М. О. Данова, В. В. Сергієнко

WEB-ДИЗАЙН

Частина 1

2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

М. О. Данова, В. В. Сергієнко

WEB-ДИЗАЙН

Частина 1

Навчальний посібник

2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

М. О. Данова, В. В. Сергієнко

WEB-ДИЗАЙН

Частина 1

Навчальний посібник

2019

УДК 004.774.6(075.8)
Д18

Рецензенти : д-р техн. наук, проф. С. Ф. Чалий,
д-р техн. наук, проф. Н. В. Шаронова

Данова, М. О.

Д-18 Web-дизайн [Електронний ресурс] : навч. посіб. / М. О. Данова, В. В. Сергієнко. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2019. – 87 с.

Наведено необхідні теоретичні відомості про розміщення й передання інформації в інтернеті, концепції й можливості найбільш популярних технологій сучасного web-дизайну. Розглянуто принципи роботи в мережі інтернет та основи створення статичних сайтів з допомогою HTML і CSS.

Для початківців-програмістів, студентів спеціальності «Інженерія програмного забезпечення» при виконанні практичних робіт.

Іл. 30. Табл. 12. Бібліогр. : 36 назв

УДК 004.774.6(075.8)

ВСТУП

Сучасний інтернет – це дуже складна й високотехнологічна система, що дає користувачеві змогу спілкуватися з людьми, які знаходяться в будь-якій точці земної кулі, швидко й комфортно відшукувати будь-яку необхідну інформацію, публікувати дані, котрі він хотів би повідомити всьому світу.

Інтернет-технології стрімко розвиваються, проникаючи в найрізноманітніші сфери професійної діяльності, у тому числі й економічну. Для компаній присутність в інтернеті – це можливість розповісти про свої товари й послуги, знайти потенційних партнерів і клієнтів, а також зменшити витрати завдяки інтернет-торгівлі, використанню «хмарних» сервісів. Навіть такі традиційно замкнені системи, як промислові автоматизовані системи керування виробництвом, у тому числі й у критичних галузях, також у більшості випадків прямо або побічно підключено до інтернету.

Рядові користувачі часто звертаються до інтернет-магазинів, інтернет-банкінгу, спілкуються в соціальних мережах, можуть отримувати через інтернет державні послуги, довіряючи інтернет-системам свої персональні дані й іншу конфіденційну інформацію.

Разом з тим нові можливості породжують і нові загрози безпеки, які не завжди достатньою мірою усвідомлюються як рядовими користувачами, так і власниками ресурсів. Для компаній найбільш тривожною є тенденція збільшення атак на корпоративні сайти й web-додатки. Це є наслідком того, що більшість компаній недооцінюють небезпеку подібних атак, а розробники web-додатків недостатньо ретельно стежать за безпекою власних продуктів. Для розгляду питань, що стосуються загроз безпеки, вразливостей і методів захисту web-додатків, необхідно мати базові навички використання основних web-технологій.

1 БАЗОВІ ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ WEB-СТОРІНОК

1.1 Основні терміни й означення

Інтернет (англ. internet) – всесвітня система об'єднаних комп'ютерних мереж, побудована на використанні протоколу TCP/IP і маршрутизації пакетів даних. Інтернет утворює глобальний інформаційний простір, є фізичною основою для Всесвітньої павутини (WWW, World Wide Web) і безлічі інших систем (протоколів) передання даних. Часто згадується як «Всесвітня мережа» і «Глобальна мережа», у побуті іноді вживають скорочене найменування «Інет».

Іншими словами, інтернет складається з безлічі домашніх і корпоративних мереж, що належать різним користувачам, компаніям і підприємствам, які працюють з найрізноманітнішими протоколами, пов'язаними між собою різними лініями зв'язку, що можуть передавати дані по телефонних проводах, оптоволокну, через супутники й радіомодеми.

Структура інтернету нагадує павутину, у вузлах якої знаходяться комп'ютери. Кожен комп'ютер має свою адресу, яку називають IP-адресою, що обов'язково має бути унікальною, так само як у будь-якого об'єкта. Існують два типи IP-адрес:

- постійні, тобто закріплені за певним комп'ютером;
- динамічні, які присвоюються в той момент, коли користувач з'єднується з інтернетом.

Структура IP-адреси влаштована таким чином, що дає змогу дізнатися, у якій країні і в якому місті знаходиться комп'ютер користувача. Ця інформація може бути корисною, наприклад, під час настроєння оголошень за контекстною рекламою з урахуванням регіону. Приклад IP-адреси: 192.168.1.2. Для того щоб дізнатися свою IP-адресу, необхідно зайти в «Панель управления» -– «Сетевые подключения» – вибрати «Подключения по локальной сети» і перейти на вкладку «Поддержка».

Для того щоб не запам'ятовувати складні цифри IP-адреси, було введено доменні імена. *Доменне ім'я* – це унікальне ім'я, яке певний постачальник послуг вибрав собі для ідентифікації, наприклад: *ukr.net* або *google.com*

Доменне ім'я може мати кілька рівнів. Домен першого рівня зазвичай визначає країну розташування сервера (*ua* – Україна; *uk* – Великобританія; *de* – Німеччина) або вид організації (*com* – комерційні організації; *edu* – наукові й навчальні організації; *gov* – урядові установи; *org* – некомерційні організації).

Приклади доменних імен:

facebook.com, books.ua – домени другого рівня;

gazeta.ukr.net – домен третього рівня.

Сайт – це сукупність електронних документів (веб-сторінок), об'єднаних під однією адресою (доменним ім'ям) і пов'язаних між собою посиланнями. Доступ до сайта здійснюється через браузер.

Сьогодні існує шість основних браузерів:

- Google Chrome (Webkit/Blink);
- Mozilla Firefox (Gecko, SpiderMonkey, Webkit for IOS);
- Internet Explorer (Trident);
- Safari (Webkit);
- Microsoft Edge (EdgeHTML, Chakra for JavaScript);
- Opera (Blink, V8 for JavaScript).

Усі вони майже однаково відображають інформацію, за винятком Internet Explorer, який деякі теги й властивості стилів відображає по-іншому, наприклад ширину блоків, і ці особливості необхідно враховувати під час верстання.

Сайти можна поділити на такі види.

1 *За вмістом*:

- статичні – вміст готується заздалегідь і видається користувачеві в тому вигляді, у якому зберігається на сервері;
- динамічні – вміст генерується з допомогою серверних мов програмування.

2 *За схемою подання інформації:*

- комерційні – сайти компаній, інтернет-магазини і т. д.;
- інформаційні – повідомляють користувачеві будь-яку інформацію;
- web-сервіси (портали) – пошукові системи, електронна пошта, форуми, соціальні мережі.

3 *За розміром:*

- фіксованої ширини – задається фіксована ширина сайта (сьогодні ширина зазвичай становить 1000 пікселів, тому що роздільна здатність сучасних моніторів за шириною починається від 1024 пікселів). Тоді всі блоки сайта не будуть змінювати розмір залежно від роздільної здатності монітора або розміру екрана браузера;
- «гумові» – ширина сторінок сайта точно не задається, а змінює розмір залежно від розміру екрана браузера або роздільної здатності монітора.

Процес розроблення сайта зазвичай складається з кількох послідовних етапів. На рисунку 1.1 показано процес розроблення сайта на прикладі веб-студії, де певні функції виконує певна людина.

Коли замовник приходить зі своєю ідеєю сайта до веб-студії, з ним починає працювати менеджер проекту. Дуже часто трапляється, що ідея у замовника є, але настільки розпливчата, що важко уявити на цьому етапі, який результат хоче отримати клієнт. Звичайно, у різних організаціях

використовується свій підхід до з'ясування потреб замовника, але як один із варіантів – пропонування заповнити спеціальну анкету, після чого можна вже уявити, які вигляд і функціонал буде мати майбутній сайт. Далі менеджер складає два технічних завдання, одне – для дизайнера на макет сайта (дизайну), а друге – для програміста із зазначенням функціоналу, який необхідно реалізувати в проекті. Після цього верстальнику відсилається макет сайта від дизайнера і різні модулі від програміста, і після того як сайт буде зверстано, копірайтер наповнює його контентом, тобто заповнює товарами, пише статті і т. д., і водночас seo-фахівець починає просувати сайт у мережі інтернет.

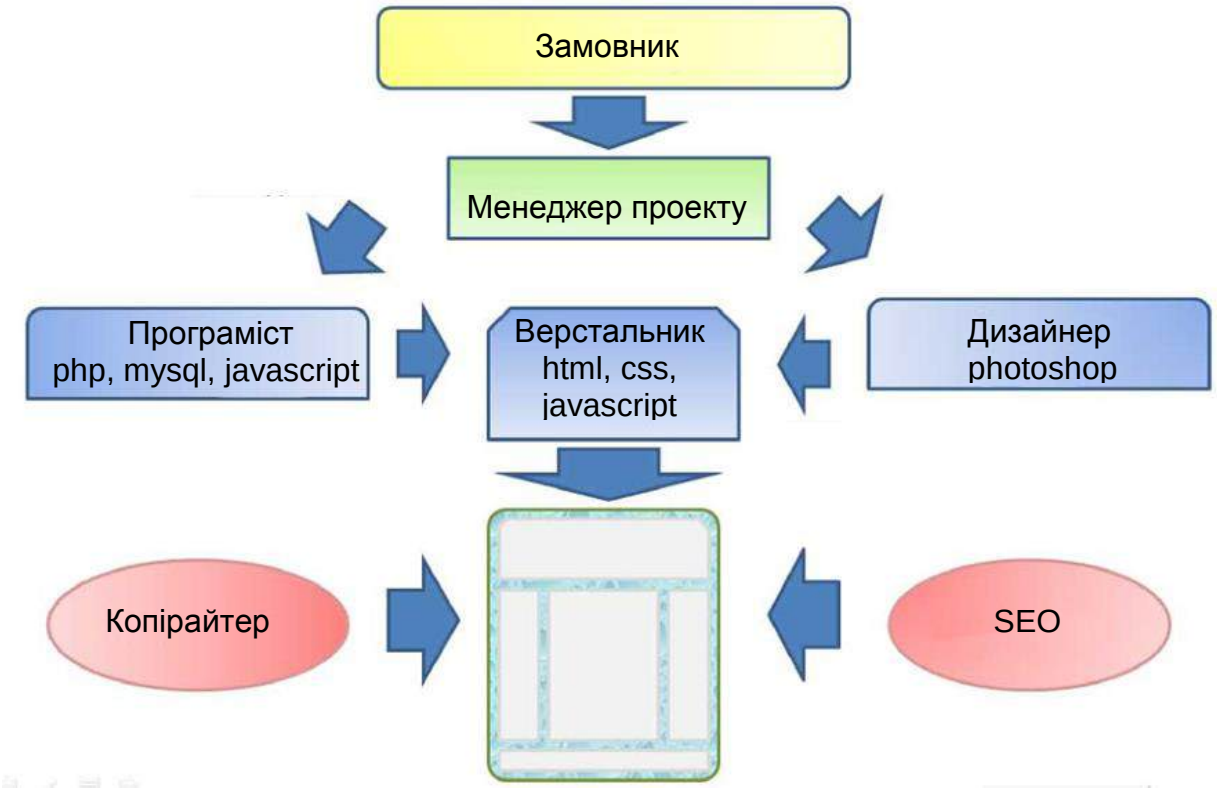


Рисунок 1.1 – Приклад процесу розроблення сайта

Для обміну даними або ж взаємодії таких компонентів, як комп'ютери, сервери, маршрутизатори, комутатори тощо, використовують так звані протоколи. Кожен **протокол** – це чіткий і певний набір правил і угод, у якому зазначалося б, яким чином обмінювати й обробляти інформацію.

Найбільш відомі протоколи, які використовують у мережі інтернет:

- HTTP (Hyper Text Transfer Protocol) – протокол передання гіпертексту, який використовується під час пересилання web-сторінок з одного комп'ютера на інший;
- HTTPS (Hyper Text Transfer Protocol Secure) – протокол для

передання гіпертексту, але в ньому використано додаткове шифрування даних для більш безпечного передання інформації;

- FTP (File Transfer Protocol) – протокол передання файлів зі спеціального файлового сервера на комп'ютер користувача. FTP дає змогу абоненту обмінюватися двійковими й текстовими файлами з будь-яким комп'ютером мережі. Установивши зв'язок з віддаленим комп'ютером, користувач може скопіювати файл з віддаленого комп'ютера на свій або зі свого комп'ютера на віддалений.

Схему HTTP-запиту web-сторінки зображено на рисунку 1.2.

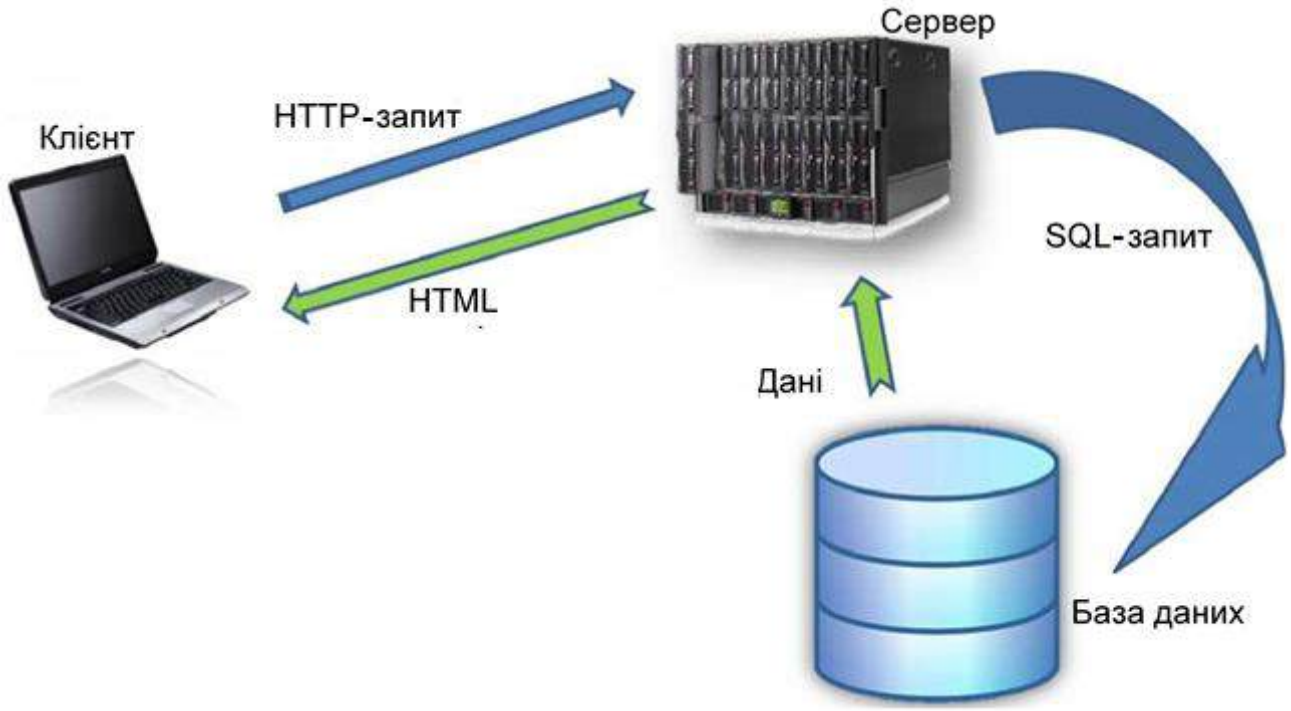


Рисунок 1.2 – Схема HTTP-запиту web-сторінки

Користувач набирає в браузері адресу необхідного йому сайта, після чого надсилає HTTP-запит серверу. Сервер визначає тип сторінки з розширення файла. Якщо розширення *.html*, то ця сторінка статична, і він відразу формує HTML-сторінку і відправляє її назад користувачеві. Якщо розширенням є, наприклад, *.php*, то в цьому випадку сторінка вже динамічна, і сервер підмикає інтерпретатор PHP, який у свою чергу може звернутися до бази даних і отримати звідти необхідні дані. Після цього сервер також формує зміст HTML-сторінки і відправляє її користувачеві, який бачить цю сторінку в браузері.

Web-сторінка – це текстовий файл, що містить власне текст сторінки і команди форматування, які називають тегами (Tag), або дескрипторами розмітки мови HTML. Мова розмітки гіпертексту HTML (HyperText Markup

Language) є базовою технологією розроблення web-сторінок, які називають також HTML-документами.

Гіпертекст – термін, що ввів Тед Нельсон 1965 року для позначення «тексту, що розгалужується або виконує дії за запитом». Зазвичай гіпертекст подається набором текстів, що містять вузли переходу між ними, які дають змогу вибирати матеріал для читання або послідовність читання. Загальновідомим і яскраво вираженим прикладом гіпертексту є web-сторінки – документи HTML, розміщені в мережі інтернет. У більш широкому розумінні гіпертекстом є будь-яка повість, словник або енциклопедія, де є відсилання до інших частин цього тексту, що стосуються певного терміну. У комп'ютерній термінології гіпертекст – це текст, сформований з допомогою мови розмічення, який потенційно містить гіперпосилання.

HTML – це система верстання web-сторінок, що визначає, які елементи і як мають розташовуватися в документі. Код HTML інтерпретується браузером (засобом перегляду web-сторінок), який виводить відформатований вміст web-сторінки на екран. Перегляд коду завантаженої сторінки є доступним у всіх браузерах.

Файли web-сторінок мають розширення .htm, .html або інші, якщо при їх створенні було використано спеціальні серверні технології (наприклад, SHTML, ASP, PHP). Тип HTML-файлів походить від назви мови створення web-сторінок HTML, тому web-сторінки часто називають HTML-документами.

HTML-документ – звичайний текстовий файл, який може містити текст, теги й стилі. Зображення й інші об'єкти зберігаються окремо. Уміст такого файла зазвичай називають HTML-кодом.

HTML дає змогу формувати на сторінці сайта текстові блоки, додавати в них зображення, організовувати таблиці, керувати відображенням кольору документа й задавати форматування тексту, додавати звуковий супровід, організовувати гіперпосилання для переходу до інших розділів сайта, інтернет-ресурсів, а також задавати взаємне розташування текстових блоків та інших компонентів сторінки.

Іноді HTML помилково називають мовою програмування, але це не так, тому що в HTML немає головного атрибута, властивого будь-якій мові програмування, – команди. На HTML не можна задати послідовність дій, а можна тільки описати, як браузер має виводити на екран той чи інший документ. Якщо ж на web-сторінці дійсно має щось виконуватися, наприклад вестися форум, то використовуються справжні мови програмування, такі як JavaScript.

HTML-документ містить текст і команди мови HTML, які називають дескрипторами розмічення, або тегами. **Тег** (tag) – це спеціальний символ розмічення, який застосовується для вставлення різних елементів на web-сторінку й для змінення їх вигляду. Для позначення тегів

використовується символ <тег>:

```
<тег атрибут1="значення" атрибут2="значення">
<тег атрибут1="значення" атрибут2="значення">... </тег>
```

Теги нечутливі до регістра, тому записи <B> і <b> є еквівалентними. Теги HTML визначають:

- зовнішній вигляд документа (формат шрифту, колір фону і т. д.);
- структуру документа (взаємне розташування текстової, графічної та іншої мультимедійної інформації);
- посилання на інші інтернет-ресурси.

Основні теги наведено в таблиці 1.1.

Таблиця 1.1 – Основні теги HTML

Тег	Опис
<html>	Визначає документ HTML
<body>	Визначає основну частину або тіло документа
<h1> -- <h6>	Визначає заголовки с 1-го по 6-й рівні
<p>	Визначає параграф
 	Вставляє одиничне перенесення рядка
<hr>	Визначає горизонтальну лінію
<!-->	Визначає коментар

Атрибут – додаткова інформація певного тега. Атрибути завжди записують усередині тега, потім іде знак рівності й деталі атрибута, узяті в подвійні лапки. Наприклад, заголовок 2-го рівня <h2> буде вирівняно відносно вікна браузера по правий бік, тому що атрибут align задано значенням "right":

```
<h2 align="right"> Заголовок другого рівня </h2>
```

Порядок тегів. Існує певна ієрархія вкладеності тегів. Наприклад, тег <title> має знаходитись усередині контейнера <Head>. Щоб не виникло помилки, необхідно стежити за тим, чи правильно розташовано теги в коді. Якщо теги є рівноцінними в ієрархії зв'язку, то їх послідовність не має значення. Так, можна поміняти місцями теги <title> і <meta>, на кінцевому результаті це ніяк не позначиться.

Типи тегів. Кожен тег HTML належить до певної групи, наприклад, табличні теги спрямовано на формування таблиць і не можуть застосовуватися для інших цілей. Умовно теги поділяють на такі типи:

- теги верхнього рівня;
- теги заголовка документа;
- блокові елементи;
- вбудовані елементи;
- універсальні елементи;
- списки;
- таблиці.

Слід ураховувати, що один і той самий тег може одночасно належати до різних груп, наприклад, теги `<ol>` і `<ul>` належать до категорії списків, але також є й блоковими елементами.

Контейнером називають парний тег, усередині якого можуть розташовуватись інші теги. Контейнер потребує закривального тега, який позначають `</тег>`. Таким чином, контейнер складається з відкривального `(<тег>)` і закривального `(</тег>)` тегів.

Слід запам'ятати одне просте правило вкладених тегів: закривати теги необхідно саме в тій послідовності, у якій їх відкривали (рисунок 1.3). Оскільки тег `<b>` було відкрито останнім, то закривати його слід першим, а далі вже всі інші теги по черзі. Необхідно стежити за тим, щоб теги відкривання й закривання знаходилися на одному рівні.

Рисунок 1.3 – Вкладені теги

Посилання (гіперпосилання) – є основою мережних документів і дають змогу переходити з однієї web-сторінки на іншу. Їх особливість полягає в тому, що саме посилання може вказувати не тільки на HTML-файли, але й на файл будь-якого типу, причому цей файл може розміщатися зовсім на іншому сайті. Головне, щоб до файла, на який робиться посилання, був доступ.

Медійні об'єкти (графіка, відео і т. д.), вбудовані у web-сторінку з допомогою тегів, зберігаються в окремих файлах відповідних типів.

Будь-який web-браузер містить інтерпретатор мови HTML, що допомагає йому коректно відображати web-сторінку разом з усім її вмістом на екрані.

Для створення простої web-сторінки необхідні такі інструменти:

- текстовий редактор – вбудований у Windows блокнот або більш функціональний і безкоштовний текстовий редактор Notepad++

(завантажити останню версію редактора з офіційного сайта можна за таким посиланням: http://notepad-plus-plus.org);

- інтернет-браузер;
- графічний редактор (Photoshop або Gimp) – для «нарізки» макета сайта, і також для підбирання необхідних кольорів.

1.2 Структура web-документа

Web-сторінка сайта може складатися з різних блоків (рисунок 1.4):

- шапка (header), у якій можуть розміщуватися логотип компанії, назва сайта, телефони організації та ін.;
- вертикальна і/або горизонтальна навігація на сайті (меню);
- основний вміст (content);
- рекламні банери й посилання на інші сайти;
- додаткові модулі, наприклад голосування, опитування, корзина покупця й т. ін.;
- нижній колонтитул (footer), де розміщується додаткова інформація: авторський знак, лічильники відвідуваності та ін.

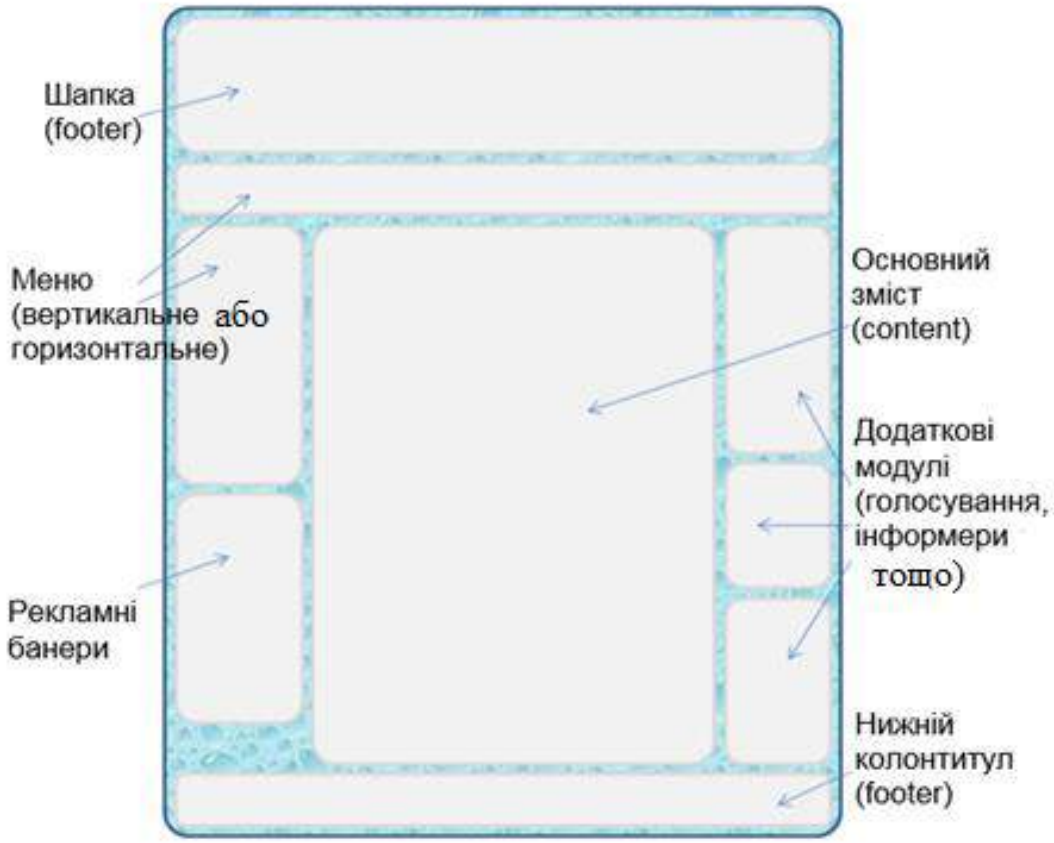


Рисунок 1.4 – Приклад структури типової web-сторінки сайта

Якщо відкрити будь-яку web-сторінку, то в ній будуть міститися типові елементи (рисунок 1.5), які залишаються вихідними незалежно від виду й напрямку сайта:

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="win-1251">
      <title>Перша сторінка</title>
 </head>
 <body>
      Зміст сторінки
 </body>
</html>
```

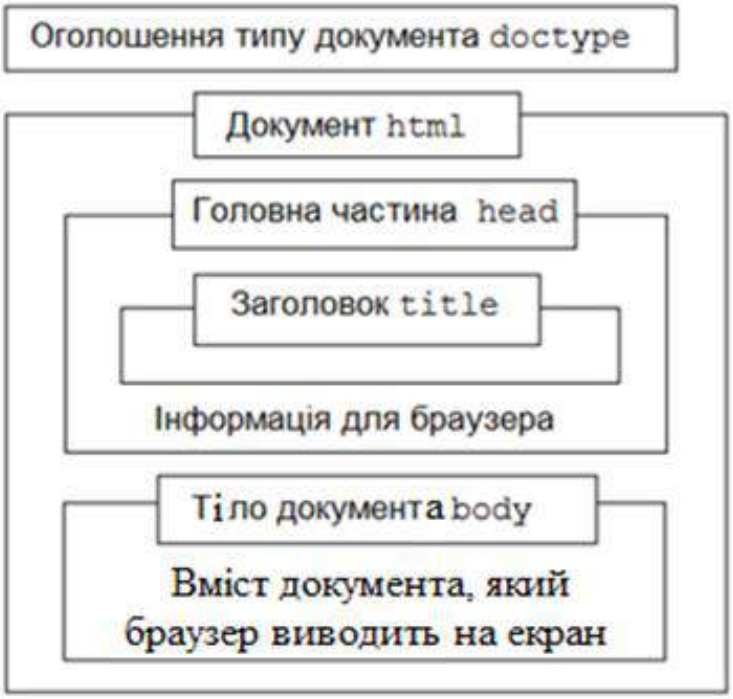


Рисунок 1.5 – Структура HTML-документа

Елемент `!DOCTYPE` призначено для вказівки типу поточного документа (DTD (document type definition)). Існує кілька видів `<!DOCTYPE>`, які різняться залежно від версії HTML.

Тег `<html>` визначає початок HTML-документа, усередині якого містяться блоки `<head>` і `<body>`.

У блоці `<head>` розміщується службова інформація, наприклад: оголошуються теги `<meta>`, які можуть мати різне призначення (кодування, опис сторінки, ключові слова); теги, які під'єднують файли каскадних таблиць стилів і файли скриптів. Усе, що знаходиться в цьому блоці, не

видно користувачеві, за винятком вмісту тега <title>, у якому вказується назва веб-сторінки. Обов'язково вказувати назву кожної сторінки у тегу <title>, наприклад: «Мегапобут – інтернет-магазин побутової техніки | Каталог товарів | Мийний пилосос Sharp».

Обов'язково слід додавати закривний тег </head>, щоб показати, що цей блок завершено. Тіло HTML-документа, а саме тег <body>, призначено для розміщення тегів і змістовної частини. Усередині блоку <body> можуть розміуватися будь-які елементи web-сторінки. Слід додати закривний тег </body>, щоб показати, що тіло документа завершено. Останнім елементом у коді є закривальний тег </html>.

Практичні завдання

Створити просту html-сторінку, для чого:

- створити на будь-якому з жорстких дисків нову папку, назвати її, наприклад, «Sites»;
- створити текстовий документ з розширенням «.txt»;
- перейменувати цей файл, наприклад на «test», і змінити розширення на «.html»;
- відкрити файл з допомогою редактора Notepad++;
- написати перше привітання світу, перейти в текстовий редактор і набрати

```
<h1>Hello world!</h1>
<p>I'm here</p>
```

- відкрити цей файл у браузері для перегляду: клацнути по файлу або вибрати пункт «Запуск в Notepad++», далі – «launch in (браузер)».

Для переміщення між різними об'єктами слід використовувати клавіатуру комп'ютера і знати кілька комбінацій клавіш:

- Alt + Tab – переміщення між відкритими вікнами;
- Ctrl + Z – скасувати дію;
- Ctrl + Y – повернути дію;
- Ctrl + X – вирізати;
- Ctrl + C – копіювати;
- Ctrl + V – вставити;
- Ctrl + S – зберегти;
- Ctrl + A – виділити все;
- Ctrl + Tab – переміщення між відкритими вкладками;
- F5 – оновити сторінку;
- F2 – перейменувати;
- Shift + Tab – повернути одну табуляцію назад;

- Shift + Home – виділити із зазначеного місця до початку рядка;
- Shift + End – виділити із зазначеного місця до кінця рядка;
- Shift + стрілка – при затиснутій кнопці Shift виділяти в зазначену стрілками сторону.

Домашнє завдання для закріплення матеріалу

1 Установити наведені вище інструменти (Notepad++, Photoshop або Gimp).

2 Створити на жорсткому диску папку, де будуть зберігатися сайти.

3 Створити html-документ і написати в ньому:

Мірошник і хлопчик сідають удвох,
Віслюк, що є сили, везе їх обох.
Сміх перехожого вітер несе:
«Діда й онука скотина везе!»

4 Знайти й скачати в інтернеті будь-який макет сайта у форматі .psd. Виділити в ньому такі частини: шапка, меню, контент, підвал і т. д.

5 Вивчити комбінації гарячих клавіш.

2 ФОРМАТУВАННЯ ТЕКСТУ І СИМВОЛІВ У HTML

2.1 Форматування тексту

Теги форматування можуть розбивати текст на блоки (абзаци, списки, таблиці) або визначати написання символів тексту всередині блоку. Тому всі теги поділяються на теги рівня блоку й теги рівня тексту. З огляду на коректність написання HTML-коду важливо, щоб теги рівня тексту обов'язково вказувалися всередині якого-небудь блоку.

Основні теги HTML, які використовуються для форматування абзаців, наведено в таблиці 2.1.

Таблиця 2.1 – Основні теги для форматування абзаців

Тег	Опис
p	Абзац <p>, відокремлений від іншого тексту порожніми рядками є тегом рівня блоку, однак не може містити елементи рівня блоку (включаючи сам тег <p>). Не рекомендується використовувати порожні елементи <p>
H1 – H6	Шість рівнів заголовків: від H1 (найверхній) до H6 (найнижчий). Візуально браузери зазвичай відображають значніший заголовок більшим шрифтом.

Продовження таблиці 2.1

Тег	Опис
H1 – H6	Елементи заголовків є частиною глобальної структури документа, їх призначено для стислого опису вмісту наступного розділу. Теги заголовків можуть зчитуватись автоматичними засобами подання документа для створення автоматичного змісту. Важливо дотримуватися правильного порядку рівнів заголовків для опису розділів і підрозділів з метою їх коректного оброблення при автоматичному зчитуванні
BR	Непарний тег, примусово розриває (закінчує) поточний рядок тексту
DIV	Абзац, який не має ніякого додаткового форматування. Зазвичай використовується для групування блоків
SPAN	Використовується для виділення частини інформації й надання їй різних стилів

Атрибут ALIGN визначає спосіб горизонтального вирівнювання для тегів P, H1 – H6, DIV. Можливі значення: left, center, right, justify (відповідно вирівнювання по лівому краю, по центру, по правому краю і по ширині). За замовчуванням має значення left. Приклади використання тегів для форматування абзаців:

```
    <html>
    <meta charset="utf-8">
    <body>
    <P align = "center"> Цей параграф по центру </P>
    <P align = "left"> Цей по лівому краю </P>
    <P> Цей теж по лівому (значення за замовчуванням) </P>
    <P align = "right"> Цей по правому краю </P>
    <P align = "justify"> У цьому параграфі текст буде вирівнюватися
по ширині (одночасно по лівому і правому краях документа)</P>
    <! – приклади заголовків від 1-го до 6-го рівня ->
    <H1 align = "center"> Заголовок 1-го рівня </h1>
    <H2> Заголовок 2-го рівня </ h2>
    <H3 align = "right"> Заголовок 3-го рівня </h3>
    <H4> Заголовок 4-го рівня </h4>
    <H5> Заголовок 5-го рівня </h5>
    <H6> Заголовок 6-го рівня </h6>
    <! – використання Div ->
    <Div align = "right">
    Починається перший рядок <br> потім іде другий <br>
    </Div>
    а ось вже і третій
    <P>
```

```
    А це – параграф, і всередині нього теж можна використовувати тег
<br> перенесення рядка. Його треба використовувати завжди, коли треба
перенести рядок </p>
    Хочете користуватися <span style="color:red;" > Adobe Dreamweaver
</span>, тоді вивчіть хоча б основи html і <br> Ви без проблем
розберетеся з цією чудовою програмою!
    </body>
    </html>
```

Дуже важливо слідкувати, щоб вміст у лапках було написано без пробілів, тобто `<p align = "right">`, а не `<p align = " right ">` – інакше буде помилка.

Тег, або параметр, що задає відступ першого рядка абзацу (червоний рядок), не визначено. Новий рядок можна задати з допомогою таблиць стилів.

Ширина блоків автоматично підстроюється під розмір вікна браузера (так само, як абзаци Word підстроюються під ширину документа завдяки автоматичному перенесенню на новий рядок).

2.2 Форматування символів

Теги рівня тексту використовуються для задання специфічного написання послідовності символів (указання тексту в лапках, верхні/нижні індекси) або параметрів шрифту (виділення жирністю, курсивом, великий, малий шрифт). Разом з тим для форматування шрифтів кращим уважається використання таблиць стилів.

Основні теги форматування символів наведено в таблиці 2.2.

Таблиця 2.2 – Основні теги форматування символів

Тег	Опис
STRONG	Використовується для виділення тексту жирністю
EM	Використовується для виділення тексту курсивом
U	Виділення тексту підкресленням
S	Виводить текст перекресленим
BIG	Виводить текст збільшеним порівняно зі стандартним
SMALL	Виводить текст зменшеним порівняно зі стандартним
SUP	Створення верхнього індексу
SUB	Створення нижнього індексу

Продовження таблиці 2.2

Тег	Опис
FONT	Змінення кольору, типу, розміру шрифту. Атрибути: SIZE – визначає розмір шрифту; можливі значення – 1, 2, 3, 4, 5, 6, 7 COLOR – визначає колір тексту; задається або RGB-значенням у шістнадцятковій системі, або одним із 16 базових кольорів FACE – визначає шрифт, який використовується
HR	Непарний тег, уставлення горизонтальної лінії. Атрибути: WIDTH – визначає довжину лінії в пікселах або відсотках від ширини вікна браузера SIZE – товщина лінії в пікселах. ALIGN – вирівнювання горизонтальної лінії. Атрибут може набувати таких значень: left – вирівнювання по лівому краю документа; right – вирівнювання по правому краю документа; center – вирівнювання по центру документа (використовується за замовчуванням) NOSHADE – визначає спосіб зафарбовування лінії як суцільний. Атрибут є прапором і не потребує вказання значення, без цього атрибута лінія відображається об'ємною COLOR – задає колір лінії; можна використовувати або RGB-значення в шістнадцятковій системі, або один із 16 базових кольорів

Приклади використання тегів із таблиці 2.2:

```
<html>
<meta charset="utf-8">
<body>
<p><strong> Цей параграф відформатовано тегом strong </strong></p>
<em> Цей текст буде виділено курсивом </em> <br><br>
<u> Цей текст буде підкреслено!!! </u><br><br>
Увага акція! нова ціна <s> 50 000 </s> 20 000 гривень!
<P> <big> Цей параграф відформатовано тегом big </big> </p>
<P> <small> Цей параграф відформатовано тегом small </small> </p>
<P> А в цьому параграфі теги не застосовуються </p>
2 <sup> 2 </sup> = 4 ;<br>
2 <sup> 3 </sup> = 8 ; <br>
2 <sup> 4 </sup> = 16 ;<br>
<br>
Формула води в хімії – H <sub> 2 </sub> O <br>
<br>
```

```
    Це звичайний текст <br>
    <FONT SIZE = "+ 2" COLOR = "red"> Збільшений червоний шрифт </FONT>
    <br>
    <FONT SIZE = "3" FACE = "Courier New" COLOR = "Violet">
Моноширинний фіолетовий текст 3-го розміру </FONT><br>
    <br>
    Текст до лінії <hr noshade width = "50%"
    align = "left"> Після лінії <br>
    А ось приклад лінії завтовшки 2px і без прапорця noshade
    <hr width = "50%" align = "left" size = "2">
    </body>
    </html>
```

Деякі символи, такі як "<", мають у HTML спеціальне значення, тому їх не можна використовувати в тексті в явному вигляді. Для їх відображення використовуються символьні елементи CER (Character Entity Reference).

Для подання символу "<" в тілі документа HTML використовується `<`, а для символу ">" – `>`. Текст переноситься по словах. Додаткову можливість перенесення можна задати, використовуючи символ «м'якого дефіса» `&shy;` (або `&#xAD;`). Якщо, навпаки, необхідно заборонити розрив рядка в якомусь місці, то можна використовувати символ нерозривного пробілу ` ` (символи-мнемоніки або ` `).

Символьний елемент відображається у вигляді

```
&Імя_символу; або      &#Номер_символу;
```

Символьні об'єкти, які найбільш часто використовують, наведено в таблиці 2.3.

Таблиця 2.3 – Символьні об'єкти, які найчастіше використовуються

Результат	Опис	Ім'я об'єкта	Номер об'єкта
	Нерозривний пробіл		
<	Менше	&lt;	&#60;
>	Більше	&gt;	&#62;
&	Амперсанд	&amp;	&#38;
"	Лапки	&quot;	&#34;
'	Апостроф	&apos;	&#39;
£	Фунт стерлінгів	&pound;	&#163;
¥	Єна	&yen;	&#165;
§	Параграф	&sect;	&#167;
©	Авторське Право	&copy;	&#169;
®	Зареєстрована торгова марка	&reg;	&#174;
x	Множення	&times;	&#215;
÷	Ділення	&divide;	&#247;

2.3 Форматування списків

Списки в html-документах поділяються на марковані й нумеровані. Різняться вони тим, що в маркованих елемент списку починається з маркерів, а в нумерованих – із цифр або букв (таблиця 2.4).

Таблиця 2.4 – Теги для форматування списків

Тег	Опис
UL	Створює маркований список
OL	Створює нумерований список. Атрибути: START – визначає перше число, з якого починається нумерація пунктів (тільки цілі числа). Якщо її не вказувати, то слід починати нумерувати з початку. TYPE – визначає стиль нумерації пунктів списку. Можливі значення: "A" – великі літери A, B, C ... "a" – малі літери a, b, c ... "I" – великі римські числа I, II, III ... "i" – маленькі римські числа i, ii, iii ... "1" – арабські цифри 1, 2, 3 ...
LI	Створює пункти списку всередині елементів OL або UL. Атрибути: VALUE – змінює порядок нумерації елементів списку. Використовується, тільки якщо елемент LI знаходиться всередині елемента OL. Як значення вказується порядковий номер елемента

Приклади списків:

```
<html>
<meta charset="utf-8">
<body>

<Ul>
<Li> Перший пункт списку </li>
<Li> Другий пункт списку </li>
<Li> Третій пункт списку </li>
<Li> Четвертий пункт списку </li>
</Ul>

<! – приклад списку з арабськими числами ->
Щоб створити сайт на домашньому комп'ютері, необхідно:
<Ol type = "1">
```

```
<Li> Вивчити html </li>
<Li> Вивчити css </li>
<Li> Ознайомитися з php </li>
</Ol>
<! – приклад списку з великими римськими числами ->
Щоб створювати сайти швидко, бажано:
<Ol type = "I">
<Li> знання програми Adobe Dreamweaver </li>
<Li> Знання Photoshop </li>
</Ol>
<! – приклад списку з маленькими буквами ->
Щоб заробляти на сайті, необхідно:
<Ol type = "a">
<Li> наявність хорошої відвідуваності </li>
<Li> наявність цифрового продукту для продажу </li>
<Li> наявність блоків реклами </li>
</Ol>
<! – приклад списку не з початку ->
Щоб стати успішним в інтернеті, необхідно:
<Ol type = "1" start = "2">
<Li> писати інформацію, корисну для людей </li>
<Li> постійно поповнювати сайт новим вмістом </li>
<Li> перший пункт придумайте самі </li>
</Ol>

Щоб створити сайт на домашньому комп'ютері, необхідно:
<Ol> <! – оскільки тип не вказано, буде використовуватися за
замовчуванням ->
<Li> Вивчити html </li>
<Em> зараз нумерація піде з п'ятого номера </em>
<Li value = "5"> Вивчити css </li>
<Li> Ознайомитися з php </li>
</Ol>

</body>
</html>
```

2.4 Вставлення посилань

Гіперпосилання можуть як зв'язувати сторінки в межах одного сайта, так і вказувати на будь-яку сторінку в інтернеті. Для створення посилання необхідно повідомити браузеру, що є посиланням, а також указати адресу документа, на який слід зробити посилання. Обидві дії виконуються з допомогою тега `<a>`. Загальний синтаксис створення посилань є таким:

```
<a href="file.html"> текст посилання, який бачить користувач </a>
```

Атрибут `href` (скорочення від англ. hypertext reference – гіпертекстове посилання), містить місце, на яке виконується перехід за цим посиланням,

зазвичай це internet-адреса та/або ім'я файла з обов'язковим атрибутом `href`, у значенні якого вказується на файл або сторінку іншого сайта або на місце в документі, куди слід перейти. У цьому тегу необхідно написати той текст, який побачить користувач. За замовчуванням у браузері текст посилання буде синього кольору і підкресленим.

Посилання можуть бути абсолютними й відносними.

Абсолютні посилання використовують для переходу на сторінки зовнішнього сайта. Для цього в значенні атрибута `href` слід указати повний шлях до тієї сторінки, на яку хочемо перейти, включаючи тип протоколу:

```
<a href="http://html-templates.info/templates"> Це посилання на
сайт с безкоштовними шаблонами html i css </a>
```

Відносні посилання використовуються для переміщення всередині поточного сайта. При створенні відносних посилань необхідно розуміти, яке значення для атрибута `href` слід указувати, оскільки це значення залежить від вихідного розташування файлів:

- файли розміщують в одній папці:

```
<a href="page2.htm"> Для переходу на page2 клацни тут! </a>
```

- файли розміщують у різних папках:

```
<a href="subfolder/page2.htm"> Для переходу на page2 клацни
тут! </a>
<a href="…/page2.htm"> Для переходу на page2 клацни тут! </a>
<a href="…/…/page2.htm"> Для переходу на page2 клацни тут!
Вихідний файл знаходиться в двох вкладених папках </a>
```

Будь-яке посилання на веб-сторінці може перебувати в одному з таких станів.

Невідвідуване посилання. Такий стан характеризується для посилань, які ще не відкривали. За замовчуванням невідвідані текстові посилання зображуються синім кольором із підкресленням.

Активне посилання. Посилання позначається як активне в момент його відкриття. Колір такого посилання за замовчуванням червоний

Відвідане посилання. Як тільки користувач відкриває документ, на який вказує посилання, воно позначається як відвідане і змінює свій колір на фіолетовий, установлений за замовчуванням.

Для змінення кольору посилань у всьому документі існують спеціальні атрибути елемента `BODY`:

`LINK` – колір невідвідуваних посилань;

`ALINK` (Active Link) – колір активних посилань;

`VLINK` (Visited Link) – колір вже відвіданих посилань.

Усі кольори задаються або RGB-значенням в шістнадцятковій системі, або одним із 16 базових кольорів. Наприклад, щоб зробити всі посилання в документі червоними, уже відвідані – зеленими, а активні – жовтими, слід написати такий html-код:

```
<body link="red" vlink="green" alink="yellow">
```

Будь-яке посилання є вбудованим елементом, тому для нього діють ті самі правила, що й для вбудованих: не можна розміщувати всередині тега `<a>` блокові елементи, але допускається робити навпаки, вкладаючи посилання в блоковий контейнер, наприклад:

```
<!DOCTYPE HTML>
<html>
<head>
 <meta charset="utf-8">
 <title>Помилки при використанні посилань</title>
</head>
<body>
 <a href="lion.html" ><h1>Типова помилка</h1></a>
 <h1><a href="lion.html" >Правильний сінтаксис</a></h1>
</body>
</html>
```

Створення посилання на адресу електронної пошти робиться майже так само, як і посилання на веб-сторінку. Тільки замість URL указується `mailto: адреса електронної пошти`. В атрибуті `href` тега `<a>` спочатку пишеться ключове слово `mailto`, потім через двокрапку бажана поштова адреса.

```
<a href="mailto:admin@html-templates.info"> Написати листа адміністратору сайта html-templates.info </a>
```

При натисканні на посилання запускається поштова програма, установлена за замовчуванням. Можна також автоматично додати тему повідомлення, приєднавши до адреси електронної пошти через символ питання `(?)` параметр `subject = "тема повідомлення"`, наприклад:

```
<a href="mailto:admin@html-templates.info?subject=Питання по HTML"> Написати листа адміністратору сайта html-templates.info </a>
```

При запуску поштової програми поле «Тема» (Subject) буде заповнено автоматично.

Слід зазначити, що за замовчуванням при переході за посиланням документ відкривається в поточному вікні або фреймі. За необхідності цю умову можна змінити атрибутом `target` тега `<a>`. Синтаксис є таким:

```
<a href="http://html-templates.info/templates" target="_blank">
Це посилання на сайт із безкоштовними шаблонами відкриється в новому вікні </a>
```

Як зарезервовані імена застосовуються такі:

`_blank` – завантажує сторінку в нове вікно браузера.

`_self` – завантажує сторінку в поточне вікно (це значення задається за замовчуванням).

Атрибут `title` указує заголовок посилання, який виникає при наведенні на нього:

```
<a href=" http://html-templates.info/templates" title="Сайт з безкоштовними шаблонами"> Посилання на сайт з безкоштовними шаблонами. При наведенні курсора виникне заголовок </a>
```

«Якори» в гіперпосиланнях. З допомогою гіперпосилань можна переміщатися не тільки між сторінками документа, але й усередині сторінки. Це є дуже корисним, коли стаття складається з кількох розділів і має досить великий вміст. У цьому випадку розумно буде використовувати так звані якорі, щоб швидко переходити в потрібний розділ. У тому елементі, на який слід перейти, необхідно визначити унікальний ідентифікатор. Для цього існує атрибут `id`, який можна визначати майже для будь-якого HTML-тега. Назва цього атрибута може бути будь-якою і обов'язково має бути унікальною. У значенні атрибута `href` тега посилання ставиться символ решітки, після якого вказується ім'я того ідентифікатора, до якого необхідно звернутися:

```
<a href="#p10">перейти до 10-го параграфа</a>
….
 <h1 id="p10">А ось і 10-й параграф </h1>
```

Практичні завдання

1 Скопіювати вміст цього прикладу й зберегти його у html-файл під ім'ям index2.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
```

```
 <meta http-equiv="Content-Type" content="text/html; charset=utf-
8">
 <title>Приклад веб-сторінки</title>
</head>
<body>
 <h1>Заголовок</h1>
 <!-- Коментар -->
 <p>Перший абзац. </p>
 <p>Другий абзац. </p>
</body>
</html>
```

Запустити браузер і відкрити файл через пункт меню «Файл > Открыть файл» (Ctrl + O). У діалоговому вікні вибору документа вказати файл index2.html.

2 Написати код HTML, щоб отримати результат, показаний на рисунку 2.1.

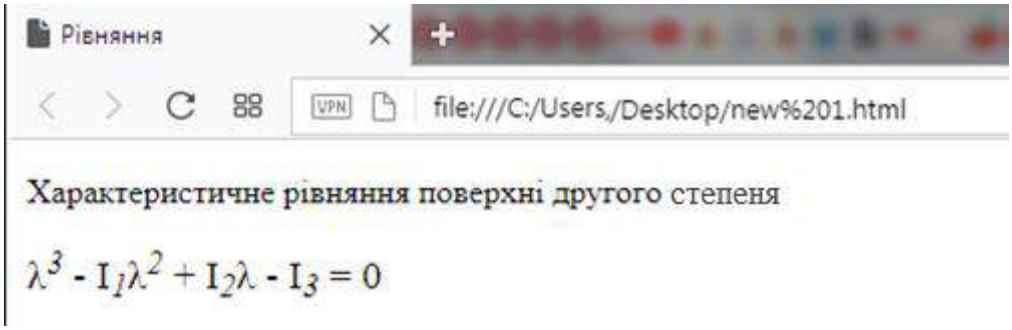


Рисунок 2.1 – Результат виконання другого завдання

3 Використовуючи вкладення тегів, зробити список, наведений на рисунку 2.2.

4 Зробити 3 html-сторінки і зв'язати їх посиланнями.

5 Зробити посилання на верх сторінки, використовуючи «якорі» в гіперпосиланнях.

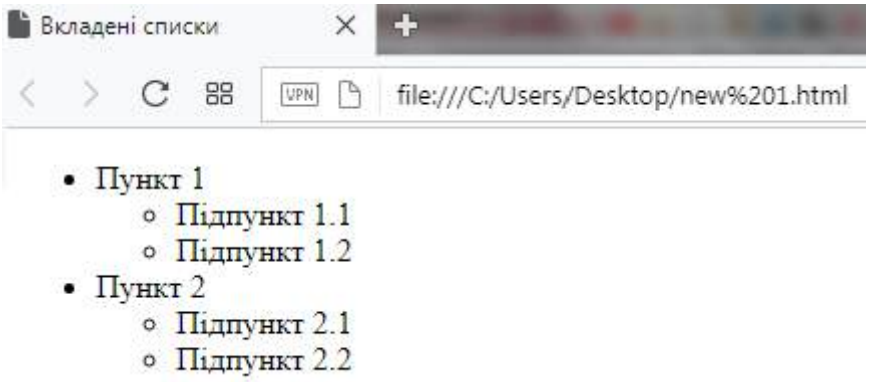


Рисунок 2.2 – Результат виконання третього завдання

3 ФОРМАТУВАННЯ ЗОБРАЖЕНЬ З ДОПОМОГОЮ HTML

Зображення додаються за два етапи: спочатку готують графічний файл бажаного розміру, потім його додають на сторінку через тег <img>. Сам HTML призначено тільки для того, щоб відобразити необхідну картинку, при цьому ніяк її не змінюючи.

При підготовці зображень слід урахувати розмір графічного файла, оскільки веб-сторінка завантажується через мережу. Чим меншим є файл, тим швидше виникне зображення.

3.1 Формати файлів зображень

Сьогодні у веб-документах використовуються три формати зображень: jpeg, gif, png. Розглянемо кожен з них докладніше.

Формат JPEG. Переваги: невеликий розмір файла; підтримує 16 млн кольорів; можна установлювати якість зображення при збереженні. Недоліки: при стисканні розмірів утрачається якість; не підтримує прозорість.

Формат GIF. Переваги: підтримує анімацію, прозорість; при стисканні розмірів не втрачається якість. Недоліки: невелика кількість кольорів – до 256.

*Формат PNG по*діляється на формати png-8 (аналог формату gif, за винятком того, що png-8 не підтримує анімацію) і png-24. Ще існує формат png-32, але його використовують нечасто через великий розмір файла.

Переваги формату PNG-8: підтримує прозорість; при стисканні розмірів не втрачається якість. Недоліком є невелика кількість кольорів – до 256.

Переваги формату PNG-24: при стисканні розмірів не втрачається якість; використовує 16 млн кольорів; плавний перехід від прозорої області до кольорової. Недолік – великий розмір файла.

Із сильних і слабких сторін розглянутих форматів зображень можна зробити висновок, що для збереження фотографій з чіткими краями найкращим є формат jpeg, оскільки розмір файла в цьому форматі невеликий. Для анімованих зображень єдиний варіант – це використовувати формат gif, а для збереження якісних зображень, у яких використовується прозорість, найкращим буде формат png-24.

3.2 Завантаження зображень на сторінку

Браузер розміщує зображення там, де в документі виникає тег <img>. Якщо розмістити тег <img> між двома параграфами, то браузер покаже перший параграф, потім зображення, а потім другий параграф. Якщо ж

розмістити тег <img> усередині параграфа, то зображення буде відображено в одному рядку з текстом:

```
<html>
<body>
<p>Перший параграф</p>
<img src="http://www.google.com/img/logo.gif">
<P>Другий параграф</p>
<p>У цьому прикладі зображення
<img src="http://www.google.com/img/logo.gif">
розміщено прямо всередині параграфа</p>
</body>
</html>
```

Цей тег має обов'язковий атрибут src, у значенні якого вказуємо шлях до зображення, який може бути як відносним, так і абсолютним. У тегу <img/> можна вказати також додаткові атрибути:

```
<img src="img/my_foto.jpg" alt="це моє фото" title="це моя фотографія" width="300" height="300" />
```

Атрибут alt означає альтернативний текст. Його використовують для того, щоб, по-перше, користувач міг побачити той текст, який указано в цьому атрибуті, якщо картинка не завантажиться на сторінку, і, по-друге, в альтернативному тексті, необхідному при просуванні сайта, можна вказати ключові слова, і тоді це зображення буде брати участь у пошуку по картинках.

Наступний атрибут – це title, який є універсальним, його можна використовувати майже для будь-якого тега. Те, що написано в його значенні, буде виводитися у вигляді підказки.

Будь-якому зображенню можна задати ширину й висоту, указавши ці значення в атрибутах width і height відповідно. З допомогою цих атрибутів зображення краще не збільшувати, оскільки якість буде дуже поганою. Якщо задати ширину й висоту зображення, то браузер при завантаженні сторінки буде відразу виділяти задану область під картинку.

Завжди зберігайте всі зображення в окремій папці, тоді будете знати, де їх шукати.

Атрибут ALIGN визначає спосіб вирівнювання зображення по горизонталі:

- align=top – зображення вирівнюється по верхньому краю поточного текстового рядка, не змінюючи позиції по горизонталі, при цьому йдеться як про текст, так і про графіку;
- align=middle – зображення центрується по вертикалі на базовій лінії поточного текстового рядка, не змінюючи позицію по горизонталі;

– align=bottom – нижній край зображення вирівнюється по горизонталі на базовій лінії поточного текстового рядка;

– align=left – зображення зміщується до лівого краю робочої зони, наступний текст відразу ж починає обтікати зображення;

– align=right – те ж саме, що й для left, тільки зображення зміщується до правої частини робочої зони.

Зображення в HTML можна окреслити прямокутною рамкою, для чого використовується атрибут border, який задає ширину рамки навколо зображення в пікселях:

```
<html>
<body>
<p> <a href="http://www.google.com"><img src="logo.gif" align="left"> </a> Незважаючи на те, що значення border не задано, зображення-гіперпосилання виводиться окресленим рамкою шириною 2 пікселя (значення за замовчуванням) </p>
<p> <a href="http://www.google.com"><img src="logo.gif" align="left" border="0"> </a> У цьому варіанті рамку навколо гіперпосилання прибрано завдяки примусовому присвоєнню параметру border нульової товщини </p>
</body>
</html>
```

Атрибути HSPACE і VSPACE – відступи в пікселях по горизонталі й вертикалі від зображення до інших об'єктів документа. Легко запам'ятати назву, якщо її просто перекласти з англійської. HSPACE – горизонтальний відступ і VSPACE – вертикальний відступ.

```
<html>
<body>
<!другий приклад з відступами й вирівнюванням по правому краю!>
<p align="justify"> <img src="pchela.jpg" width="65" height="59" hspace="15" vspace="15" align="right"> Будь-який інтернет-бізнес є немислимим без сайта. Тому насамперед необхідно навчитися створювати сайти. У цьому випадку фраза «навчитися створювати сайти» буде означати: знати на хорошому рівні мову гіпертекстової розмітки HTML і каскадні таблиці стилів CSS, які значно розширюють можливості HTML і дають змогу створювати дійсно стильні і якісні сайти. </p>
</body>
</html>
```

3.3 Колір фону й тексту

За замовчуванням текст веб-сторінки відображається у браузері чорним кольором на білому фоні. Для того щоб це змінити, є декілька атрибутів тега <body>.

Атрибут BACKGROUND визначає зображення для «заливки» фону. Значення задається у вигляді повного URL або імені файла з картинкою у форматі GIF або JPG.

Атрибут BGCOLOR визначає колір фону документа.

Атрибут TEXT визначає колір тексту в документі.

Значення кольорів задаються або RGB-значенням в шістнадцятковій системі, або одним із шістнадцяти базових кольорів. Наприклад, задамо фоновий колір і колір тексту:

```
<body bgcolor="#FFF8D2" text="red">
<p> Цей текст буде червоним, тому що ми змінили колір тексту в тезі <body> і тепер увесь текст на сторінці за замовчуванням буде червоним </p>
<font color ="green">
<p> У цьому абзаці текст буде зеленим, тому що ми вклали його в теги <font> і надали відповідного кольору </p>
</font>
<p> Тепер текст знову буде червоним </p>
</body>
```

Тепер задамо фонове зображення і колір тексту:

```
<body background="fon.jpg" text="red">
<p> Цей текст буде червоним, тому що ми змінили колір тексту в тезі <body>, і тепер увесь текст на сторінці за замовчуванням буде червоним </p>
<p> Тепер тут текст теж червоний і тільки <font color ="green"> ці слова зелені </font>
</p>
<p> Тут текст теж червоний </p>
</body>
```

Практичні завдання

1 Відформатувати сторінку так, як показано на рисунку 3.1. Для цього завдання можна використати уривок з будь-якої казки.

2 Створити свою домашню сторінку відповідно до зразка на рисунку 3.2.

Рисунок 3.1 – Зразок html-сторінки для завдання 1

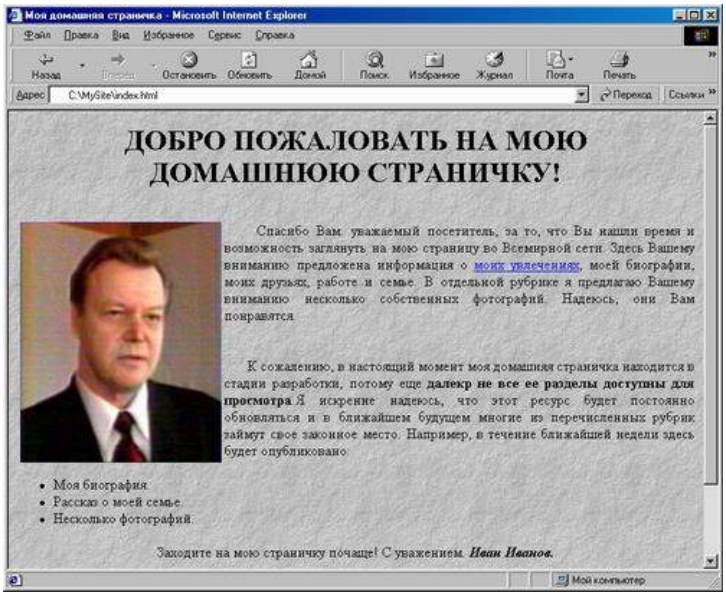


Рисунок 3.2 – Зразок html-сторінки для завдання 2

4 ФОРМАТУВАННЯ ТАБЛИЦЬ З ДОПОМОГОЮ HTML

HTML-таблиці зазвичай використовують не тільки для відображення таблиць як таких, а й для створення невидимого каркаса HTML-сторінки, який допомагає розташувати текст і зображення належним чином. Раніше всі сайти мали табличну структуру, зараз все більшої популярності набуває верстання з допомогою блоків (`<div>`). Класичний приклад табличної структури HTML-сторінки показано на рисунку 4.1.

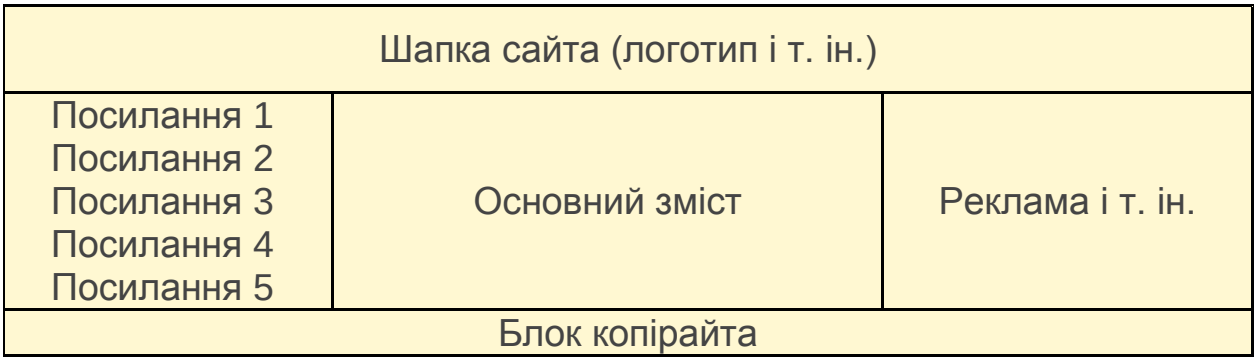


Рисунок 4.1 – Таблична структура html-сторінки

Таблиця складається з рядків і стовпців осередків, які можуть містити текст і рисунки. Основні теги для форматування таблиць наведено у таблиці 4.1.

Таблиця 4.1 – Основні теги html для форматування таблиць

Тег	Опис
`<table>`	Визначає таблицю
`<th>`	Визначає заголовки в таблиці
`<tr>`	Визначає рядок таблиці
`<td>`	Визначає елемент таблиці
`<caption>`	Визначає назву таблиці
`<colgroup>`	Визначає групи стовпців таблиці
`<col>`	Визначає атрибути для одного або декількох стовпців таблиці

Тег `<table>` обов'язково повинен мати початковий і кінцевий теги. За замовчуванням HTML-таблиця друкується без рамки, а розмітка здійснюється автоматично залежно від обсягу інформації, що міститься у ній. Тег `<tr>` (table row) створює новий рядок таблиці. Закривальний тег є обов'язковим. Тег `<td>` (table data) починає і закінчує кожну клітинку стовпця HTML-таблиці. Обов'язковим є закривальний тег. Тег `<caption>` призначено для створення заголовка таблиці й може розміщуватися лише всередині контейнера `<table>`, причому відразу після відкривального тега. Такий заголовок являє собою текст, який за замовчуванням відображається перед таблицею і описує її вміст. Тег `<colgroup>` призначено для задання ширини і стилю однієї або кількох колонок таблиці. Цей тег дає змогу зменшити код таблиці завдяки зменшенню повторюваних атрибутів, і за наявності цього тега браузер починає показувати вміст таблиці, не чекаючи

її повного завантаження. Тег <colgroup> можна використовувати в комбінації з тегом <col>, який визначає характеристики однієї або кількох колонок.

Наприклад:

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
 </head>
 <body>
  <table border="1">
   <caption> Змінення видобутку ресурсів за роками</caption>
   <colgroup width="150">
    <col span="5">
    <col span="4">
   </colgroup>
   <tr>
    <td> </td>
    <td>1995</td>
    <td>1996</td>
    <td>1997</td>
    <td>1998</td>
    <td>1999</td>
    <td>2000</td>
    <td>2001</td>
    <td>2002</td>
    <td>2003</td>
   </tr>
    <tr>
    <td>Нафта</td>
    <td>5</td>
    <td>7</td>
    <td>2</td>
    <td>8</td>
    <td>3</td>
    <td>34</td>
    <td>62</td>
    <td>74</td>
    <td>57</td>
   </tr>
   <tr>
    <td>Золото</td>
    <td>3</td>
    <td>6</td>
    <td>4</td>
    <td>6</td>
    <td>4</td>
    <td>69</td>
    <td>72</td>
```

```
        <td>56</td>
        <td>47</td>
       </tr>
       <tr>
        <td>Дерево</td>
        <td>5</td>
        <td>8</td>
        <td>3</td>
        <td>4</td>
        <td>7</td>
        <td>73</td>
        <td>79</td>
        <td>34</td>
        <td>86</td>
       </tr>
      </table>
     </body>
    </html>
```

4.1 Основні атрибути тегів форматування таблиць

Для задання розміру таблиці (ширини й висоти) використовують теги <width> і <height>. Тег <width> визначає ширину таблиці html. Ширина задається або в пікселях, або в процентному відношенні до ширини вікна браузера. Тег <height> визначає висоту таблиці. Висота задається або в пікселях, або в процентному відношенні до висоти вікна браузера. За замовчуванням ці атрибути визначаються автоматично, залежно від обсягу матеріалу, що міститься в таблиці.

Для визначення відстані (у пікселях) між рамкою кожної комірки html-таблиці й матеріалом, що міститься в ній, використовують тег <cellpadding>, який визначає відстань (у пікселях) між межами сусідніх осередків таблиці html.

Тег <border> установлює товщину рамки в пікселях.

Наприклад:

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
   </head>
<body>

 <table border="1" width="400" height="100" cellpadding="10"
cellspacing="10">
<tr>
<td> рядок 1 комірка 1 </td><td>рядок 1 комірка 2 </td>
</tr>
```

```
<tr>
<td> рядок 2 комірка 1 </td> <td> рядок 2 комірка 2 </td>
</tr>
</table>
 </body>
</html>
```

Для об'єднання комірок таблиці використовують теги <colspan> і <rowspan>. Тег <colspan> установлює кількість комірок, які має бути об'єднано по горизонталі. За замовчуванням цей тег має значення 1. Тег <rowspan> визначає кількість комірок, які має бути об'єднано по вертикалі, за замовчуванням – має значення 1.

Наприклад:

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
   </head>
<body>
 <table border="1">
<tr>
<td> рядок 1 комірка 1 </td> <td> рядок 1 комірка 2 </td>
</tr>
<tr>
<td colspan="2"> рядок 2 комірка 1+2 </td>
</tr>
<tr>
<td rowspan="2"> рядок 3+4 комірка 1 </td>
<td> рядок 3 комірка 2 </td>
</tr>
<tr>
<td> рядок 4 комірка 2 </td>
</tr>
</table>
 </body>
</html>
```

Тег <align> визначає спосіб горизонтального вирівнювання html-таблиці або вмісту комірок. Можливі значення: left, center, right; значення за замовчуванням – left. Тег <valign> визначає спосіб вертикального вирівнювання таблиці або вмісту елементів таблиці. Можливі значення: top, bottom, middle (притиснути до верху, притиснути до низу, установити посередині).

Тег <bgcolor> визначає колір фону елементів таблиці. Задається або RGB-значенням в шістнадцятковій системі, або одним із шістнадцяти базових кольорів. Тег <background> дає змогу заповнити фон таблиці

рисунком. Як значення необхідно вказати URL рисунка. Якщо картинка є меншою за комірку, то її буде продубльовано, а якщо більше – то буде відображатися та частина, яка вміститься в комірку.

Наприклад:

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
   </head>
<body>
<!-- задаємо ширину, висоту, границю і фоновий колір усієї таблиці, вирівнювання залишаємо за замовчуванням (по лівому краю)-->
<table width="400" height="100" border="1" bgcolor="#FFF8D2">
<tr>
<!-- горизонтально по центру, вертикально притискаємо до верху комірки-->
<td align="center" valign="top" > рядок 1 комірка 1 </td>
<!-- горизонтально по центру, вертикально за замовчуванням (по центру) і робимо фоновий рисунок-->
<td align="center" background="pchela.jpg"> рядок1 комірка 2 </td>
</tr>
<tr>
<!-- горизонтально по правому краю, вертикально притискаємо до низу -->
<td align="right" valign="bottom"> рядок 2 комірка 1 </td>
<!-- горизонтально по центру, вертикально за замовчуванням (по центру) і змінюємо фоновий колір -->
<td align="center" bgcolor="#33FF99"> рядок 2 комірка 2</td>
</tr>
</table>
</body>
</html>
```

Практичні завдання

1 Створити дві таблиці за зразками й заповнити їх інформацією.

Таблиця 1 – Назва

<table>
<tr><td colspan="4">Назва стовбця</td><td>Назва стовбця</td></tr>
<tr><td rowspan="2">Вміст</td><td>Вміст</td><td rowspan="2">Вміст</td><td rowspan="2">Вміст</td><td rowspan="2"></td></tr>
<tr><td rowspan="2">Вміст</td></tr>
<tr><td>Вміст</td><td>Вміст</td><td>Вміст</td></tr>
</table>

Таблиця 2 – Назва

<table>
<tr><th>Назва стовбця</th><th>Назва стовбця</th><th>Назва стовбця</th></tr>
<tr><td rowspan="3">Вміст</td><td>Вміст</td><td>Вміст</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td>Вміст</td><td>Вміст</td></tr>
</table>

Для заповнення відповідної комірки можна використовувати свій рисунок.

2 Зробити таблицю html і відформатувати її так, як показано на рисунку 4.2.

Без відступу:

Перший	Рядок
Другий	Рядок

З відступом (cellpadding) :

Перший	Рядок
Другий	Рядок

Рисунок 4.2 – Відображення таблиці в браузері

5 ФОРМИ HTML

Форми html призначено для організації взаємодії з користувачем. З їх допомогою можна вводити текст, вибирати із запропонованих значень з допомогою списків або кнопок, організовувати інтерактивний обмін інформацією між Web-сторінкою і сервером. Можна визначити форми як електронні бланки для заповнення різних даних, таких як ім'я, вік, країна проживання, та інших. Зазвичай форма працює спільно з установленим на сервері сценарним додатком, що обробляє введену інформацію.

Документ може містити будь-яку кількість форм, але одночасно на сервер відправляється тільки одна форма, тому дані форм мають бути незалежними. У таблиці 5.1 наведено основні теги для роботи з формами html.

Таблиця 5.1 – Основні теги для роботи з формами html

Тег	Опис
`<form>`	Визначає форму html
`<input>`	Визначає поле введення
`<textarea>`	Визначає текстову область (елемент керування для введення багаторядкового тексту)
`<label>`	Визначає мітку для елемента керування
`<fieldset>`	Впливає на доступність полів
`<legend>`	Визначає назву для набору полів
`<select>`	Визначає список вибору (спадне поле)
`<optgroup>`	Визначає групу варіантів вибору
`<option>`	Визначає варіант зі спадного поля
`<button>`	Визначає кнопку

Форма визначається з допомогою тегів `<form> </form>`, між якими розташовуються поля введення, кнопки, а також усі необхідні елементи оформлення форми.

Тег `<form>` має кілька атрибутів, з яких необхідно виокремити атрибути `action` і `method`, без яких форма не зможе передати інформацію від користувача на сервер. Наприклад:

```
<form action = "html_form_action.asp" method = get>
  ...
</form>
```

Атрибут `action` указує URL-адресу об'єкта, який отримає дані форми.

Атрибут `method` – метод протоколу http, який може мати два значення: `get` і `post`.

Значення атрибута `method = get` змушує Web-браузер передати всі дані формуляра за URL-адресою, заданою в атрибуті `action`. При цьому введені при заповненні форми дані просто додаються в адресний рядок з використанням роздільника – знака запитання. Цей метод є зручним для невеликих форм.

Значення атрибута `method = post` змушує Web-браузер насамперед зв'язатися із сервером, що обробляє форму, і тільки після установлення зв'язку розпочати передання даних, для оброблення яких будуть використовуватися спеціальні сценарії.

Тег `<input>` призначено для створення текстових полів, різних кнопок, перемикачів і прапорців. Основний атрибут тега `<input>`, що визначає вид елемента, – `type`. З його допомогою можна задавати такі елементи форми: текстове поле (`text`), поле з паролем (`password`), перемикач (`radio`), прапорець (`checkbox`), приховане поле (`hidden`), кнопка (`button`), кнопка для відправлення форми (`submit`), кнопка для очищення форми (`reset`), поле для відправлення файла (`file`) і кнопка із зображенням (`image`). Для кожного елемента існує свій список атрибутів, які визначають його вид і характеристики. Обов'язковими для тега `<input>` є також атрибут `name`, який приписує певному елементу введення унікальне ім'я, що використовується для подальшого оброблення форми.

5.1 Текстові поля і їх основні атрибути

Текстове поле (`type = text`) визначає однорядкове поле введення і використовується, коли необхідно ввести у форму дані в довільній формі, але обмежені за обсягом (слова, словосполучення, числа і т. ін.). Атрибут `size` використовується для змінення ширини текстового поля (яке за замовчуванням становить 20 символів), тобто обмежує тільки видиму область уведення даних. Зазначимо, що атрибути для різних типів полів виведення і керування можуть різнитися. Так, наприклад, атрибут `size` для текстових полів уведення (`type = "text"` або `type = "password"`) указує максимальну кількість символів, що відображаються у полі, а для інших елементів – їх розмір по горизонталі в пікселях. Для змінення довжини введеного рядка використовується атрибут `maxlength`.

Наприклад. Створимо просту форму для введення імені та прізвища, при цьому поле для введення імені обмежене десятьма символами, а максимальна кількість символів для введення імені дорівнює трьом:

```
<html>
<body>
<form>
  Ім'я:
```

```
  <input type="text" name="firstname" size="10" maxlength="3">
  <br>
  Прізвище:
  <input type="text" name="lastname">
</form>
</body>
</html>
```

При використанні текстових полів можна задати значення за замовчуванням. Для цього використовується атрибут `value`. Його застосування покажемо на такому прикладі:

```
<html>
<body>
<form>
  Ім'я:
  <input type="text" name="firstname" value="Іван">
  <br>
  Прізвище:
  <input type="text" name="lastname" value="Іванов">
</form>
</body>
</html>
```

5.2 Поле пароля

Поле пароля (`type = password`) створює захищене поле введення, яке дає змогу користувачеві при заповнені форми ввести текст. Однак на відміну від звичайного текстового поля дані, що вводяться, при відображенні на моніторі замінюються зірочками або точками.

Наведемо просту форму для введення імені користувача і пароля:

```
<html>
<body>
<form>
  Ім'я користувача:
  <input type="text" name="user" value="Іван">
  <br>
  Пароль:
  <input type="text" name="password" value="rtrui">
</form>
</body>
</html>
```

5.3 Перемикачі й прапорці

Перемикачі, або радіокнопки (`type = radio`), визначають поля вибору одного значення з декількох доступних. Поля цього типу часто

використовуються в діалогових вікнах. Для кожної позиції перемикача створюється свій тег <input type = radio>. Групуються перемикачі з однаковим іменем, що задається атрибутом name. Необхідно зазначити, що на відміну від текстового поля й поля пароля атрибут value задає значення, яке буде передано до сервера для подальшого оброблення в разі вибору цього перемикача. Для вибору за замовчуванням одного з можливих значень групи перемикачів використовується атрибут checked:

```
<html>
<body>
<form>
  Укажіть Вашу стать:
  <br>
  <input type="radio" name="sex" value="male" checked="checked">
чоловік
  <br>
  <input type="radio" name="sex" value="female"> жінка
</form>
</body>
</html>
```

Прапорці type = checkbox використовуються, коли необхідно, щоб користувач вибрав один або кілька варіантів з обмеженої їх кількості. Прапорці у формі не залежать один від одного, їх можна встановити або скинути у будь-якій комбінації. Для кожного прапорця необхідно задати своє унікальне ім'я з допомогою атрибута name. Створення двох прапорців з одним ім'ям не призведе до помилки при відображенні форми, але не дасть змогу сценаріям оброблення на сервері коректно обробити дані, які передаються з форми.

З допомогою атрибута checked можна встановити, які з прапорців будуе вибрано за замовчуванням при завантаженні сторінки. Відмінність від перемикачів полягає тільки в тому, що для прапорців можна вказати відразу кілька варіантів. При відправленні форми з прапорцями на сервер вибраним прапорцям присвоюється значення за замовчуванням «on». Зазвичай, цього достатньо для коректного оброблення даних, але в деяких випадках зручніше поставити кожному прапорцю своє значення з допомогою атрибута value:

```
<html>
<body>
<form>
 Цього року я збираюся придбати:
  <br>
  <input type="checkbox" name="computer" checked="checked">
```

```
      Комп'ютер
      <br>
      <input type="checkbox" name="notebook">
      Ноутбук
      <br>
      <input type="checkbox" name="printer">
      Принтер
      <br>
      <input   type="checkbox"   name="scanner"   checked="checked"
value="Принтер">
      Сканер
    </form>
    </body>
    </html>
```

5.4 Командні кнопки

Командна кнопка відправлення (`type = submit`) використовується для виконання пересилання даних форми на сервер. Метод відправлення й адреса файла, який обробляє отриману інформацію, задаються тегом `<form>` з допомогою атрибутів `method` і `action`. Командна кнопка скидання `(type = reset)` повертає форму до вихідного стану (очищує форму), при цьому дані не передаються. Крім кнопок відправлення і скидання існує також можливість додавати спеціальні кнопки (`type = button`), які можуть використовуватися для виконання процедур (скриптів) безпосередньо на Web-сторінці.

Форма пошуку з кнопками надсилання й скидання:

```
    <html>
    <body>
      <form name="input" action="html_form_action.asp" method="get">
      Знайти:
      <input type="text" name="search" size=25>
      <br>
      <input type="submit">
      <input type="reset">
      <input type="button" value="Кнопка">
    </form>
    </body>
    </html>
```

Якщо ввести в текстове поле якісь символи й натиснути кнопку «Отправить», то введену інформацію буде надіслано на сторінку з ім'ям "`html_form_action.asp`". При натисканні на кнопку «Сбросить» текстове поле очиститься. Написи на кнопках «Отправить» і «Сбросить»

установлено за замовчуванням. Для їх змінення необхідно використовувати атрибут value.

5.5 Поле вибору файла

Для створення поля вибору файла, який буде завантажено на сервер разом з інформацією форми, використовується значення type = file. Командна кнопка «Выберите файл» відкриває стандартне діалогове вікно вибору файла. Для поля вибору файла, за аналогією з текстовим полем можна використовувати атрибути size, maxlength, value:

```
<html>
<body>
  <form name="input" action="html_form_action.asp" method="get">
   Прикріпити файл:
  <br>
  <input type="file" size="50">
</form>
</body>
</html>
```

5.6 Списки вибору

Існують два типи списків вибору: такі, що розкриваються (спадне меню), і з множинним вибором. Незалежно від типу списки описуються однаково з допомогою пари тегів <select> </select>. Окремі елементи списку задаються з використанням тега <option>. Тип списку визначається з допомогою атрибутів тега <select>.

Атрибут name задає ім'я поля для відправлення вибраних пунктів списку на сервер, атрибут multiple дає змогу використовувати множинний вибір, атрибут size визначає, яку кількість пунктів списку буде одночасно відображено на екрані. При цьому якщо атрибут multiple не задано і size = 1, то на екрані відображається спадний список, якщо ж задано атрибут multiple або значення size більше одиниці, то список відображається розгорнутим.

Спадні списки вибору за своїм призначенням відповідають перемикачам, водночас їх використовують переважно в тих випадках, коли кількість варіантів вибору є досить великою. Зазвичай при виборі з понад трьох варіантів бажано замість перемикачів використовувати спадні списки. Зазначимо, що за замовчуванням вибирається перше значення зі списку. З допомогою атрибута selected тега <option> це значення можна змінити. У наступному прикладі показано список вибору розміру екрана ноутбука з

попередньо встановленим значенням 15.4:

```
<html>
<body>
<form>
  Вибір розміру екрана ноутбука
  <select name="tft">
    <option value="tft-12">12"
    <option value="tft-13">13"
    <option value="tft-14">14"
    <option value="tft-15">15"
    <option value="tft-15-4" selected="selected">15.4"
    <option value="tft-17">17"
  </select>
</form>
</body>
</html>
```

Розгорнуті списки вибору використовуються, якщо необхідно вибрати багато значень. Для списків з множинним вибором значення атрибута `size` за замовчуванням установлюється таким, що дорівнює чотирьом, крім того, жодне зі значень не є вибраним за замовчуванням. У наступному прикладі наведено розгорнутий список вибору ноутбука за виробником висотою в шість пунктів з можливістю множинного вибору і з попередньо встановленими значеннями «Compaq» і «Sony»:

```
<html>
<body>
<form>
  Вибір ноутбука за виробником:
  <select name="notebook" multiple size=6>
    <option value="acer">Acer
    <option value="asus">Asus
    <option value="compaq" selected="selected">Compaq
    <option value="hp">HP
    <option value="sony" selected="selected">Sony
    <option value="toshiba">Toshiba
  </select>
</form>
</body>
</html>
```

5.7 Текстова область

На відміну від текстового поля `<input type = text>` текстова область дає змогу вводити багаторядковий текст великого обсягу. Такі області часто використовуються при введенні повідомлень, коментарів.

Текстова область описується з допомогою тегів <textarea> </textarea>, між якими можна розмістити попередньо відформатований стандартний текст. Атрибути cols і rows задають розмір видимої області текстового поля.

У наступному прикладі створено текстову область з попередньо введеним текстом:

```
<html>
<body>
  <textarea rows="7" cols="30">
У цьому прикладі ми створили текстову область шириною 30 символів і висотою 7 рядків. Задане значення висоти не обмежує загальний обсяг уведеного тексту, а впливає тільки на розмір текстової області, яка відображається на екрані.
Для перегляду всього тексту необхідно скористатися смугою прокручування.
  </textarea>
</body>
</html>
```

5.8 Групування полів форми

Теги <fieldset> </fieldset> об'єднують поля форми в групи, які виділяють їх візуально та спрощують навігацію у формі з допомогою клавіші [Tab] (насамперед обходяться поля в межах групи). Для того щоб об'єднати кілька елементів уведення або керування в групу, достатньо помістити їх усередині тегів <fieldset> </fieldset>. Закривний тег </fieldset> є обов'язковим, а з допомогою пари тегів <legend> </legend> створеній групі полів можна присвоїти ім'я. У наступному прикладі створено форму, яка містить поля для введення імені й прізвища, об'єднаних у першу групу, поля для введення інформації про місце роботи й займану посаду, об'єднаних у другу групу, а також кнопку відправлення. Кожній групі полів присвоєно свій заголовок:

```
<html>
<meta charset="utf-8">
<body>
<form name="input" action="html_form_action.asp" method="get">
  <fieldset>
    <legend> Особисті дані </legend>
    Ім'я:
    <input type="text" name="firstname" value="Іван">
    <br>
    Прізвище:
    <input type="text" name="lastname" value="Іванов">
    <br>
```

```
        Стать:
        <input       type="radio"       name="Sex"       value="Male"
checked="checked"> Чоловіча
        <input type="radio" name="Sex" value="Female"> Жіноча
      </fieldset>
      <fieldset>
        <legend> Дані про роботу </legend>
        Місце роботи:
        <input type="text" name="work" value="XAI">
        <br>
        Посада:
        <input type="text" name="status" value="Викладач">
      </fieldset>
       <input type="submit" value="Відправити">
    </form>
    </body>
    </html>
```

Практичні завдання

1 Створити форму html «Реєстрація в системі» за зразком, як показано на рисунку 5.1.

2 Створити форму html «Форма пошуку» за зразком, як показано на рисунку 5.2.

3 Створити форму html «Відправлення e-mail з форми» за зразком, як показано на рисунку 5.3.

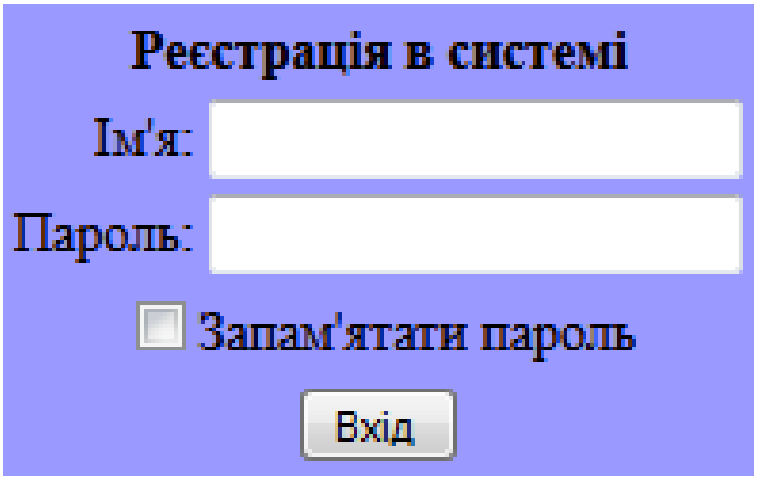


Рисунок 5.1 – Форма html «Реєстрація в системі»

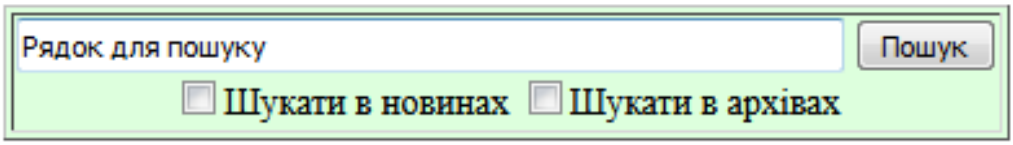


Рисунок 5.2 – Форма html «Форма пошуку»

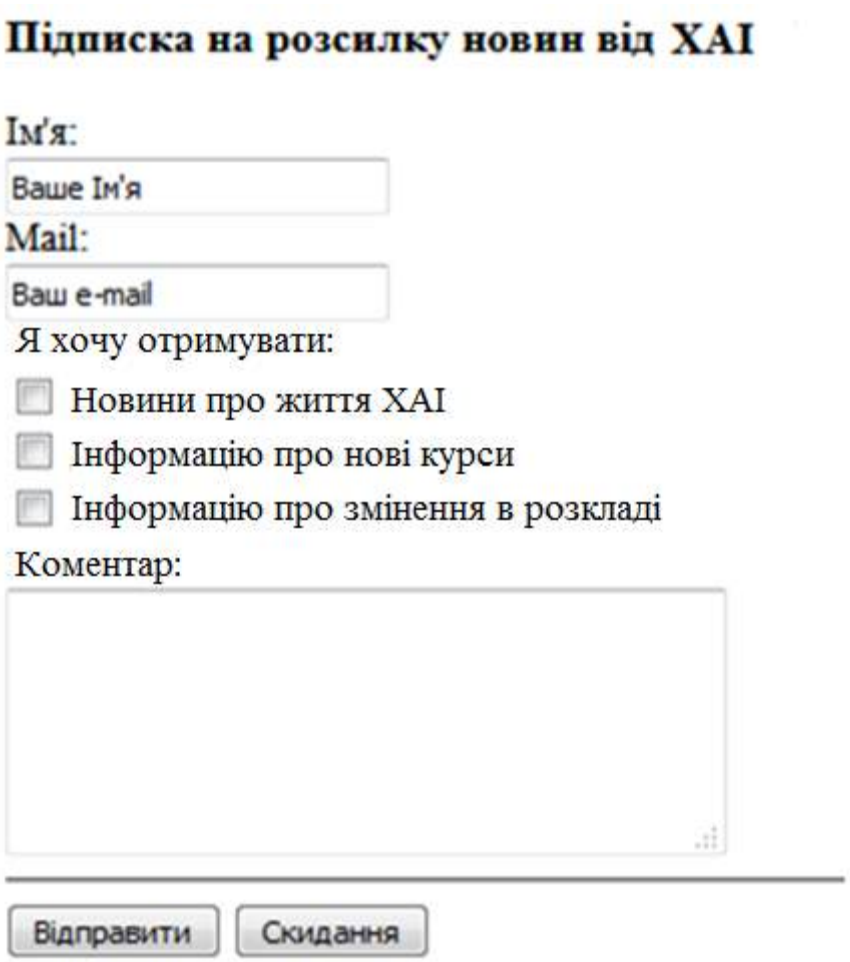


Рисунок 5.3 – Форма html «Відправлення e-mail з форми»

6 КАСКАДНІ ТАБЛИЦІ СТИЛЮ

Каскадні таблиці стилю (**CSS –** Cascade Style Sheets) – це проста технологія визначення й приєднання стилів оформлення до документів html. *Стиль оформлення* – це все те, що визначає зовнішній вигляд документа: розмір, колір і вид шрифту тексту, колір фону тексту, наявність меж, підкреслення, вирівнювання тексту і т. д. Стиль визначається набором правил відображення тегів, що задаються таблицею стилів. *Таблиця стилів*

– це шаблон, який керує форматуванням тегів html у web-документі. Таблиця стилів складається з набору правил опису стилю. Будь-яке правило каскадних таблиць стилів складається з двох частин: селектора й визначення. *Селектор* – це елемент або група елементів web-сторінки, для яких визначається форматування. *Визначення* описує конкретний вид форматування і складається з двох частин: властивості й значення, розділених знаком двокрапки. Наприклад, колір тексту абзацу можна задати з допомогою стилю (рисунок 6.1).

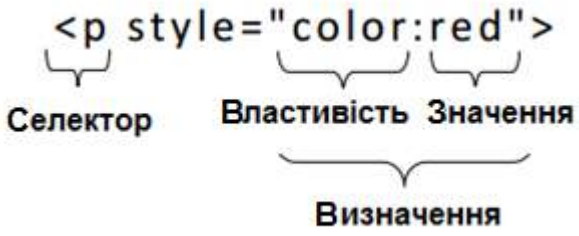


Рисунок 6.1 – Синтаксис правила таблиці стилів CSS

Основні переваги CSS:

- керування дизайном будь-якої кількості документів з допомогою однієї таблиці стилів;
- більш точний дизайн сторінок, який підтримується всіма браузерами;
- поділ документа на дві складові: структура й дизайн, завдяки чому вихідний код стає більш оптимізованим;
- нові розширені можливості порівняно зі звичайним html.

Змінюючи стильові правила у зовнішній таблиці стилів, можна керувати дизайном якої завгодно великої кількості сторінок (рисунок 6.2). Для цього необхідно під'єднати зовнішню таблицю стилів до всіх сторінок html для керування їх дизайном. Розташування опису стилів в окремому файлі має сенс у випадку, якщо ці стилі будуть застосовуватися до кількох сторінок. Для цього потрібно створити звичайний текстовий файл, описати з допомогою мови CSS необхідні стилі, розмістити цей файл на Web-сервері, а в коді Web-сторінок, які будуть використовувати стилі з цього файла, слід зробити посилання на нього.

У html-коді CSS таблиці під'єднуються з допомогою визначення атрибуту `href` тега `<link>`, наприклад:

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

Іншими словами, браузер буде використовувати правила відображення html-файла із CSS-файла. Перші два параметри цього тега є зарезервованими іменами, які потребуються для того, щоб повідомити браузеру, що на цій сторінці буде використовуватися CSS. Третій

параметр – `href = "URL"` – указує на файл, який містить опис стилів. Цей параметр повинен містити або відносний шлях до файла (у разі, якщо він знаходиться на тому ж сервері, що й документ, з якого до нього звертаються), або повний `url` («`http://...`»), у разі якщо файл стилів знаходиться на іншому сервері.

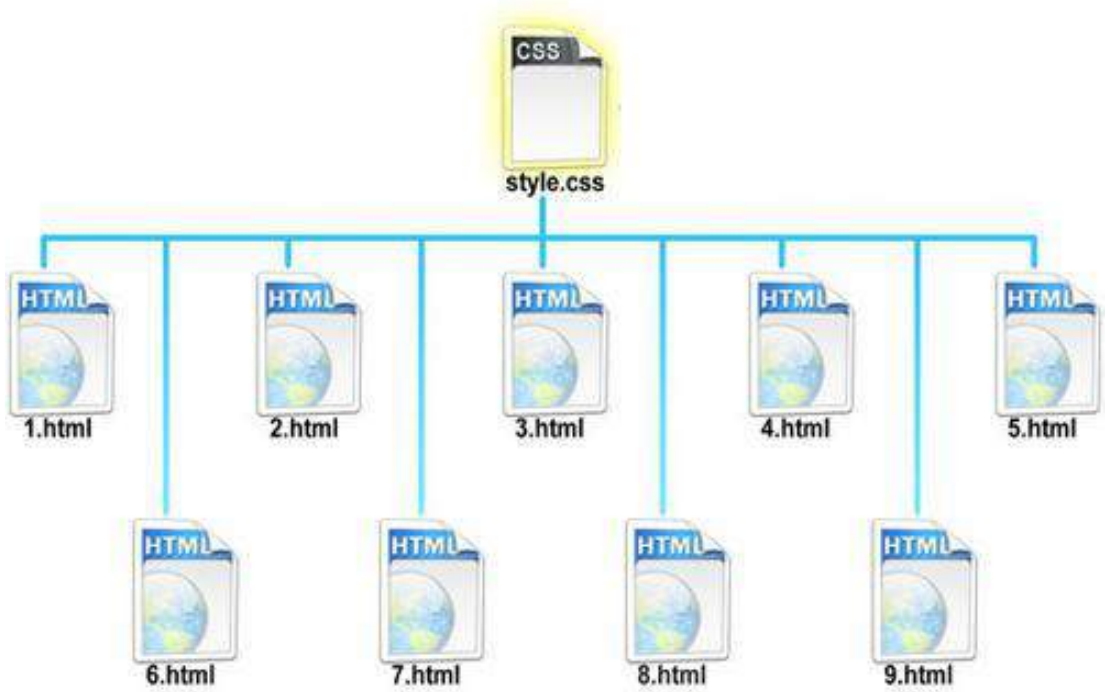


Рисунок 6.2 – Керування дизайном html-сторінок з допомогою таблиці стилів CSS

Наприклад. Зміст файла index.html:

```
<html>
<head>
<title>Працюємо зі стилями</ title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<h1>Це моя перша таблиця стилів, і якщо все працює, то, незважаючи на те, що це заголовок першого рівня, він відобразиться висотою всього лише 14 пікселів і буде блакитного кольору</ h1>
</body>
</html>
```

Зміст файла style.css:

```
h1 {color: blue; font-size: 14px}
```

6.1 Колір і фон у CSS

Основні властивості кольору й фону в CSS:

- color;
- background-color;
- background-image;
- background -repeat;
- background-attachment;
- background-position;
- background.

Властивість color задає основний колір (колір переднього плану) того чи іншого елемента. Наприклад, якщо необхідно зробити колір усіх заголовків першого рівня червоним, а колір тексту параграфів зеленим, то таблиця стилів буде мати такий вигляд:

```
H1 {color:red;}
P {color:green;}
```

Властивість background-color задає фоновий колір елемента. На відміну від html, у якому фоновий колір можна використовувати тільки для сторінки або елемента таблиці (тобто вони мають атрибут bgcolor), у CSS фоновий колір можна задавати для чого завгодно: для таблиць, заголовків, параграфів, посилань та ін. Наприклад, щоб змінити фоновий колір усієї сторінки, слід задати цю властивість елементу BODY, тому що саме він відповідає за тіло документа, тобто за всю сторінку:

```
BODY
{background-color: #FFEE8C;}
H1{color:red;
background-color:blue;}
P{color:green;}
```

Так само можна було б задати фоновий колір і для параграфів.

Властивість background-image використовується для задання фонового зображення. У прикладі задано фонове зображення для всієї сторінки, тобто для елемента BODY:

```
BODY{background-color:#FFEE8C;background-image:url(smile.png);}
H1{color: red; background-color: blue;}
P{color:green;}
```

Як значення властивості вказується шлях до зображення, тобто на початку пишеться url, а потім без пробілів у круглих дужках – розташування картинки. Якщо картинка знаходиться у тій же папці, то пишемо просто її назву, як в прикладі вище, якщо в папці, наприклад img,

то пишемо так: `url(img/smile.png)`.

Властивість `background-repeat` задає повторення фонового зображення у вікні браузера. За замовчуванням зображення повторюється, починаючи з верхнього лівого кута як по вертикалі, так і по горизонталі, поки не заповнить усе вікно браузера. Значення, яких може набувати властивість `background-repeat`, наведено в таблиці 6.1. Наприклад, при виконанні CSS-коду

```
BODY{
 background-image:
 url (smile.png);
 background-repeat:repeat-x;}
```

в браузері буде відображено картинку так, як показано на рисунку 6.3.

Таблиця 6.1 – Значення властивості `background-repeat`

Синтаксис	Значення
background-repeat: repeat-x;	Повторення по горизонталі
background-repeat: repeat-y;	Повторення по вертикалі
background-repeat: repeat;	Повторення по вертикалі й по горизонталі (значення за замовчуванням)
background-repeat: no-repeat;	Не повторюється

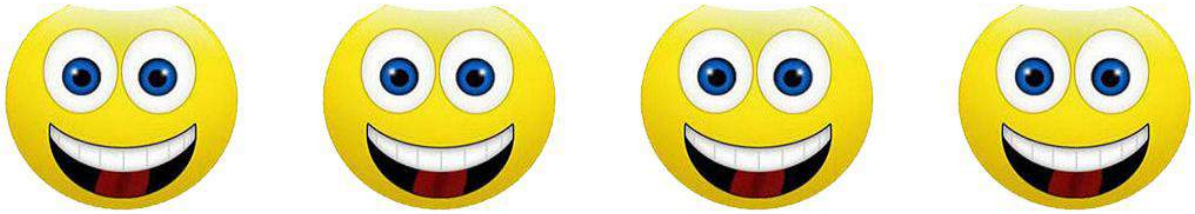


Рисунок 6.3 – Відображення картинки в браузері при виборі повторення по горизонталі у властивості `background-repeat`

Якщо не вказувати цю властивість, то буде використовуватися його значення за замовчуванням, тобто фонове зображення буде повторюватися як по вертикалі, так і по горизонталі.

Властивість `background-attachment` при наявності фонового рисунка встановлює, чи буде фонове зображення прокручуватися із вмістом сторінки, чи буде нерухомим. Може набувати два значення: `scroll` – фон прокручується разом із вмістом; `fixed` – фон строго зафіксовано. Значення за замовчуванням – `scroll`, тобто якщо взагалі не писати цю

властивість, то фон буде прокручуватися разом із вмістом, як у першому прикладі.

Приклади:

```
BODY{ background-image: url (smile.png);
      background-repeat: no-repeat;
      background-attachment: scroll;}
```

або

```
BODY{   background-image: url (smile.png);
       background-repeat: no-repeat;
       background-attachment: fixed;}
```

Властивість `background-position` задає позицію фонового зображення. Значення можна задавати у відсотках, одиницях довжини й з допомогою ключових слів. Відлік зазвичай починається з лівого верхнього кута, де й розташовується фонове зображення за замовчуванням. На початку вказуємо координату по горизонталі (по осі X), потім через пропуск координату по вертикалі (по осі Y). Координати можна задавати у відсотках від ширини вікна і в пікселях. Положення можна задавати також таким значеннями атрибутів та їх комбінацією (рисунок 6.4): `left` – ліво, `right` – право, `center` – центр, `top` – верх, `bottom` – низ.

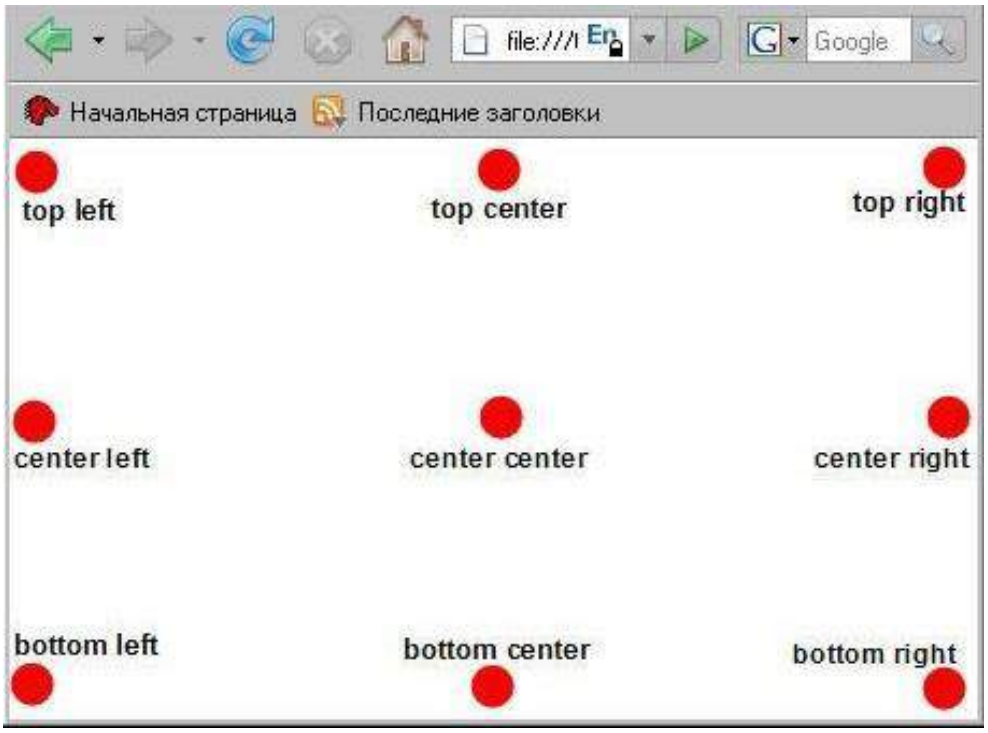


Рисунок 6.4 – Варіанти позиції фонового зображення при використанні властивості background-position

Приклади стилів:

```
BODY {background-image : url(smile.png) ;
      background-repeat: no-repeat;
      background-position: top right;}

BODY {background-image : url(smile.png) ;
      background-repeat: no-repeat;
      background-position: 300px 500px ;}

BODY {background-image : url(smile.png) ;
       background-repeat: no-repeat;
       background-position: 75% 25%;}
```

Скорочена форма запису – BACKGROUND . Властивість призначено для скороченого запису всіх розглянутих вище властивостей.

Порядок властивостей цього елемента є таким:

```
background-color_background-image_background-repeat_background-
attachment _background-position
```

тобто просто записується п'ять значень властивостей через пропуск у рядок:

```
BODY {background: #ffee8c url(smile.png) no-repeat fixed top right ;}
```

6.2 Ідентифікація і групування елементів (class і id)

Іноді необхідно буде застосувати особливий стиль до певного елемента або конкретної групи елементів. Відповімо на запитання: як можна використовувати `class` і `id` для специфікування властивостей вибраних елементів, тобто як змінити колір конкретного заголовка окремо від інших заголовків на своєму web-сайті? Як групувати посилання по категоріях і задавати для кожної категорії особливий стиль?

CSS реалізує можливість присвоювати стилі не всім однаковим елементам сторінки, а вибірково. Для цього використовується параметр `class = «ім'я класу»` або ідентифікатор `id = «ім'я елемента»`, який присвоюється будь-якому елементу сторінки. Розглянемо ці можливості докладніше.

Групування елементів з допомогою параметра class. Цей параметр застосовується у разі, якщо необхідно створити однаковий стиль для декількох, але не всіх елементів сторінки (однакових або різних). Припустимо, у нас є два списки посилань: сорти винограду для білого й червоного вина. HTML-код може бути таким:

```
<p> Виноград для білого вина: </ p>
<ul>
<li> <a href="ri.htm"> Рислінг </a> </ li>
<li> <a href = "ch.htm"> Шардоне </a> </ li>
<li> <a href="pb.htm"> Піно Блан </a> </ li>
</ ul>
<p> Виноград для червоного вина: </ p>
<ul>
<li> <a href="cs.htm"> Каберне Совіньон </a> </ li>
<li> <a href="me.htm"> Мерло </a> < / li>
<li> <a href="pn.htm"> Піно Нуар </a> </ li>
</ ul>
```

Необхідно, щоб посилання на біле вино були жовтого кольору, на червоне вино – червоного, а інші посилання на цій самій сторінці залишалися синіми. Для цього поділимо посилання на дві категорії з допомогою присвоєння класу кожному посиланню атрибутом `class`.

Спробуємо встановити класи для попереднього прикладу:

```
<p> Виноград для білого вина: </ p>
<ul>
<li> <a href="ri.htm" class="whitewine">Рислінг </a> </li>
< li> <a href="ch.htm" class="whitewine">Шардоне </a> </li>
<li> <a href="pb.htm" class="whitewine">Піно Блан </a> </li>
</ ul>
<p> Виноград для червоного вина: </ p>
<ul>
<li> <a href="cs.htm" class="redwine">Каберне Совіньон </a> </li>
<li> <a href="me.htm" class="redwine">Мерло </a> </ li>
<li> <a href="pn.htm" class="redwine">Піно Нуар </ a> </ li>
</ ul>
```

Визначимо спеціальні властивості для посилань `whitewine` і `redwine`, відповідно:

```
a {color:blue;}
a.whitewine {color:#FFBB00;}
a.redwine {color:#800000;}
```

Як показано в прикладі, можна визначати властивості для елементів, що належать до певного класу, з допомогою `.ім'я_класса` у таблиці стилів документа.

Ідентифікація елемента з допомогою `id`. Крім групування елементів може знадобитися ідентифікація одного унікального елемента. Це можна реалізувати з допомогою атрибута `id`. Його особливість полягає в тому, що в документі не може бути більше одного елемента з конкретним `id`. Кожен `id` має бути унікальним. В інших випадках використовується атрибут `class`.

Наприклад, є заголовки документа, поділеного на глави або параграфи. Призначимо id кожному розділу:

```
<h1 id ="c1">Глава 1 </ h1>
<h2 id ="c1-1">Глава 1.1 </ h2>
<h2 id = "c1-2">Глава 1.2 </ h2>
<h1 id ="c2">Глава 2 </ h1>
<h2 id ="c2-1">Глава 2.1 </ h2>
<h3 id ="c2-1-2">Глава 2.1.2 </ h3>
```

Нехай заголовок «Глава 1.2» буде червоним. Це робиться відповідно до CSS, при цьому властивості конкретного елемента визначаються з допомогою #id у таблиці стилів документа:

```
# c1-2 {color:red;}
```

6.3 Групування елементів (span і div)

Елементи і використовуються для структурування документа, часто разом з атрибутами class і id.

Групування з допомогою . Елемент є нейтральним, який нічого не додає до вмісту документа, але в поєднанні з CSS може використовуватися для візуальних ефектів. Наприклад:

```
 <p> Хто рано лягає і рано встає, той буде здоровим, багатим і розумним </ p>
```

Щоб позначити деякі переваги з допомогою , кожному блоку присвоїмо параметр class, який потім визначимо в таблиці стилів:

```
 <p> Хто рано лягає і рано встає, той буде <span class = "benefit"> здоровим </span>, <span class = "benefit"> багатим </ span> і <span class = "benefit"> розумним </span>.</ p>
```

У CSS:

```
 span.benefit {color:red; }
```

Можна також використовувати id для визначення стилю -елементів. Для наведеного вище прикладу необхідно присвоїти унікальний id кожному з трьох -елементів.

Групування з допомогою . Тег застосовується для групування одного або більше блок-елементів. Наприклад, два списки президентів США згруповано за їх політичною належністю:

```
<div id = "democrats">
<ul>
<li> Франклін Д. Рузвелт </ li>
<li> Гаррі Трумен </ li>
<li> Джон Ф. Кеннеді </ li>
<li> Ліндон Б. Джонсон </ li>
<li> Джиммі Картер </ li>
<li> Білл Клінтон </ li>
</ ul>
</ div>

<div id = "republicans" >
<ul>
<li> Дуайт Д. Ейзенхауер </ li>
<li> Річард Ніксон </ li>
<li> Джералд Форд </ li>
<li> Роналд Рейган </ li>
<li> Джордж Буш </ li >
<li> Джордж У. Буш </ li>
</ ul>
</ div>
```

У таблиці стилів буде таке:

```
#democrats {background:blue;}
#republicans {background:red;}
```

6.4 Шрифти в CSS

Розглянемо основні властивості шрифтів: сім'я, вага, стиль, варіант, розмір, а також дізнаємося, за яким принципом браузер вибирає необхідний шрифт.

Шрифту в CSS відповідають такі властивості:

- font-family
- font-style
- font-variant
- font-weight
- font-size
- font

Властивість FONT-FAMILY визначає гарнітуру шрифту. Узагалі font family з англійської означає сім'ю шрифів. Розглянемо три основних сім'ї:

Serif – шрифти з характерними «зарубками», найбільш яскравим представником є Times New Roman.

Sans-serif – шрифти рубані, без «зарубок», наприклад Arial або Verdana.

Monospace – моноширинні шрифти (з однаковою відстанню між символами, на зразок друкарської машинки), такі як Courier New.

Правило: якщо в назві шрифту трапляються пропуски, то її слід брати в лапки.

При використанні властивості `font-family` на початку пишуть пріоритетніший шрифт, потім менш пріоритетний, а потім бажано писати ім'я сім'ї. Наприклад:

```
h1{font-family:verdana, arial,sans-serif;}
```

Усі заголовки першого рівня з прикладу вище будуть відображатися шрифтом Verdana. Якщо з якихось причин цей шрифт не встановлено на комп'ютері, то браузер поставить шрифт Arial, а якщо і його немає, то будь-який доступний шрифт із сім'ї Serif.

Властивість `FONT-STYLE` задає стиль шрифту, може набувати трьох значень:

- `normal` **–** звичайний;
- `italic` **–** курсивний;
- `oblique` **–** похилий.

Значення `italic` – це використання вбудованого в шрифт курсивного накреслення, оскільки майже кожен шрифт містить усі символи й букви в нормальному, напівжирному й курсивному виконанні. Значення `oblique` **–** це штучно створений курсив, тобто створений нахилом стандартних букв на певний кут. Наприклад:

```
h1 {font-family: verdana, arial, sans-serif;
    font-style:normal; }
h2 {font-family:verdana, arial, sans-serif;
    font-style:italic;}
```

Властивість `FONT-VARIANT` використовується для вибору варіанта написання букв нижнього регістра, може набувати двох значень:

– `normal` – значення за замовчуванням, текст відображається звичайним чином.

– `Small-caps` – букви нижнього регістра відображаються як зменшені великі.

За замовчуванням (тобто якщо взагалі не записувати цю властивість) текст буде відображатися звичайним шрифтом. Приклад запису стилю:

```
h1{font-family: verdana, arial, sans-serif;
font-variant:small-caps; }
h2{font-family: verdana, arial, sans-serif;}
```

Властивість FONT-WEIGHT визначає насиченість шрифту, тобто з її допомогою можна зробити шрифт жирним. Основні значення:

- `normal` – звичайний;
- `bold` – жирний.

Наприклад:

```
p {font-family: arial, verdana, sans-serif;}
td{font-family: arial, verdana, sans-serif;
   font-weight:bold; }
```

Властивість FONT-SIZE регулює розмір шрифту. Як одиницю виміру найкраще використовувати піксель, тому що це універсальний спосіб і в усіх браузерах результат буде однаковим. З допомогою цієї властивості можна перевизначати вид заголовків, що може бути корисним, наприклад, під час просування в пошукових системах, адже тексту в заголовку пошукові системи надають більшого значення, ніж звичайному.

Наприклад:

```
h1{font-family: arial, verdana, sans-serif;
   font-size:18px; }
h2{font-family: arial, verdana, sans-serif;
   font-size:36px;
   color: red;}
```

Скорочений запис. Властивість FONT. Усі перелічені вище властивості можна записати в скороченій формі. Це допомагає економити час і робити код стилів більш легким. При цьому потрібно записувати значення всіх властивостей через пропуск у такій послідовності: `font-style_font-variant_font-weight_font-size_font-family`.

Наприклад:

```
P {font-style:italic;
   font-variant:normal;
   font-weight:bold;
   font-size:30px;
   font-family:arial, sans-serif; }
```

Те ж саме в скороченій формі:

```
P {font:italic normal bold 30px arial, sans-serif; }
```

Якщо будь-яку властивість не вказано, то їй присвоюється значення за замовчуванням.

6.5 Текст у CSS

Розглянемо основні властивості CSS, які відповідають за форматування тексту:

- `text-align`;
- `text-decoration`;
- `text-indent`;
- `text-transform`;
- `letter-spacing`;
- `word-spacing`.

Властивість `TEXT-ALIGN` – це властивість вирівнювання тексту, аналогічна атрибуту `ALIGN`, який використовується в html і може набувати чотирьох значень:

- `left` – вирівнювання по лівому краю (значення за замовчуванням);
- `right` – вирівнювання по правому краю;
- `center` – вирівнювання по центру;
- `justify` – вирівнювання по ширині (по правому і лівому краях одночасно).

Наприклад:

```
h1 {text-align:center;}
h2 {text-align:left;}
h3 {text-align:right;}
p {text-align:justify;}
```

Властивість `TEXT-DECORATION` дає змогу оформляти текст певним чином. Розглянемо чотири основних значення цієї властивості:

- `none` – значення за замовчуванням, тобто додаткового оформлення не відбувається;
- `underline` – текст підкреслюється знизу;
- `overline` – риска підкреслення розташовується над текстом;
- `line-through` – текст перекреслюється.

Наприклад:

```
h1{text-align:center;
   text-decoration:underline;}
h2{text-align:center;
   text-decoration:overline;}
h3{text-align:center;
   text-decoration:line-through;}
```

Ця властивість часто використовується при заданні оформлення посилань, наприклад, щоб при наведенні курсора на посилання текст

підкреслювався.

Властивість TEXT-INDENT створює відступи першого рядка, тобто абзаци. Значення краще задавати в пікселях.

Наприклад:

```
h1{text-align: center;}
p {text-indent:40px;}
```

Однак розмір можна задати й у відсотках від загальної довжини базового рядка (рядка без відступу):

```
h1{text-align: center;}
p {text-indent:20%;}
```

Властивість TEXT-TRANSFORM змінює великі літери на маленькі або навпаки, може мати такі значення:

– `capitalize` – змінює перші букви кожного слова на великі. Наприклад: фраза «створіть сайт зараз» стане «Створіть Сайт Зараз».

– `uppercase` – змінює всі букви на великі. Наприклад: «текст у css» стане «ТЕКСТ У CSS»

– `lowercase` – змінює всі букви на маленькі. Наприклад: «TRANSFORM» стане «transform».

– `none` – не змінює регістр; це значення використовується за замовчуванням.

Наприклад:

```
h1 {text-transform: uppercase;}
li {Text-transform: capitalize;}
```

Властивість LETTER-SPACING змінює відстань між буквами. Значення краще вказувати в пікселях.

Наприклад:

```
h1{letter-spacing:10px;}
p{letter-spacing:4px;}
```

У прикладі вище для заголовків установлено інтервал між буквами 10 px, а для параграфів – 4px.

Властивість WORD-SPACING дає змогу змінювати відстань між словами, її значення також краще задавати в пікселях:

```
h1{word-spacing:20px;}
p{word-spacing:10px;}
```

6.6 Списки в CSS

Розглянемо основні властивості CSS, що відповідають за зовнішній вигляд списків:

- `list-style-type`;
- `list-style-image`;
- `list-style`.

Усі ці властивості є універсальними, тобто можуть застосовуватися як до впорядкованих, так і до невпорядкованих списків. У цьому й полягає перевага CSS, що можна з невпорядкованого списку зробити впорядкований і навпаки.

Властивість `LIST-STYLE-TYPE` дає змогу визначати вид маркера елементів списку. Це можуть бути цифри, букви, квадратики, кружечки та ін. Основні значення цієї властивості:

- `disk` – маркер у вигляді закрашеного кола;
- `circle` – маркер у вигляді незаповненого кола;
- `square` – маркер у вигляді зафарбованого квадрата;
- `decimal` – звичайні десяткові числа (1, 2, 3, 4, 5 і т. д.);
- `upper-roman` – великі римські цифри (I, II, III, IV, V і т. д.);
- `lower-roman` – маленькі римські цифри (i, ii, iii, iv, v і т. д.);
- `upper-alpha` – великі літери (A, B, C, D, E і т. д.);
- `lower-alpha` – малі літери (a, b, c, d, e і т. д.);
- `none` – маркера списку немає.

Наприклад:

```
ol { list-style-type:upper-roman;}
ul { list-style-type:square;}
```

Властивість `LIST-STYLE-IMAGE` дає змогу поставити замість маркера будь-яке зображення. Як значення вказується ключове слово `url` і потім в круглих дужках – шлях до зображення.

Наприклад:

```
ul {list-style-image:url(galka.gif);}
```

Цей приклад працює, якщо зображення знаходиться в тій же папці, де й CSS-файл. В іншому випадку необхідно вказати шлях до нього.

Скорочена форма `LIST-STYLE`. Перелічені вище властивості можна записати більш компактно. Для цього існує скорочений `варіант list-style`, тобто варіант з трьох рядків

```
ul {list-style-type:square;
    list-style-position:inside;
    list-style-image:url(galka.gif); }
```

раціонально замінити таким:

```
ul {list-style:square inside url(galka.gif)}
```

Властивості вказуються через пропуск, і послідовність у цьому випадку значення не має. Якщо будь-яку властивість не вказано, то їй присвоюється значення за замовчуванням.

6.7 Типи селекторів у CSS

Селектор призначено для того, щоб однозначно визначити елемент в html-документі, до якого буде застосуватися той чи інший стиль CSS:

- селектор за елементом;
- селектор за класом;
- селектор за `id`;
- контекстний селектор.

Селектор за елементом. Як селектор використовується безпосередньо ім'я html-елемента, до кого буде застосуватися певний стиль. Написавши клас, наприклад, для параграфа (`P`), усі параграфи на сторінці набудуть стилю цього класу:

```
P{font-family: arial, verdana, sans-serif; font-size: 12px}
```

Селектор за класом. Селектори класу дають змогу задавати різні стильні описи для одного й того самого html-елемента. Назва класу вказується після назви елемента й відрізняється точкою**.** Наприклад:

```
div.red {color: # FF0000; }
```

При створенні спеціального класу записують туди тільки ті властивості, які необхідно додати або змінити відносно базового стилю для цього елемента.

Селектор за `id`. Атрибут `id` задає html-елементу унікальний у межах документа ідентифікатор, який можна використовувати з різною метою, зокрема для задання стилю цього елемента. Значення атрибута `id` відокремлено від імені HTML-елемента символом #. Наприклад, нехай у зовнішньому файлі описано такі правила стилю:

```
h1 # special {color: green}
#comment {color: red}
```

Перше правило (зелений колір символів) буде зіставлено з елементом h1, значення атрибута id якого дорівнює special.

Друге правило (червоний колір символів) буде зіставлено з будь-яким елементом, значення атрибута id якого дорівнює comment.

Контекстний селектор. Контекстні селектори в CSS використовуються за наявності вкладених тегів. Їх застосування дає змогу визначити стилі тега, який вкладено в якийсь конкретний тег.

Наприклад:

```
B I { font-family: Arial;
      font-weight: bold;
      color: navy;
      font-style: italic; }
```

У прикладі жирний курсивний шрифт Arial синього кольору буде застосовано до тега <I>, коли він знаходитиметься всередині тега <B>.

Практичні завдання

1 Зробіть усі абзаци <p> червоного кольору.

2 Зробіть усі заголовки першого рівня <h1> зеленого кольору.

3 Перетворить упорядкований список на невпорядкований.

4 Заголовки першого рівня вирівняйте по центру; заголовки другого рівня – по лівому краю, а заголовки третього рівня – по правому краю.

5 З допомогою id виділіть перший заголовок на сторінці, щоб він був червоного кольору й підкресленим.

6 Створіть селектор, який вибере всі заголовки другого рівня <h2> усередині тега .

7 БЛОКОВА МОДЕЛЬ У CSS

У html розрізняють блокові й рядкові елементи, причому блокові елементи – це окрема структурна одиниця, яка завжди відокремлюється абзацним відступом, тобто не можна розташувати два блокових елемента на одному рядку. Прикладом блоків у html можуть бути елементи h1 – h6, p, div. Отже, блоки в CSS є самостійною структурною одиницею, що має форму прямокутника. Кожен елемент у дереві елементів документа – це самостійний блок. З цього випливає, що в CSS є блоки, які структурно відокремлені від інших, а є рядкові блоки, які можуть знаходитися всередині структурних блоків. Однак усі вони мають однакову структуру (рисунок 7.1).

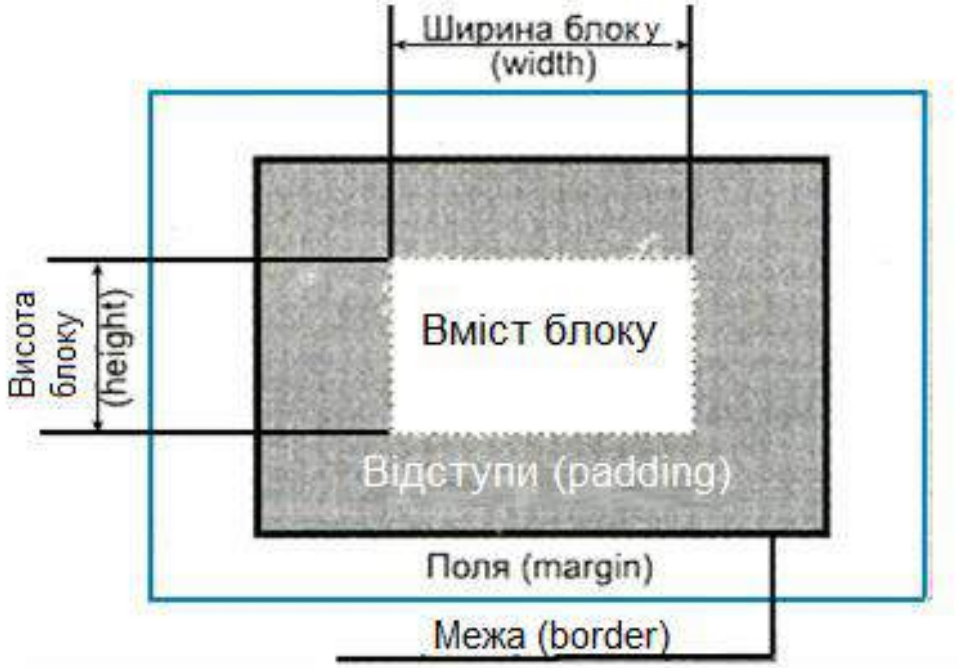


Рисунок 7.1 – Блокова структура в CSS

Кожен такий блок обов'язково має інформаційну частину, або вміст, яким може бути текст, зображення або інша інформація. Цю частину блоку називають його контентом, або вмістом. Наприклад, для елемента `P` вмістом блоку є текст абзацу.

Блок також може мати відступи `(padding)` – додатковий вільний простір навколо його межі. Згідно зі специфікацією CSS поля, межі й відступи не належать до ширини блоку. Таким чином, указуючи ширину блоку, задають ширину лише тієї його частини, яку відведено для вмісту.

Безпосередньо за полями проходить межа блоку `(border)`, яка може мати певну товщину і стиль ліній. Ширина блоку може бути довільною – від нульової (межи в цьому випадку не видно) до довільно заданої в одиницях виміру довжини. Стиль ліній може бути різним: від простої лінії до об'ємної. Крім того, межа може мати довільний колір.

Навколо області контенту можуть бути порожні області, не зайняті вмістом, які називають полями (`margin`). З огляду на дизайн, поля забезпечують для вмісту блоку естетично більш привабливий вигляд. За наявності полів певного розміру текст не прилягає впритул до меж блоку. Можна провести аналогію з полями, що встановлюються під час друку документів на папері. Якби полів не було, то текст починався би зразу з краю аркуша. За наявності полів є незайняті текстом області вздовж країв аркуша паперу, і надрукований текст у цьому випадку читати приємніше і зручніше (рисунок 7.2).

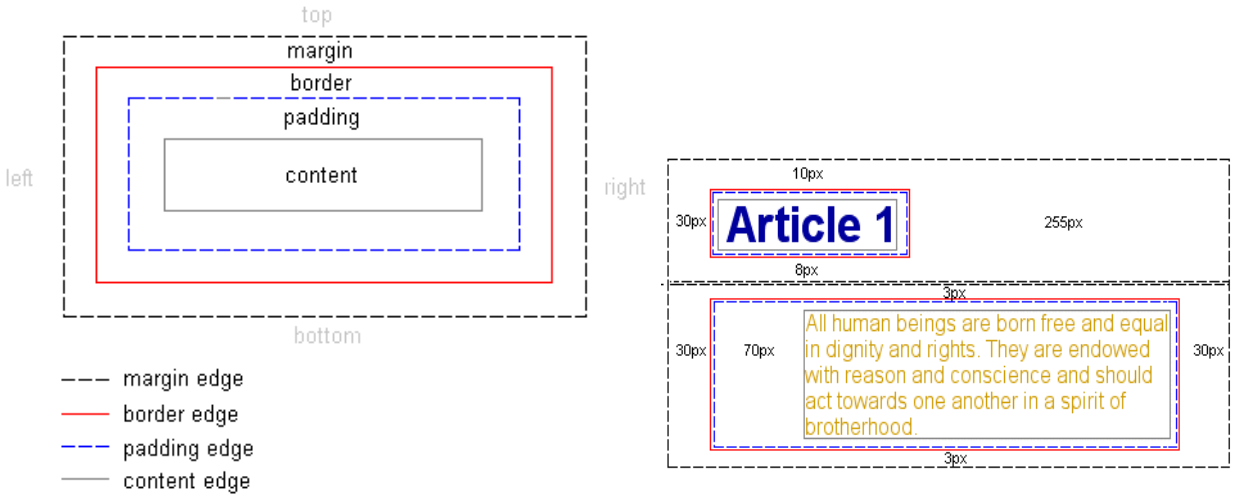


Рисунок 7.2 – Приклади полів у CSS

7.1 Рамки в CSS (BORDER)

Основні властивості рамок у CSS:

- `border-width`;
- `border-color`;
- `border-style`;
- Скорочена форма – `border`.

Властивість BORDER-WIDTH задає товщину рамки. Значення зазвичай указується в пікселях, але також можна вказувати ключовими словами: `thin` (2px), `medium` (4px) і `thick` (6px). На рисунку 7.3 показано, який вигляд на екрані браузера будуть мати лінії товщиною від 1 до 10 пікселів.

Рисунок 7.3 – Ширина ліній товщиною від 1 до 10 пікселів

Властивість BORDER-COLOR визначає колір рамки. Значення кольору задається звичайним чином, наприклад: «`#ff3344`» або «`gold`».

Властивість BORDER-STYLE визначає вид рамки. На рисунку 7.4 показано вісім основних значень цієї властивості. Усі рамки в прикладі виконано кольором `gold` і мають ширину 6 px.

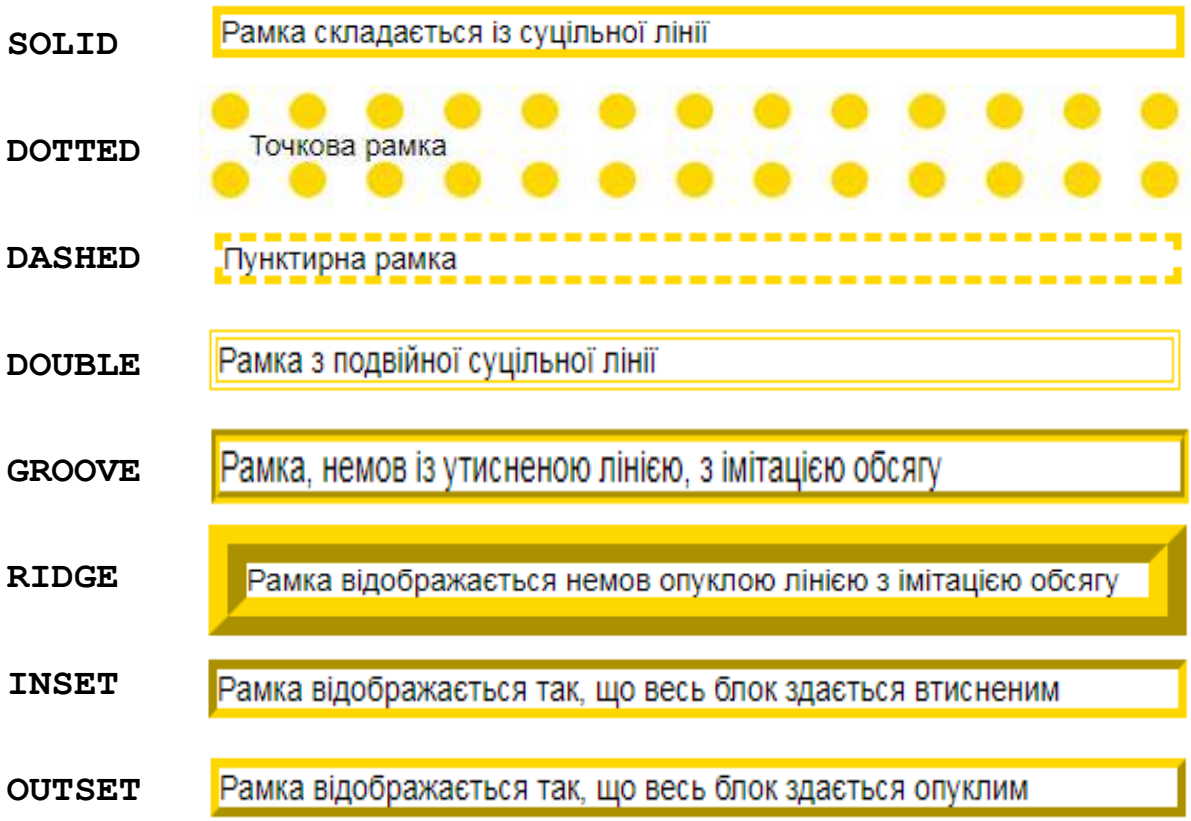


Рисунок 7.4 – Відображення дїії основних значень властивості BORDER-STYLE

Мінімальна ширина рамки типу `double` має становити 3 px, інакше її буде відображено некоректно.

Приклади стилів:

```
h1 {border-width: 4px; border-style: dotted; border-color: red; }
h2 {border-width: 18px; border-style: inset; border-color: red; }
P {border-width: 2px; border-style: solid; border-color: blue; }
```

СКОРОЧЕНА ФОРМА BORDER. Як і інші властивості, рамка має скорочену форму – `border`. Ці властивості відокремлюються одна від одної пробілами – на початку пишуть товщину, потім стиль і колір.

Наприклад:

```
h1 {border:30px outset red;}
h2 {border:20px dashed gold;}
P {border:1px dotted blue}
```

7.2 Поля (margin) і відступи (padding)

MARGIN (поля) – це відстань від межи (рамки) блоку до найближчих елементів, або (якщо їх немає) до країв документа.

PADDING (відступи) – внутрішня відстань між межею (рамкою) і вмістом блоку.

Приклад: створимо три стилі для трьох різних параграфів з різними значеннями margin і padding:

```
    .p1{background-color:  #  FFE446;  border:  1px  solid  red;
margin:70px; }
    .p2{background-color:  #  FFE446;  border:  1px  solid  red;
padding:70px;  }
    .p3{background-color:  #  FFE446;  border:  1px  solid  red;
margin:50px; padding:20px; }
```

Додаючи ключові слова top, right, bottom та left, можна регулювати відступи й поля відповідно зверху, справа, знизу, зліва.

```
    p  {margin-top:50px;  margin-right:50px;  margin-bottom:50px;
margin-left:150px; }
```

Можливим є також такий варіант запису: значення записують за ходом годинникової стрілки – верхнє, праве, нижнє, ліве.

Наприклад:

```
    p {margin:50px 50px 50px 150px; }
```

Відмінності:

- відступи (padding) розташовують усередині блоку, а поля (margin) – за його межами;
- фон блоку й фонове зображення поширюються тільки на відступи, але не на поля, тобто поля завжди є прозорими, і крізь них просвічує фон батьківського елемента.

7.3 Висота (height) і ширина (width) блоків

За замовчуванням висота й ширина блоків визначаються автоматично, тобто чим більше тексту (або іншого вмісту), тим ширшим і вищим буде блок. З допомогою технології CSS можна задавати необхідні ширину й висоту блоків:

- властивість HEIGHT установлює висоту блоку;
- властивість WIDTH установлює ширину блоку.

Зазвичай як блоки в CSS використовують елемент DIV**.** Однак властивість ширини й висоти можна застосувати й до параграфів, списків та ін.

Приклади:

1 Ширина є фіксованою і становить 300 пікселів, а висота встановлюється за замовчуванням залежно від вмісту:

```
.box1 {width:300px; border: 1px solid red; background: # FFE446;}
```

2 Висота є фіксованою, а ширина розтягується по вмісту:

```
.box2 {height:300px; border: 1px solid red; background: #FFE446;}
```

3 Висота і ширина є фіксованими:

```
.box3 {width:300px; height:600px; border: 1px solid red;
background: # FFE446; }
```

Згідно зі специфікацією CSS поля, межі й відступи не входять до ширини блоку. Таким чином, указуючи ширину блоку, задається ширина лише тієї його частини, яку відведено для вмісту.

7.4 Позиціонування блоків

Існують дві основні моделі позиціонування:

- абсолютне (POSITION: ABSOLUTE;);
- відносне (POSITION: RELATIVE;).

Абсолютне позиціонування (POSITION: ABSOLUTE;). У технології CSS властивість позиціонування позначається як POSITION**,** а щоб вказати, що це позиціонування є абсолютним, пишуть значення ABSOLUTE**.** Використовуючи ключові слова top, right, bottom та left, указують необхідні координати, які відлічуються від країв вікна браузера.

Рамка боксу на рисунку 7.5 позначає, звідки відлічуються координати.

Бокс з абсолютним позиціонуванням розташовується за заданими координатами, а з того місця, де він має бути, він автоматично видаляється. Приклад стилю:

```
.smile {position:absolute; bottom:300px; left:100px; }
```

Для фіксації блоку, тобто щоб він не прокручувався разом з основним вмістом, необхідно вказати значення FIXED:

```
.smile {position:fixed; bottom:300px; left:100px; }
```

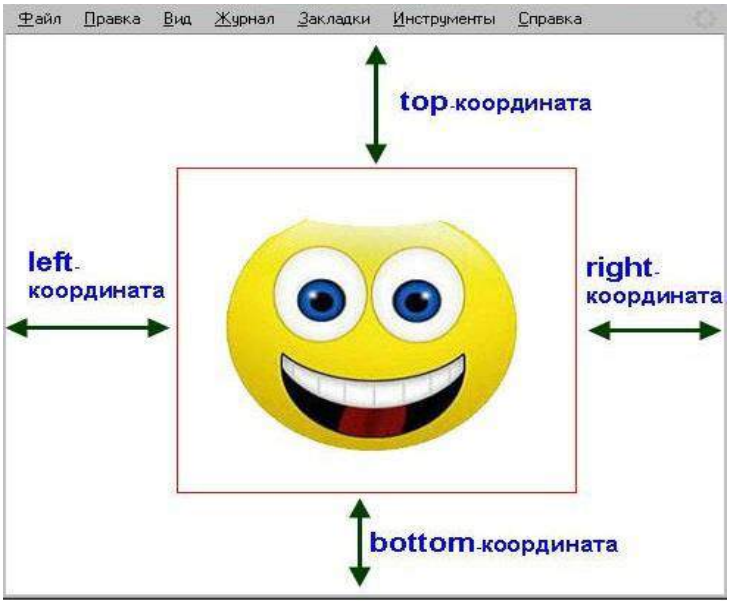


Рисунок 7.5 – Абсолютне позиціонування у CSS

Відносне позиціонування (`POSITION: RELATIVE;`). При відносному позиціонуванні бокс зміщується, але його початкове місце ніщо не заповнює. Координати на відміну від абсолютного позиціонування відлічуються від попереднього положення блоку.

Наприклад (рисунок 7.6):

```
    .smile{ border:1px solid red; position:relative; top:200px;
left:100px; }
```

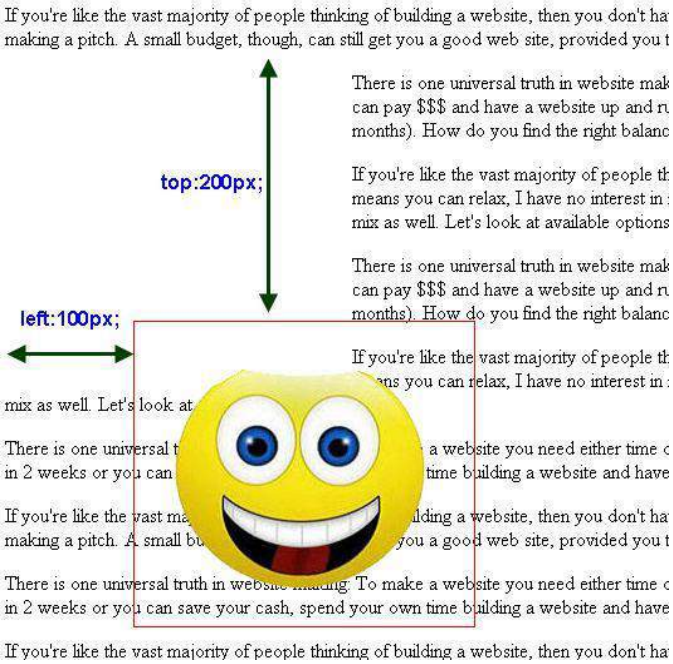
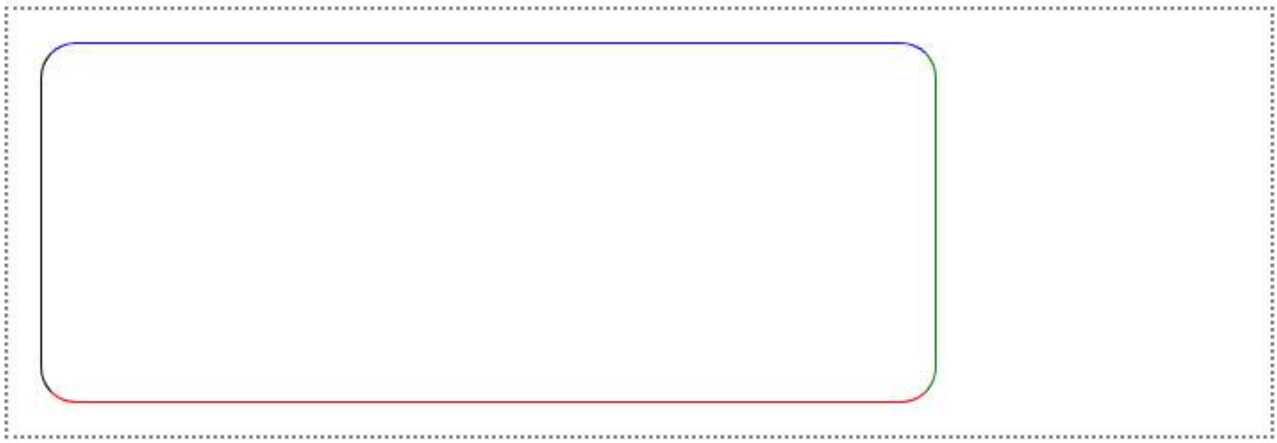


Рисунок 7.6 – Відносне позиціонування у CSS

Практичні завдання

1 Повторіть сторінку за цим зразком:



2 Створіть три блоки. Для основного блоку встановіть фіксоване позиціювання, зміщення зверху – 30 px, зліва – 35 px, ширина – 300 px, висота – 400 px, внутрішні відступи – 15 px. Для другого блоку: ширина – 70 px, висота – 50 px, внутрішні відступи – 5 px, ширина межі – 2 px, блок «плаває» зліва. Для третього блоку: ширина – 70 px, висота – 50 px, внутрішні відступи – 5 px, ширина межі – 2 px, блок «плаває» праворуч.

8 СУЧАСНІ МЕТОДИ ВЕРСТАННЯ

8.1 Одновимірна макетна модель верстання Flex

CSS flexbox (Flexible Box Layout Module) — модуль макета гнучкого контейнера, що є способом компонування елементів. Flexbox складається з flex-контейнера – батьківського контейнера і flex-елементів – дочірніх блоків. Дочірні елементи можуть розташовуватися в рядок або стовпчик, а вільний простір розподіляється між ними різними способами.

Flexbox дає змогу вирішувати такі завдання:

- розташовувати елементи в одному з чотирьох напрямків: зліва направо, справа наліво, зверху вниз або знизу вверх;
- перевизначати порядок відображення елементів;
- автоматично визначати розміри елементів таким чином, щоб вони вписувалися в доступний простір;
- вирішувати проблему з горизонтальним і вертикальним центруванням;

– переносити елементи всередині контейнера, не допускаючи їх переповнення;

– створювати стовпчики однакової висоти;

– створювати притиснутий до низу сторінки «підвал» сайта.

Модель flexbox-розмітки пов'язана зі значенням CSS-властивості `display` батьківського html-елемента, що містить дочірні блоки. Для керування елементами з допомогою цієї моделі необхідно встановити властивість `display` батьківського контейнера:

```
.container { display: flex; /* or inline-flex */ }
```

Flex-контейнер не є блоковим, тому для внутрішніх блоків не працюють такі CSS-властивості, як `float`, `clear`, `vertical-align`. На flex-контейнер також не впливають властивості `column`, що створюють стовпчики в тексті й псевдоелементи `::first-line` і `::first-letter`.

Компонування з `flexbox` виконується по двох осях: основній (`main axis`) і поперечній (`cross axis`). Основна вісь визначається властивістю `flex-direction`, а поперечна розташовується перпендикулярно до неї.

При заданні `row` або `row-reverse` (рисунок 8.1) основна вісь буде розташовуватися паралельно до рядків, а додаткова – перпендикулярно до них.

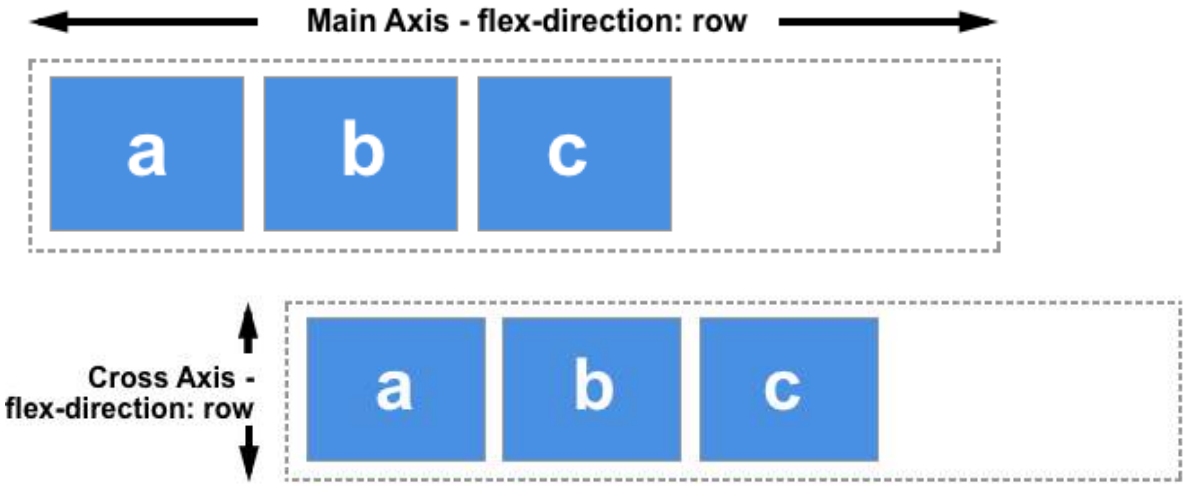


Рисунок 8.1 – Компонування з `flexbox`

Flexbox-властивості: `flex-direction`, `justify-content`, `align-items`, `flex-wrap`, `flex-flow` (задаються для батьківського контейнера), `order`, `flex-grow`, `flex-shrink`, `flex-basis`, `flex`, `align-self` (задаються для дочірніх елементів `flex`).

Властивість flex-direction має чотири можливі значення: `row`, `row-reverse`, `column`, `column-reverse` (таблиця 8.1).

Таблиця 8.1 – Значення для `flex-direction`

Значення	Опис
`row`	Головна вісь спрямована зліва направо. Елементи розташовані в рядок і за замовчуванням притиснуті до лівого краю (це регулюється властивістю `justify-content`), їх нумерація має звичайний порядок – зліва направо
`row-reverse`	Головна вісь спрямована справа наліво. Елементи розташовані в рядок і за замовчуванням притиснуті до правого краю (див. `justify-content`), їх нумерація має зворотний порядок – справа наліво
`column`	Головна вісь спрямована зверху вниз. Елементи розташовані в стовпчик і за замовчуванням притиснуті до верху (див. `justify-content`), їх нумерація має звичайний порядок – зверху вниз
`column-reverse`	Головна вісь спрямована знизу вверх. Елементи розташовані в стовчик і за замовчуванням притиснуті до низу (див. `justify-content`), їх нумерація має зворотний порядок – знизу вверх

Властивість `justify-content` задає вирівнювання елементів уздовж головної осі.

Можливі значення `justify-content: flex-start, flex-end, center, space-between, space-around` (їх опис наведено в таблиці 8.2).

Таблиця 8.2 – Значення для `justify-content`

Значення	Опис
`flex-start`	Блоки притиснуті до початку головної осі
`flex-end`	Блоки притиснуті до кінця головної осі
`center`	Блоки розташовані по центру головної осі

Продовження таблиці 8.2

Значення	Опис
`space-between`	Блоки розподілені вздовж головної осі, при цьому перший елемент притиснутий до початку осі, а останній – до кінця
`space-around`	Блоки розподілені вздовж головної осі, при цьому між першим блоком і початком осі, останнім блоком і кінцем осі такий самий проміжок, як і між іншими блоками. Однак вони не є однаковими: проміжки сумуються і між двома блоками відстань є в два рази більшою, ніж між блоком і початком/кінцем осі

Властивість `align-items` задає вирівнювання елементів уздовж поперечної осі.

Можливі значення `align-items: flex-start, flex-end, center, baseline, stretch` (їх опис наведено в таблиці 8.3).

Таблиця 8.3 – Значення для `align-items`

Значення	Опис
`flex-start`	Блоки притиснуті до початку поперечної осі
`flex-end`	Блоки притиснуті до кінця поперечної осі
`center`	Блоки розташовані по центру поперечної осі
`baseline`	Елементи вирівнюються по своїй базовій лінії. Базова лінія (англ. Baseline), або лінія шрифту, – це уявна лінія, яка проходить по нижньому краю символів без урахування звисання, наприклад, як у літер 'ц', 'д', 'р', 'щ'
`stretch`	Блоки розтягнуті, займають усе доступне місце по поперечній осі, при цьому все ж ураховуються `min-width` і `max-width`, якщо їх задано. Якщо ж задано ширину й висоту елементів, то `stretch` буде проігноровано

Властивість `flex-wrap` задає багаторядкове розміщення блоків по головній осі.

Можливі значення `flex-wrap: nowrap, wrap, wrap-reverse` (їх опис наведено в таблиці 8.4).

Таблиця 8.4 – Значення для `flex-wrap`

Значення	Опис
`nowrap`	Однорядковий режим – блоки розташовуються в один рядок
`wrap`	Блоки розташовуються в декілька рядків, якщо не поміщаються в один
`wrap-reverse`	Те ж саме, що й `wrap`, але блоки розташовуються в іншому порядку (спочатку останній, потім перший)

Властивість `flex-flow` – це скорочення для `flex-direction` і `flex-wrap`. Значення за замовчуванням – `row nowrap`.

Наступні властивості задаються на рівні окремого елемента.

Властивість `align-self` задає вирівнювання вздовж поперечної осі для окремо взятого flex-блоку. По суті `align-self` – це властивість `align-items`, але для конкретного блоку.

Можливі значення `align-self: auto, flex-start, flex-end, center, baseline, stretch` (їх опис наведено в таблиці 8.5).

Таблиця 8.5 – Значення для `align-self`

Значення	Опис
`flex-start`	Блок притиснутий до початку поперечної осі
`flex-end`	Блок притиснутий до кінця поперечної осі
`center`	Блок розташований по центру поперечної осі
`baseline`	Блок вирівнюється по своїй базовій лінії. Базова лінія (англ. `Baseline`) або лінія шрифту – це уявна лінія, що проходить по нижньому краю символів без урахування звисань, наприклад, як у літер 'ц', 'д', 'р', 'щ'
`auto`	Блок буде вирівняно так, як задано у властивості `align-items` (за замовчуванням)

Продовження таблиці 8.5

Значення	Опис
`stretch`	Блок розтягнутий, займає все доступне місце по поперечній осі, при цьому все ж ураховуються `min-width` і `max-width`, якщо їх задано. Якщо ж задано ширину й висоту елемента, то `stretch` буде проігноровано

Властивість `order` задає порядок прямування окремо взятого flex-блока всередині `flex`-контейнера, `order` – додатне або від'ємне число.

Чим меншим є число, тим раніше буде стояти елемент, незалежно від місцезнаходження в HTML-коді відносно інших елементів.

Властивість `flex-basis` задає розмір конкретного flex-блоку до застосування до нього інших `flex`-властивостей:

```
flex-basis: будь-які CSS-одиниці (і відсотки) | auto;
```

Властивість `flex-grow` визначає те, на скільки окремий flex-блок може бути більшим від сусідніх елементів, якщо це необхідно:

```
flex-grow: додатне число;
```

Наприклад, якщо всі `flex`-блоки всередині `flex`-контейнера мають

```
flex-grow:1
```

то вони будуть однакового розміру. Якщо один із них має

```
flex-grow:2
```

то він буде у два рази більшим, ніж усі інші.

Властивість `flex-shrink` визначає, наскільки `flex`-блок буде зменшуватися відносно сусідніх елементів усередині `flex`-контейнера, якщо вільного місця замало:

```
flex-shrink: додатне число;
```

Наприклад, якщо всі `flex`-блоки всередині `flex`-контейнера мають

```
flex-shrink: 1
```

то вони будуть однакового розміру. Якщо один із них має

```
flex-shrink: 2
```

то він буде в два рази меншим, ніж усі інші.

Властивість `flex` – скорочення для `flex-basis`, `flex-shrink` і `flex-grow`.

Приклад використання наведених вище властивостей (рисунок 8.2):

```
<!doctype html>
<html><head>
 <meta charset='utf-8'>
 <title>flex example</title>
<style type="text/css">
 body {margin: 20px; font-size:24px; line-height:30px; color:#fff; }
 .flex-container {border: 1px solid #000; display: flex; min-height:100px; flex-direction: row; justify-content: center; align-items: flex-end; }
 .box {text-align: center; width: 100px; margin: 10px; }
 .box-1 {width:50px; background: #ff0000; }
 .box-2 {width:170px; align-self:stretch; background: #066; order: 1; }
 .box-3 {align-self:center; width:90px; background: #008000; }
 .box-4 {align-self:flex-start; background: #0000ff; order: 1; }
</style>
</head><body>
<div class="flex-container"> <div class="box box-1">div 1</div> <div class="box box-2">div 2</div> <div class="box box-3">div 3</div> <div class="box box-4">div 4</div> </div>
</body></html>
```

Рисунок 8.2 – Приклад будування макета на `flexbox`

Інші приклади з описом використання `flexbox` можна подивитися за посиланнями у бібліографічному списку.

8.2 Двовимірна макетна модель верстання Grid

На відміну від `Flexbox` `Grid` дає змогу керувати елементами в двовимірному просторі. Як і для `Flexbox`, усі елементи поміщаються всередині батьківського контейнера. `Grid`-контейнер визначається властивістю `display: grid` і являє собою набір перетинних горизонтальних (вісь стовпчиків) і вертикальних (вісь рядків) `grid`-ліній. Ці лінії поділяють простір на `grid`-області, у які можна поміщати `grid`-елементи.

`Grid`-контейнер – базовий елемент, усередині якого поміщаються інші елементи; задається властивістю `display: grid`.

`Grid`-елементи (`grid-items`) – діти (лише **прямі** нащадки) `grid`-контейнера.

`Grid`-лінії – горизонтальні й вертикальні роздільники `grid`-контейнера. Ці лінії знаходяться по обидва боки від стовпчика або рядка.

`Grid`-смуга – смуга, обмежена парою сусідніх `grid`-ліній. Вертикальні `grid`-смуги – це `grid`-стовчики (аналог стовпчиків таблиці), горизонтальні – рядки (аналог рядків).

`Grid`-комірка – це найменша неподільна одиниця `grid`-контейнера, на яку можна посилатися при позиціонуванні `grid`-елементів; утворюється на перетині `grid`-рядка і `grid`-стовчика; є аналогом комірки таблиці.

`Grid`-область – це простір усередині `grid`-контейнера, у який можна помістити один або більше `grid`-елементів. Цей елемент може складатися з однієї або більше `grid`-комірок (рисунок 8.3).

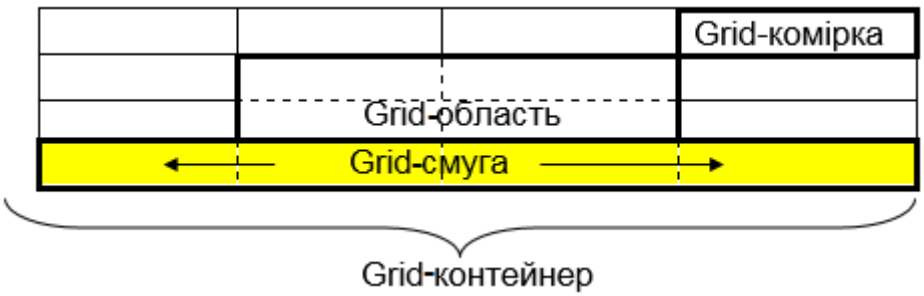


Рисунок 8.3 – `Grid`-елементи

`Grid`-інтервал – це простір між двома суміжними `grid`-лініями (вертикальними або горизонтальними). Аналог `border-spacing` у таблиці. Задається властивостями `grid-column-gap` і `grid-row-gap`.

Властивості Grid-контейнера:

display; grid-template-columns; grid-template-rows; grid-template-areas; grid-template; grid-column-gap; grid-row-gap; grid-gap; justify-items; align-items; place-items; justify-content; align-content; place-content; grid-auto-columns; grid-auto-rows; grid-auto-flow; grid.

Властивості Grid-елемента:

grid-column-start; grid-column-end; grid-row-start; grid-row-end; grid-column; grid-row; grid-area; justify-self; align-self; place-self.

Властивість Display визначає цей елемент як контейнер grid і встановлює новий контекст форматування grid для його вмісту. Можливі значення:

- grid – генерує grid як блоковий елемент;
- inline-grid – генерує grid як рядковий елемент.

Властивості grid-template-columns; grid-template-rows визначають стовчики й рядки сітки зі списком значень, розділених пропусками.

Властивість grid-template-areas визначає шаблон grid, посилаючись на назви областей grid, які вказано у властивості grid-area.

Властивість grid-template – це скорочення для установки grid-template-rows, grid-template-columns і grid-template-areas.

Властивості grid-column-gap; grid-row-gap визначають розмір ліній grid.

Властивість grid-gap – це скорочений варіант grid-column-gap; grid-row-gap.

Властивість justify-items вирівнює об'єкти grid уздовж рядка (на відміну від align-items, яка вирівнює вздовж осі стовчика). Це значення застосовується до всіх об'єктів grid усередині контейнера.

Властивість align-items – вирівнює об'єкти уздовж осі колонки (на відміну від justify-items). Це значення застосовується до всіх об'єктів grid усередині контейнера.

Властивість place-item установлює одночасно align-items і justify-items.

Властивість justify-content вирівнює grid уздовж рядка (на відміну від align-content, який вирівнює сітку вздовж осі стовчиків).

Властивість align-content вирівнює grid уздовж осі стовчиків (на відміну від justify-content).

Властивість place-content установлює одночасно align-content і justify-content.

Властивості grid-auto-columns і grid-auto-rows визначають розмір будь-якої grid-смуги, яку було згенеровано автоматично.

Властивість grid-auto-flow застосовується в разі, якщо є об'єкти, які явно не поміщаються в grid, тоді алгоритм починає розміщувати елементи автоматично. Ця властивість контролює, як працює алгоритм автоматичного розміщення. Можливі значення – row (по стовчиках за замовчуванням), column (по рядках), dense (з урахуванням попередніх пустих місць, куди може бути поміщено невеликий елемент, що надійшов пізніше).

Властивість grid установлює одночасно grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns і grid-auto-flow.

Властивості grid-column-start, grid-column-end grid-row-start і grid-row-end визначають розташування об'єкта grid у grid, посилаючись на певні лінії grid. Grid-column-start/grid-row-start – це лінія, де починається елемент, grid-column-end/grid-row-end – де закінчується елемент.

Властивості grid-column, grid-row – це скорочення для grid-column-start і grid-column-end, та grid-row-start і grid-row-end відповідно.

Властивість grid-area дає ім’я елементу, щоб на нього міг посилатися шаблон, створений з допомогою властивості grid-template-areas.

Властивість justify-self вирівнює елемент grid усередині комірки вздовж осьової лінії (рядка) (на відміну від align-self, який вирівнює вздовж осі блоку (стовчика)). Це значення застосовується до об'єкта grid усередині окремої комірки.

Властивість align-self вирівнює елемент сітки всередині комірки вздовж осі блоку (стовчика) (на відміну від justify-self). Це значення застосовується до вмісту всередині одного об'єкта grid.

Властивість place-self установлює align-self і justify-self.

Приклад використання grid (рисунок 8.4):

```
<!doctype html>
<html>
<head> <meta charset='utf-8'> <title>grid example</title>
 <style type="text/css">
 body {margin: 5px; font-size:24px; color:#fff; background: white}
 .grid {display: grid; grid-gap:20px; grid-template: 40px 40px 40px / 1fr 1fr 1fr 1fr;
        grid-template-areas:
        "a a a b"
        "a a a b"
        "c d d d"; }
```

```
          .box {border-radius: 5px;  padding: 5px; } .item-1 {grid-
area: a; background: #f00; } .item-2 {grid-area: b; background: #066;}
.item-3 {grid-area: d;background: #008000; } .item-4 {grid-area: c;
background: #00f; }
     </style>
    </head>
    <body> <div class="grid"> <div class="box item-1">div 1</div> <div
class="box item-2">div 2</div> <div class="box item-3">div 3</div>
<div class="box item-4">div 4</div> </div> </body> </html>
```

Рисунок 8.4 – Приклад будування макета з `grid`

8.3 Верстання багатостовпчикового тексту з допомогою `Column`

З допомогою властивості `Column` можна розбити контент на стовпчики й вирівняти їх за висотою. Стовчики можуть містити заголовки, текст, таблиці, картинки й будь-які інші `inline`-елементи.

Як працюють CSS-стовпчики:

1 Задається кількість стовпчиків з допомогою властивості `column-count`.

2 Браузер розбиває вміст блоку, для якого задано цю властивість, таким чином, щоб стовпчики були однаковими за висотою. При зміненні ширини блоку текст знову розподілиться таким чином (додається або прибирається з одного стовпчика до іншого), щоб висота стовпчиків знову була однаковою.

Примітка. CSS3 `columns` розбиває вміст абзацу, при цьому текст плавно «перетікає» з однієї колонки в іншу, тобто умовно з одного елемента робиться три. Це – суттєва відмінність від `flex`-блоку, де вміст завжди буде одним і тим самим. `Flexbox` керує положенням елементів на сторінці, вирівнюючи їх у заданому напрямку, змінюючі їх розміри, не змінюючи при цьому структуру їх внутрішнього вмісту.

8.4 Верстання з допомогою CSS-фреймворків

З часом у кожного верстальника накопичується певний набір готових рішень і шаблонів, які він використовує у своїх проектах, а необхідність співпраці та спрощення супроводу потребує стандартизації шаблонів. Так, стали створюватися бібліотеки та фреймворки CSS. Подальші кроки з освоєння верстання можна спрямувати на вивчення таких фреймворків.

CSS-Framework (CSS FW) – фреймворк, створений для полегшення роботи веб-розробника і веб-дизайнера, пришвидшення розроблення й запобігання максимальній кількості помилок.

Одним із найпопулярніших CSS-фреймворків, який можна рекомендувати до вивчення й використання, є Bootstrap – клієнтський фреймворк, призначений для створення веб-сайтів і веб-додатків, який містить шаблони CSS і HTML, а також додаткові розширення JavaScript і спрощує розроблення динамічних веб-сайтів і веб-додатків.

Ознайомлення з фреймворком Bootstrap виходить за межі цього посібника, наведемо лише основні його інструменти:

- сітки (`grid`) – наперед задані, готові до використання стовпчики;
- шаблони (`template`) – фіксовані або адаптивні шаблони сторінок;
- типографіка (`typography`) – опис і визначення класів для шрифтів, таких як шрифти для коду, цитат тощо;
- мультимедіа (`media`) – засоби керування зображеннями й відео;
- таблиці (`table`) – засоби оформлення таблиць, які зокрема забезпечують сортування;
- форми (`form`) – класи для оформлення як форм, так і деяких подій;
- навігація (`nav`, `navbar`) – класи для оформлення вкладок, сторінок, меню й панелей навігації;
- сповіщення (`alert`) – класи для оформлення діалогових вікон, підказок і спливних вікон;
- іконочний шрифт (`icon font`) – набір іконок у вигляді шрифту, який містить майже 500 компонентів.

Практичні завдання

1 Засобами `flexbox` реалізувати макет сайта (рисунок 8.5), що містить вирівняний по центру контейнер, усередині якого знаходяться:

- шапка;
- основний розділ;
- бічна панель;

– підвал.

2 Розмістити чотири основні розділи макета.

3 Зробити сторінку адаптивною (бічну панель опустити нижче від основного вмісту на екранах смартфонів) (рисунок 8.6).

4 Вирівняти вміст шапки: навігація – зліва, кнопка – праворуч.

5 Виконати попереднє завдання розділу (реалізувати макет сайта за рисунками 8.5 і 8.6) з допомогою `grid`-моделі.

6 Оцінити переваги й недоліки кожного способу (`flex` і `grid`).

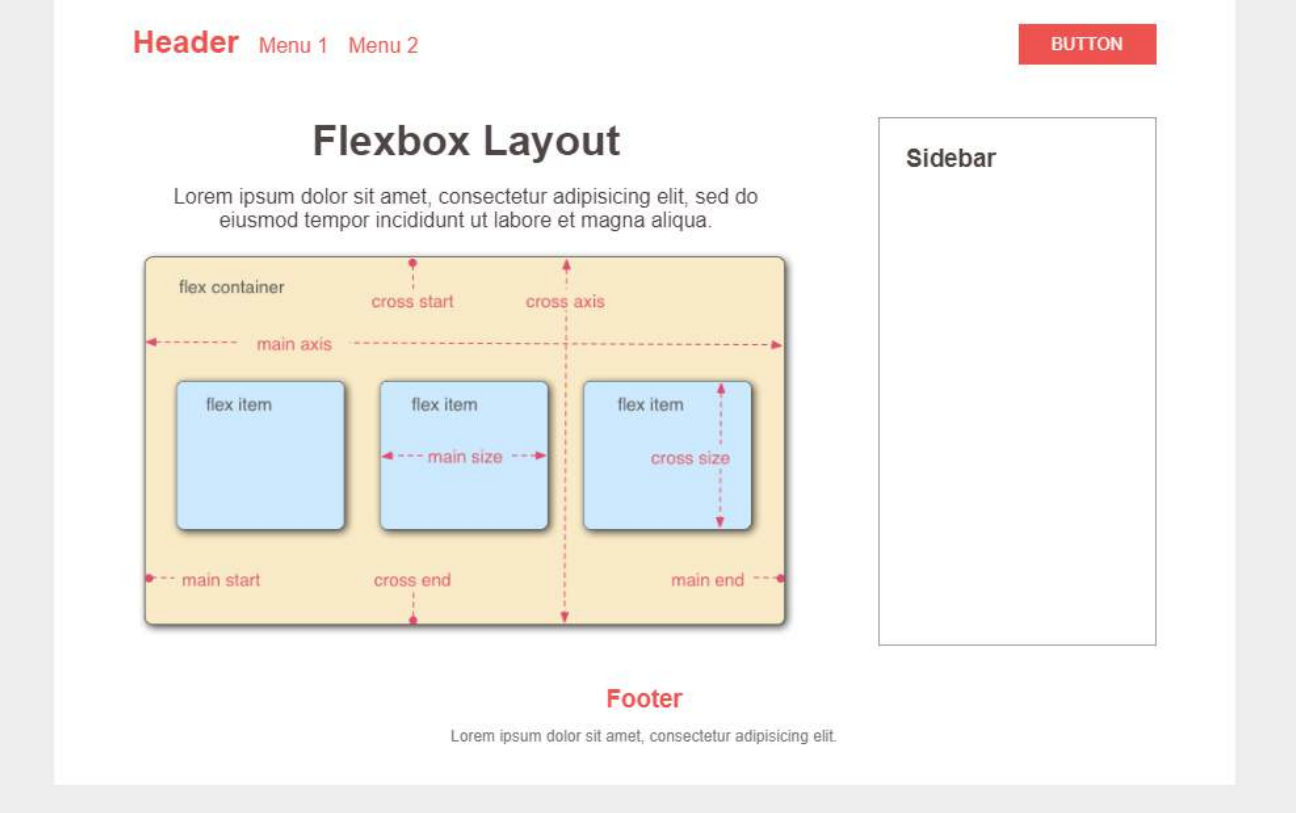


Рисунок 8.5 – Макет сайта

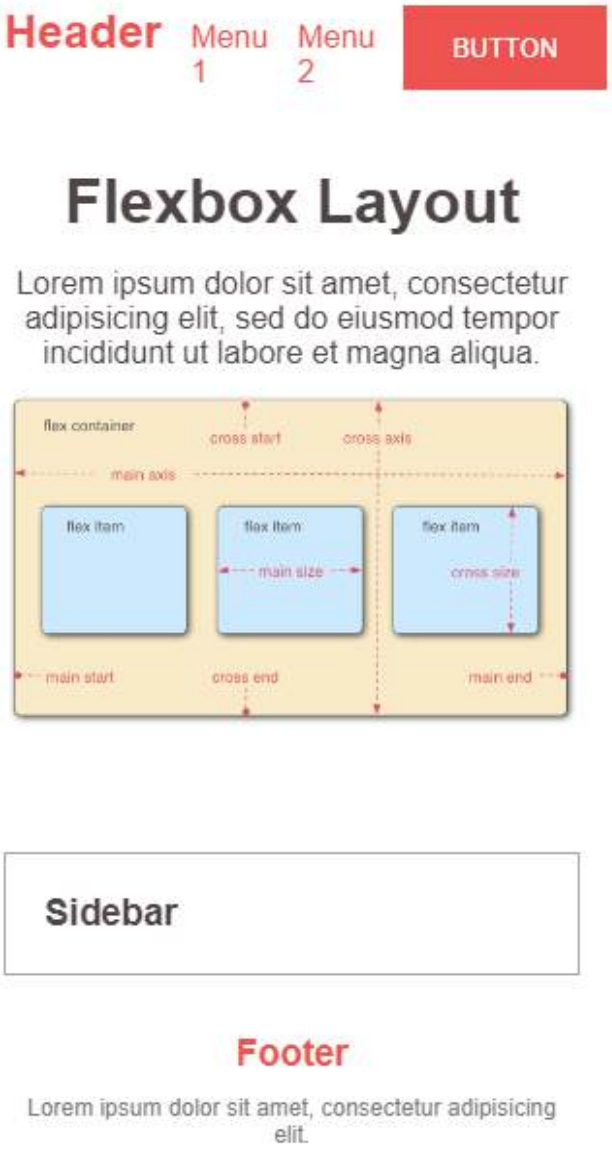


Рисунок 8.6 – Вигляд сайта на невеликому екрані

БІБЛІОГРАФІЧНИЙ СПИСОК

1 A Complete Guide to Grid. Chris House [Electronic resource]. – Access mode : https://css-tricks.com/snippets/css/complete-guide-grid/

2 Best HTML And CSS Cheat Sheets [Electronic resource]. – Access mode : https://cssauthor.com/html-and-css-cheat-sheets/

3 Bootstrap Documentation [Electronic resource]. – Access mode : https://getbootstrap.com/docs/4.1/getting-started/introduction/

4 Bootstrap. Документация на русском языке [Электронный ресурс]. – Режим доступа : http://bootstrap-4.ru/

5 Coyier, Ch. A Complete Guide to Flexbox [Electronic resource]. – Access mode : https://css-tricks.com/snippets/css/a-guide-to-flexbox/

6 CSS Flexbox [Electronic resource]. – Access mode : https://www.w3schools.com/css/css3_flexbox.asp

7 CSS Flexible Box Layout [Electronic resource]. – Access mode : https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

8 CSS Flexible Box Layout Module Level 1 W3C Candidate Recommendation [Electronic resource]. – Access mode : https://www.w3.org/TR/css-flexbox-1/

9 CSS Grid Layout Module Level 1 W3C Candidate Recommendation [Electronic resource]. – Access mode : https://www.w3.org/TR/css-grid-1/

10 Flexbox [Электронный ресурс]. – Режим доступа : https://developer.mozilla.org/ru/docs/Learn/CSS/CSS_layout/Flexbox

11 Font Size Idea: px at the Root, rem for Components, em for Text Elements [Electronic resource]. – Access mode : https://css-tricks.com/rems-ems/

12 HTML Tutorials. And Stuff [Electronic resource]. – Access mode : https://htmldog.com

13 MDN web docs. Основные понятия Grid Layout [Электронный ресурс] – Режим доступа : https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout

14 The World Wide Web Consortium (W3C). Official site [Electronic resource]. – Access mode : http://www.w3.org/

15 Блог веб-разработчика. CSS: создание закругленных углов на сайтах [Электронный ресурс]. – Режим доступа :

http://tigor.com.ua/blog/2008/07/25/rounded-corners-techniques-with-css/

16 Веб-разработка [Электронный ресурс]. – Режим доступа : http://www.websovet.com/category/web-development

17 Дронов, В. А. PHP, MySQL, HTML5 и CSS 3. Разработка современных динамических Web-сайтов / В. А. Дронов. – СПб. : БХВ-Петербург, 2016. – 688 с.

18 Дунаев, В. В. Основы WEB дизайна : самоучитель / В. В. Дунаев. – 2-е изд. – СПб. : БХВ-Петербург, 2012. – 479 с.

19 Журнал для веб-мастеров и блогеров от школы создания сайтов. Web-разработка [Электронный ресурс]. – Режим доступа : https://w3school.ru/category/blog/web-razrabotka

20 Клименко, Р. А. Веб-мастеринг: изучаем HTML5, CSS3, JavaScript, PHP, CMS, AJAX, SEO / Р. А. Клименко. – СПб. : Питер, 2014. – 508 с.

21 Колисниченко, Д. Н. PHP и MySQL. Разработка веб-приложений / Д. Н. Колисниченко. – 5-е изд. – СПб. : БХВ-Петербург, 2015. – 592 с.

22 Мак-Дональд, М. Создание Web-сайта : пер. с англ. / М. Мак-Дональд. – 3-е изд. – СПб. : БХВ-Петербург, 2013. – 612 с.

23 Многоколоночный текст [Электронный ресурс] – Режим доступа : http://htmlbook.ru/css3-na-primerah/mnogokolonochnyy-tekst

24 Модуль CSS3 columns [Электронный ресурс] – Режим доступа : https://html5book.ru/css3-columns/

25 Модуль макета гибкого контейнера [Электронный ресурс]. – Режим доступа : https://html5book.ru/css3-flexbox/

26 Обзор CSS Grid — технологии для упрощения разметки HTML-страниц [Электронный ресурс] – Режим доступа : https://dou.ua/lenta/articles/css-grid-guide/

27 Основы и секреты front-end разработки [Электронный ресурс]. – Режим доступа : http://www.xiper.net/learn/

28 Полное руководство по Flexbox [Электронный ресурс]. – Режим доступа : https://frontender.info/a-guide-to-flexbox/

29 Руководства по веб-технологиям [Электронный ресурс]. – Режим доступа : http://stepbystep.htmlbook.ru/

30 Руководство по HTML5 и CSS3 [Электронный ресурс]. – Режим доступа : https://metanit.com/web/html5/

31 Свойство flex-direction [Электронный ресурс]. – Режим доступа : http://code.mu/css/flex-direction.html

32 Современные уроки CSS для начинающих [Электронный ресурс]. –

Режим доступа : http://ab-w.net/CSS/CSS.php

33 Современный учебник CSS. Учимся создавать веб-страницы, отвечающие современным требованиям [Электронный ресурс]. – Режим доступа : https://idg.net.ua/blog/uchebnik-css

34 Спецификация grid [Электронный ресурс] – Режим доступа : https://ru.coursera.org/lecture/tonkosti-verstki/spietsifikatsiia-grid-k0rBk

35 Уроки з Bootstrap v3 [Електронний ресурс]. – Режим доступу : https://tokar.ua/read/6901

36 Фримен, Эл. Изучаем HTML, XHTML и CSS: пер. с англ. / Эл. Фримен, Эр. Фримен. – СПб. : Питер, 2016. – 720 с.

ЗМІСТ

Навчальне видання

Данова Марія Олександрівна
Сергієнко Володимир Володимирович

WEB-ДИЗАЙН

Частина 1

Редактор О. Ф. Серьожкіна

Зв. план, 2019
Підписано до видання 11.09.2019
Ум. друк. арк. 4,8. Обл.-вид. арк. 5,44. Електронний ресурс

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001