лише 10 % користувачів узагалі не користуються мобільними пристроями для перегляду сайтів в Інтернеті.

Що таке адаптивний вебдизайн?

За допомогою адаптивного вебдизайну один і той самий сайт можна без особливих проблем переглядати як на комп'ютерах, так і на мобільних пристроях. Водночас ви не відчуєте будь-якого дискомфорту. Адаптивний дизайн дозволяє відображати сторінки та їх вміст відповідно до пристрою, на якому користувач їх відкриває.

Виділяють кілька типів адаптивних макетів для різних сайтів:

1) *гумовий* – вирізняється простотою реалізації, основні блоки вебсторінки стискаються до ширини пристрою, на якому користувач її переглядає (якщо це неможливо, то блоки перебудовуються в один ряд);

2) *перенос* – якщо у вас багатоколонковий сайт, то цей спосіб є очевидним (зі зменшенням ширини екрана додаткові бічні блоки будуть переноситися в нижню частину сторінки);

3) *перемикання* – є найзручнішим для читання способом, ґрунтується на розробленні окремих макетів для екранів із різною шириною (вирізняється трудомісткістю);

4) *проста адаптивність* – підходить для сайтів вз нескладним дизайном, її особливість – просте масштабування тексту й зображень;

5) *використання панелей* – поява вертикального або горизонтального меню після натискання на спеціальну кнопку (основний недолік – користувачі не завжди розуміють, як функціонує сайт і на що потрібно натискати).

Адаптивний дизайн імейл-листів

Якщо ваш лист красиво сприймається на робочому столі, це зовсім не означає, що він буде виглядати так само на мобільному пристрої. Саме тому адаптація веблистів під будь-які гаджети є дуже важливою для бізнесу. Велика частина аудиторії найрізноманітніших компаній, що займаються імейл-маркетингом, відкриває повідомлення саме на смартфонах або планшетах. Згідно з результатами дослідження Campaign Monitor 2011 року близько 20 % усіх листів переглядають саме на мобільних пристроях. Сьогодні ця цифра значно вища.

Створюючи адаптивні шаблони листів, необхідно враховувати, що:

– на мобільних девайсах найкраще виглядають одностовпчикові шаблони (ширина – не більше ніж 600 пікселів);

– новинні анонси компанії «Apple» свідчать про те, що юзабіліті листів істотно падає, якщо використовувати велику кількість маленьких посилань, за якими складно перейти;

– найкраще створювати лаконічний і зрозумілий лист, маючи у своєму розпорядженні всю найважливішу інформацію у його верхній частині.

Чому сайт повинен бути адаптивним?

Можна виділити кілька основних моментів:

– сьогодні широко використовують найрізноманітніші пристрої, за допомогою яких можна виходити в Інтернет, що відрізняються розміром екрана й роздільною здатністю. З огляду на це вебсайт на кожному з них буде відображатися по-різному;

– популярність мобільних пристроїв продовжує швидко зростати, інтернет-трафік збільшується, тому значна частина аудиторії будь-якої компанії переглядає сайт саме зі смартфонів або планшетів;

– якщо на вашому ресурсі часто з'являється термінова інформація, яку користувач може в будь-який момент прочитати на телефоні, найкраще використовувати саме адаптивний дизайн.

Що таке Mobile First і чому це важливо?

Історично так склалося, що вебдизайнери спочатку розробляють сайт для великих екранів, тому що десктопи пропонують більшу функціональність. Згідно з результатами останніх досліджень, проведених у США, не менше ніж 25 % осіб використовують для інтернет-серфінгу саме мобільні пристрої. А в Китаї цей показник уже досяг 45 %.

Багато елементів функціоналу та дизайну, що добре виглядають на робочому столі, неможливо успішно перенести

на смартфони або планшети. Дуже часто подібна спроба призводить до того, що сайт стає складним або непридатним для перегляду. Філософія Mobile First означає, що дизайнер спочатку продумує дизайн сайта на мобільному девайсі, а вже потім переносить його на ПК або ноутбук.

Навіщо звертати увагу на Mobile First? Це допоможе вашій компанії отримати більше прибутку. Ваш сайт буде завантажуватися на невеликих екранах значно швидше, що дасть чимало переваг. Насамперед, висока швидкість завантаження позитивно впливає на рейтинг сайта в гуглі. Зазначена пошукова система віддає перевагу сайтам із добре оптимізованим адаптивним вебдизайном.

Крім того, завдяки використанню Mobile First ви зможете заощадити на вартості пропускної здатності. Сьогодні все більше покупців використовують саме мобільні пристрої для купівлі. Якщо ви займаєтеся онлайн-бізнесом, Mobile First допоможе вам одержати значну фінансову вигоду.

Основні відмінності адаптивного дизайну від мобільної версії

Попередньо не розібравшись у темі, багато замовників думають, що мобільна та адаптивна версії сайта – це одне й те саме. Проте вони мають чимало відмінностей. Мобільний дизайн розробляють спеціально для перегляду сторінки через девайс із невеликим екраном. Він також здатний вирішити проблему зі зручністю перегляду сайта, але має кілька істотних недоліків:

– вам доведеться створювати окремий мобільний додаток або різний дизайн під кожний тип операційної системи, що потребує додаткових грошових витрат;

– для використання мобільного додатка користувачам буде потрібно завантажувати його на свій пристрій. Не всі з них погоджуються на це, особливо якщо невпевнені, що будуть переглядати сайт щодня;

– через поділ трафіку відвідуваність сайта буде меншою;

– вам доведеться інтегрувати матеріали.

На відміну від мобільних версій адаптивний вебдизайн дозволяє створити один сайт з однією адресою та одним дизайном.

Які переваги адаптивного дизайну?

Основними перевагами адаптивного вебдизайну є:

– охоплення аудиторії й просування – ресурси, створені за цією технологією, краще ранжуються пошуковими системами, тому вирізняються ефективнішим просуванням;

– ви не будете втрачати відвідувачів і потенційних клієнтів – згідно з останніми даними 57 % користувачів закривають сторінку, якщо вона довго завантажується на мобільному пристрої, а завдяки адаптивному дизайну цю проблему можна просто вирішити;

– значний приріст конверсії – створення такого дизайну передбачає детальне опрацювання юзабіліті, що позитивно впливає на конверсію.

Також має значення конкурентоспроможність сайта. За допомогою адаптивного дизайну ви зможете значно випередити своїх конкурентів, навіть успішних, а також залучити більше користувачів.

Основні правила створення адаптивної вебсторінки

Створюючи адаптивні вебсторінки, обов’язково додайте елемент `<meta>`:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Це дозволить установити вікно перегляду сторінки, що повідомить інструкціям браузера, як управляти її розмірами й масштабуванням.

Адаптивні зображення

Адаптивними є зображення, що якісно розміщуються на вебсторінці за будь-якого розміру вікна веббраузера. Якщо CSS-властивість `width` (ширину) встановлено на 100 %, зображення буде реагувати й масштабуватися, зменшуючись або збільшуючись, заповнюючи всю ширину сторінки:

```
<img src="img.jpg" style="width:100%;">
```

Зверніть увагу, що в наведеному вище прикладі зображення може бути розширеним навіть до більших розмірів, ніж його початковий. Кращим рішенням буде використання властивостей `max-width` замість `width`, тобто встановлення максимальної ширини:

```
<img src="img.jpg" style="max-width:100%;height:auto;">
```

HTML-елемент `<picture>` дозволяє визначати різні зображення для різних розмірів вікна браузера.

Адаптивний розмір тексту

Розмір тексту можна встановити за допомогою модуля «vw», що означає «viewport width» (ширину вікна перегляду). Отже, текст відповідає розміру вікна веббраузера.

```
<h2 style="font-size:10vw">Hello World</h2>
```

Примітка 1 Viewport – це розмір вікна браузера. 1vw = 1 % від ширини вікна перегляду. Якщо область перегляду становить 50 см, 1vw – 0,5 см

Є багато шаблонних CSS-фреймворків, що пропонують адаптивний дизайн. Вони безкоштовні й прості у використанні. Один зі способів створити адаптивний дизайн – використання відповідної таблиці стилів, наприклад W3.CSS. Фреймворк W3.CSS дозволяє легко розробляти сайти, що добре виглядають на будь-якому екрані: ПК, ноутбука, планшета або телефона. Інший популярний фреймворк – **Bootstrap**. Він використовує HTML, CSS і jQuery для створення адаптивних вебсторінок.

Контрольні питання до теми 6

1. У чому різниця між адаптивною та мобільною версіями сайта?
2. Що таке Mobile First?
3. Основні правила створення адаптивної вебсторінки.
4. Правила задання адаптивного зображення.
5. Правила задання адаптивного тексту.

Список літератури

1. Зубик Л. В. Основи сучасних web-технологій : навчальний посібник / Л. В. Зубик, І. М. Карпович, О. М. Степанченко. – Рівне : НУВГП, 2016. – Ч. 1. – 290 с.
2. Пасічник В. В. Веб-технології та веб-дизайн : підручник / В. В. Пасічник, О. В. Пасічник, Д. І. Угрин. – Львів : Магнолія 2006, 2018. – Кн. 1. – 336 с.
3. Веб-технології та веб-дизайн : навчальний посібник / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда. – Одеса : Фенікс, 2019. – 284 с.
4. Довідкове керівництво по html/css [Електронний ресурс]. – Режим доступу : https://css.in.ua/.
5. Електронний посібник по HTML, CSS та PHP [Електронний ресурс]. – Режим доступу : https://www.w3schools.com/.
6. Довідкове керівництво по WordPress [Електронний ресурс]. – Режим доступу : https://uk.wordpress.org/.
7. Створення карти сайту в xml [Електронний ресурс]. – Режим доступу : https://www.xml-sitemaps.com/.
8. Онлайн-редактор для HTML/CSS/JavaScript [Електронний ресурс]. – Режим доступу : https://liveweave.com/.

Додаток А Глобальні атрибути

Атрибут	Опис, значення
accesskey	Генерує поєднання клавіш для доступу до поточного елемента. Складається з розділеного пробілами списку символів. Браузер у першу чергу вибиратиме ті клавіші, які є на клавіатурній розкладці. Застосовується до наступних елементів: <a>, <area>, <button>, <input>, <label>, <legend>, <textarea>. Можливі значення: перелік клавіш.
class	Визначає ім'я класу для елемента (використовується для визначення класу у таблиці стилів). Можливі значення: ім'я класу.
contenteditable	Визначає можливість редагування контента для користувача. Дозволяє перетворити будь-яке HTML-поле в елемент для редагування. Можливі значення: true/false.
dir	Визначає напрямок тексту контента в елементах <bdo> та <bdi>. Можливі значення: ltr/rtl/auto.
draggable	Надає користувачу можливість перемістити елемент. Можливі значення: true/false/auto.
hidden	Вказує на те, що елемент повинен бути прихований. Можливі значення: hidden.
id	Визначає унікальний ідентифікатор елемента. Можливі значення: id – ідентифікатор елемента.

Продовження додатка А

lang	Визначає код мови контенту в елементі. Можливі значення: код мови.
spellcheck	Вказує на необхідність перевірки контента елемента перевірці на орфографію та граматику. Можливі значення: true/false.
style	Вказує на код CSS, що застосовується для оформлення елемента. Можливі значення: код CSS.
tabindex	Визначає порядок переходу до елемента за допомогою клавіші TAB. Можливі значення: порядковий номер.
title	Визначає додаткову інформацію про елемент у вигляді спливаючої підказки. Можливі значення: текст.
translate	Дозволяє або забороняє переклад тексту всередині елемента. Можливі значення: yes/no.

Електронне навчальне видання

Методичні вказівки

до практичних і лабораторних занять
із курсу «**Інформаційні та вебтехнології**»
для студентів спеціальності 171 «*Електроніка*»
освітнього ступеня «бакалавр»
денної форми навчання

У двох частинах

ЧАСТИНА 1

Статичні вебтехнології

Відповідальний за випуск І. Ю. Проценко
Редактор О. В. Федяй
Комп'ютерне верстання Ю. М. Шабельника

Формат 60×84/16. Ум. друк. арк. 3,48. Обл.-вид. арк. 3,25.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.