РОЗРОБКА ВЕБ САЙТІВ ДЛЯ ПОЧАТКІВЦІВ

HTML, CSS, JAVASCRIPT

MOZILLA

РОЗРОБКА ВЕБ САЙТІВ ДЛЯ ПОЧАТКІВЦІВ

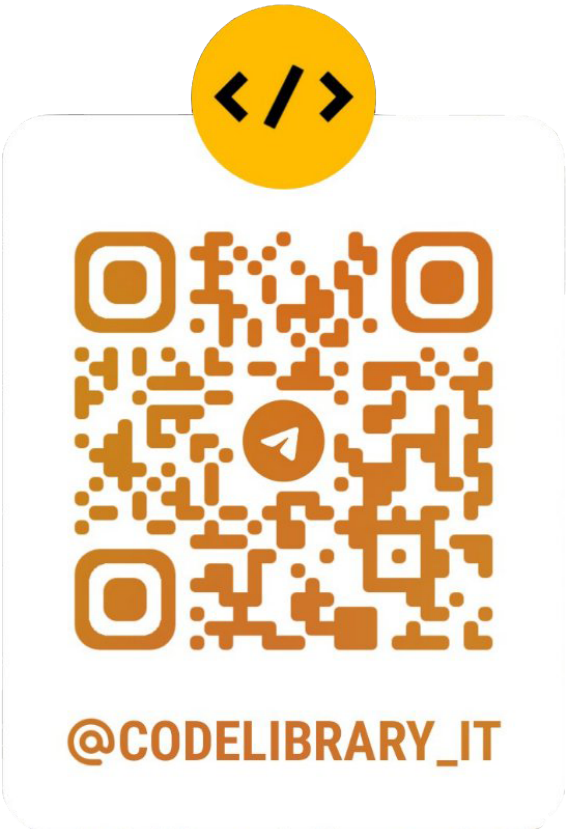
HTML - CSS - JAVASCRIPT

Версія 2022.12.11

Посібник із HTML, CSS та JavaScript. Ці мови програмування повинні вивчити всі веб-розробники. HTML - це мова розмітки вмісту веб-сторінок. CSS визначає макет веб-сторінок. За допомогою JavaScript програмують поведінку веб-сторінок. Я не автор тексту, лише використав інформацію з різних безкоштовних сайтів, переклав з англійської та оформив красиво. Цей методичний посібник добре допомагає школярам і студентам, яким я викладаю програмування.

Джерела інформації: developer.mozilla.org, w3schools.com та інші.

Сергій Курінний
ТЕЛ: +38(096) 311-54-55.
ПОШТА: general.loki.2018@gmail.com
САЙТ: https://kurin.home.blog/



ВСТУП

ЩО ПОТРІБНО ЗНАТИ ПРОГРАМІСТУ

Три мови, які повинні вивчити всі веб-розробники:

1. HTML для визначення вмісту веб-сторінок.
2. CSS для визначення макету веб-сторінок.
3. JavaScript для програмування поведінки веб-сторінок.

Якщо потрібно програмувати серверну частину, необхідно знати:

1. SQL для роботи з базами даних в діалектах MySQL, MS SQL Server, SQLite.
2. PHP для написання серверної частини коду.
3. C# для створення серверної частини коду на веб-сервері Microsoft.

ІНСТРУМЕНТИ

КОМП'ЮТЕР. Можливо, це очевидно для багатьох людей, але деякі з вас читають цю книгу з телефону або планшета. Для веб-розробки краще придбати комп'ютер. Причому екран ноутбука має бути не менше 15 дюймів. Оперативна пам'ять - щонайменше 8 гігабайт. У процесорі - щонайменше два ядра.

ТЕКСТОВИЙ РЕДАКТОР:

Notepad++ (http://notepad-plus-plus.org/).

Visual Studio Code (https://code.visualstudio.com/).

ВЕБ-БРАУЗЕР:

Firefox (https://www.mozilla.org/ru/firefox/new/).

Chrome (https://www.google.com/chrome/browser/).

Opera (https://www.opera.com/).

ГРАФІЧНИЙ РЕДАКТОР:

Paint.NET (http://www.getpaint.net/).

Inkscape (https://inkscape.org).

УТИЛІТИ:

PortableApps (https://portableapps.com/)

КОРИСНІ САЙТИ:

w3schools HTML (https://www.w3schools.com/html/)

w3schools CSS (https://www.w3schools.com/css/)

w3schools Javascript (https://www.w3schools.com/js/)

w3schools SQL (https://www.w3schools.com/sql/)

ПЛАНУВАННЯ

Обдумайте план і дизайн веб-сайту, перш ніж приступити до написання коду, у тому числі: Яку інформацію буде містити веб-сайт? Що робитиме сайт?

Перед тим, як робити щось, вам потрібні ідеї. Що веб-сайт має фактично робити? Веб-сайт може робити все що завгодно, але для першої спроби ви повинні дотримуватися простих речей. Ми почнемо зі створення простої веб-сторінки, яка містить заголовок, зображення та кілька абзаців.

Для початку потрібно відповісти на такі питання: Про що веб-сайт? Яку інформацію ви надаєте про цей предмет? Напишіть заголовок і кілька абзаців і подумайте над зображеннями, які ви хочете показати на сторінці.

Як виглядатиме веб-сайт у простих термінах високого рівня? Який колір тла? Який вид шрифту буде доречним: діловий, мультяшний, жирний чи тонкий?

Комплексні проекти потребують детальних посібників, які включають всі деталі кольорів, шрифтів, відстані між елементами на сторінці, відповідний стиль сторінок тощо. Їх іноді називають посібниками з проектування або бренд-бук.

МАЛЮНКИ ДИЗАЙНУ:

Візьміть ручку та папір і зробіть зразок малюнка того, як ви хочете, щоб виглядав сайт. Для веб-сторінки має вийти невеликий малюнок, і ви повинні взяти це за звичку.

Навіть у реальних, складних веб-сайтах команда розробників зазвичай починає з ескізу на папері і потім будує цифрові макети, використовуючи графічні редактори або веб-технології.

АКТИВИ

На цьому етапі непогано почати збирати воєдино весь контент, який розташовуватиметься на веб-сторінці. У вас має бути текст, розбитий на заголовки та параграфи. Дотримуйтесь цього правила.

КОЛЬОРОВА СХЕМА:

Перейдіть до інструмента вибору кольору та виберіть колір, який вам подобається. [https://tinyurl.com/vdalnvl]

Коли ви натиснете на колір, то побачите дивний код з шести цифр, наприклад, #660066. Це шістнадцятковий код і він представляє колір. Скопіюйте цей код кудись прямо зараз.

ЗОБРАЖЕННЯ:

Щоб вибрати зображення, перейдіть до Google Картинки і знайдіть щось потрібне. Коли ви знайдете потрібне зображення, клацніть на ньому. Натисніть кнопку "В повному розмірі" (View image). На наступній сторінці, клацнувши праворуч на зображенні, виберіть "Зберегти зображення як..." (Save Image As...) і вкажіть місце для зберігання зображення.

Більшість зображень в Інтернеті, використаних у Google Картинках, мають авторські права. Щоб зменшити ймовірність порушення авторських прав, використовуйте фільтр ліцензії.

ВИБРАТИ ШРИФТ:

Перейдіть на Google Fonts (http://www.google.com/fonts) і прокрутіть список вниз, доки не знайдете шрифт, який вам сподобається. Ви також можете використовувати елементи керування зліва для подальшої фільтрації результатів. Клацніть на кнопці плюс поряд зі шрифтом, який ви хочете вибрати. Далі клацніть на панелі у нижній частині сторінки, яка містить список вибраних шрифтів. Коли панель відкриється, скопіюйте код для вставки на сторінку. Він буде схожий на:

```
<link href="https://fonts.googleapis.com/css?family=Gelasio&display=swap"
rel="stylesheet">
```

ФАЙЛИ І ПАПКИ

ДЕ РОЗМІСТИТИ НА КОМП'ЮТЕРІ:

Веб-сайт складається з безлічі файлів: текстового контенту, коду, стилів, медіа-контенту тощо. Коли ви створюєте веб-сайт, ви повинні зібрати ці файли до раціональної структури на локальному комп'ютері.

Коли ви працюєте на веб-сайті локально на комп'ютері, то повинні тримати всі пов'язані файли в одній папці, яка відображає файлову структуру опублікованого веб-сайту на сервері. Ця папка може розташовуватися будь-де, але ви повинні покласти її туди, де зможете легко її знайти, можливо, на робочому столі, в домашній папці або в корені жорсткого диска.

Виберіть місце для зберігання проектів веб-сайту. Тут створіть нову папку з ім'ям web-projects (або аналогічним). Це місце, де зберігатимуться всі проекти. Всередині цієї першої папки створіть іншу папку для зберігання першого веб-сайту. Назвіть її test-site (чи якось інакше).

РЕГІСТР І ПРОБІЛИ:

Ми просимо завжди називати папки та файли повністю в нижньому регістрі без пробілів. Багато комп'ютерів і веб-серверів є чутливими до регістру. Наприклад, якщо ви поклали зображення на веб-сайт у test-site/MyImage.jpg, а потім в іншому файлі ви намагаєтеся викликати зображення як test-site/myimage.jpg, він може не спрацювати.

Браузери, веб-сервери та мови програмування не обробляють послідовно пробіли. Якщо ви використовуєте пробіли в імені файлу, деякі системи можуть подумати, що це імена двох різних файлів. Деякі сервери замінюють пробіли в імені файлу на %20 (код для пробілів в URI), порушуючи всі посилання. Краще розділяти слова за допомогою тире та нижнього підкреслення: my-file.html або my_file.html. Найкраще завжди писати назви папок і файлів у нижньому регістрі та без пробілів. Так ви зіткнетеся з меншою кількістю проблем.

СТРУКТУРА САЙТУ:

У будь-якому сайті є: індексний файл HTML, папки з картинками, файли стилів і файли скриптів. Створіть їх!

index.html: Цей файл зазвичай містить контент домашньої сторінки, тобто текст та зображення, які люди бачать, коли вони вперше потрапляють на сайт. Використовуючи текстовий редактор, створіть новий файл з ім'ям index.html та збережіть його прямо всередині папки test-site.

Папка images: Ця папка, як правило, містить зображення, які ви використовуєте на сайті. Створіть папку з іменем images всередині папки test-site.

Папка styles: Ця папка містить CSS код, який використовується для стилізації контенту (наприклад, налаштування тексту та кольору фону). Створити папку styles всередині test-site.

Папка scripts: Ця папка містить весь JavaScript-код, який використовується для додавання інтерактивних функцій на сайті. Створіть папку з іменем scripts усередині test-site.

ВІДОБРАЖЕННЯ РОЗШИРЕНЬ ФАЙЛІВ НА WINDOWS:

На комп'ютерах під керуванням Windows можуть виникнути проблеми з відображенням імен файлів, оскільки Windows має налаштування під назвою “Приховувати розширення для відомих типів файлів”. Воно включено за замовчуванням. Зазвичай ви можете вимкнути його, перейшовши у провідник, вибрати варіант "Властивості папки..." і зняти прапорець “Приховувати розширення для відомих типів файлів”, потім клацнути OK. Щоб отримати докладнішу інформацію, виконайте пошук в Інтернеті.

ШЛЯХ ДО ФАЙЛА:

Щоб файли спілкувалися один з одним, ви повинні вказати файлам шлях між ними - зазвичай один файл знає, де знаходиться інший. Щоб продемонструвати це, ми вставимо трохи HTML у файл index.html та навчимо його відображати зображення.

Скопіюйте зображення, яке ви вибрали, у папку images.

Відкрийте файл index.html та вставте такий код у файл саме у цьому вигляді. Не турбуйтеся про те, що все це означає – пізніше ми розглянемо код докладніше.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Це заголовок</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>
  <body>
    <img src="" alt="Моє зображення">
  </body>
</html>
```

Рядок <img src="" alt="Моє зображення"> - HTML код, який вставляє зображення на сторінку. Ми повинні сказати HTML, де знаходиться зображення. Воно знаходиться всередині папки images, яка розташована в тій же папці, що й index.html. Пройшовши вниз файловою структурою від index.html до зображення, отримаємо шлях до файлу, який нам потрібен. Він виглядає як images/your-image-filename. Наприклад, зображення, назване firefox-icon.png, має такий шлях до файлу: images/firefox-icon.png.

Вставте шлях до файлу в HTML-код між подвійними лапками src="". Збережіть HTML файл і завантажте його в браузері (подвійне клацання по файлу). Ви маєте побачити веб-сторінку, яка відображає зображення.

КІЛЬКА ПРАВИЛ ПРО ШЛЯХИ ДО ФАЙЛІВ:

Для посилання на файл в тій же папці, що і HTML файл, використовуйте ім'я файла без нічого, наприклад, my-image.jpg. Для посилання на файл у підпапці, напишіть ім'я папки на початку шляху, плюс косу межу. Наприклад: subdirectory/my-image.jpg. Для посилання на файл у папці вище за HTML файл, напишіть дві точки. Наприклад, якщо index.html знаходиться всередині підпапки test-site, а my-image.png - усередині test-site, ви можете звернутися до my-image.png із index.html, використовуючи ../my-image.png.

Ви можете комбінувати їх так, як вам подобається, наприклад, ../subdirectory/another-subdirectory/my-image.png.

Файлова система Windows прагне використовувати зворотний сліш, а не косу межу, наприклад C:\windows. Це не має значення, навіть якщо ви розробляєте веб-сайт на Windows, ви все одно повинні використовувати звичайні сліші в коді.

СТРУКТУРА ПАПКИ МАЄ ВИГЛЯДАТИ ТАК:

HTML

ЩО ТАКЕ HTML?

HTML (Hypertext Markup Language) - код, який використовується для структурування та відображення веб-сторінки та її контенту (вмісту). Контент може бути структурований всередині множини параграфів, маркованих списків або з використанням зображень і таблиць даних.

HTML не є мовою програмування. Це мова розмітки, яка використовується, щоб повідомляти браузеру, як відображати веб-сторінки, які ви відвідуєте. Він може бути складним або простим, залежно від того, як веб-дизайнер хоче.

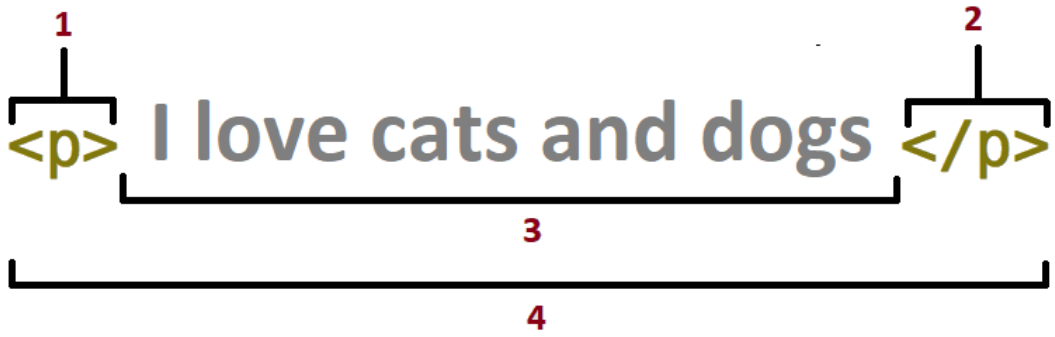
HTML складається з елементів, які використовують, щоб вкладати або обертати різні частини контенту, щоб змусити контент відображатись або діяти певним чином. HTML теги можуть, наприклад, зробити слово або зображення посиланням на щось ще, можуть зробити слова курсивом, зробити шрифт більшим або меншим і так далі.

Візьмемо рядок: Мій кіт нудьгує. Якщо ми хочемо рядок, що стоїть сам по собі, треба вказати, що це параграф, укладаючи рядок в тег параграфа <p>:

```
<p>Мій кіт нудьгує</p>
```

АНАТОМІЯ ЕЛЕМЕНТУ

Елемент складається з кількох частин.



1. ВІДКРИВАЮЧИЙ ТЕГ: складається з імені елементу (в даному випадку "p"), укладеного в кутові дужки, що відкривають і закривають. Вказує, де елемент починається чи починає діяти, у даному випадку - де починається параграф.

2. ЗАКРИВАЮЧИЙ ТЕГ: те ж саме, що й відкриваючий тег, за винятком того, що він включає в себе косу межу перед ім'ям елементу. Вказує, де елемент закінчується, у даному випадку - де закінчується параграф. Відсутність тега, що закриває, є однією з поширених помилок і може призводити до дивних результатів.

3. КОНТЕНТ: контент елементу, який цього разу є просто текстом.

4. ЕЛЕМЕНТ: контент і теги, що відкривають і закривають контент, разом складають елемент.

Елементи можуть мати атрибути, які мають такий вигляд:

Атрибути містять додаткову інформацію про елемент, який ви не хочете показувати в контенті. У даному випадку, class це ім'я атрибута, а editor-note – це значення атрибута. Клас дозволяє дати елементу ідентифікаційне ім'я, яке може пізніше використовуватися, щоб звертатися до елементу з інформацією про стиль та інші речі.

Атрибут завжди повинен мати:

- Простір між ним та ім'ям елементу (або попереднім атрибутом, якщо елемент вже має один або кілька атрибутів).

- Назва атрибута, а потім знак рівності.

- Значення атрибута, укладене з двох сторін у лапки.

ВЛОЖЕНІ ЕЛЕМЕНТИ

Ви можете розташовувати елементи всередині інших елементів - це називається вкладенням. Якщо ми хочемо заявити, що кіт ДУЖЕ сумує, ми можемо укласти слово "дуже" в елемент , який вказує, що слово має бути акцентоване:

```
<p>Мой кіт <strong>дуже</strong> сумує.</p>
```

Ви також повинні переконатися, що елементи вкладені правильно: у прикладі вище ми відкрили перший елемент <p>, потім елемент, потім ми повинні закрити спочатку елемент, потім <p>. Це помилка:

```
<p>Мой кіт <strong>дуже сумує.</p></strong>
```

Елементи повинні відкриватися та закриватися правильно. Розташуйте їх явно всередині або зовні один одного. Якщо вони перекриваються, веб-браузер намагатиметься зробити найкраще припущення на основі того, що ви намагалися сказати, і ви отримаєте неправильні результати.

ПОРОЖНІ ЕЛЕМЕНТИ

Деякі елементи не мають контенту і називаються порожніми елементами. Візьмемо елемент <img>, який вже є в HTML:

```
<img src="images/firefox-icon.png" alt="Моє тестове зображення">
```

Він містить два атрибути, але не має тега, що закриває </img>, і внутрішнього контенту. Це тому, що елемент зображення не обертає контент для впливу на нього. Його метою є вставка зображення в HTML сторінку в потрібному місці.

КОМЕНТАРІ

Коментарі не відображаються у браузері, але вони можуть допомогти документувати код. Щоб додати коментар, використовуйте такий синтаксис:

```
<!-- Пишіть свої коментарі тут -->
```

Зверніть увагу, що в початковому тегу є знак оклику (!), а потім два мінуса. За допомогою коментарів ви можете розміщувати повідомлення та нагадування в HTML-коді:

```
<!-- Це коментар -->
<p>Це параграф.</p>
<!-- Необхідно додати більше інформації тут -->
```

Коментарі можна використовувати для приховання контенту. Це може бути корисно, якщо ви тимчасово приховуєте контент:

```
<p>Це параграф.</p>
<!-- <p>Це інший параграф.</p> -->
<p>Це теж параграф.</p>
```

Ви також можете приховати більше одного рядка. Все, що знаходиться між знаками **<!--** і **-->**, буде приховано від користувача. Приховати частину коду:

```
<p>Це параграф.</p>
<!--
<p>Погляньте на цю картинку:</p>
<img border="0" src="pic_trulli.jpg" alt="Trulli">
-->
<p>Це теж параграф.</p>
```

Коментарі також відмінно підходять для налагодження HTML, оскільки ви можете коментувати рядки коду HTML по одному за раз для пошуку помилок. Коментарі можна використовувати для приховування частин у середині HTML-коду. Приховати частину абзацу:

```
<p>This <!-- great text --> is a paragraph.</p>
```

АНАТОМІЯ ДОКУМЕНТА

Повернімося до коду, який ми записували в index.html.

<!DOCTYPE html> - доктайп. У минулому доктайпи мали виступати як посилання на набір правил, яким HTML сторінка мала дотримуватися.

Нині ніхто не дбає про це, і він насправді просто історичний артефакт, який має бути включений для того, щоб все працювало правильно.

<html></html> - кореневий елемент, він обертає весь контент на сторінці.

<head></head> - елемент виступає як контейнер для всього, що ви бажаєте включити в HTML сторінку, але не є контентом, який ви показуєте користувачам сторінки. До них відносяться такі речі, як ключові слова та опис сторінки, які будуть з'являтися в результатах пошуку, CSS стилі контенту, кодування та інше.

<body></body> - елемент містить весь контент, який ви хочете показувати користувачам, коли вони відвідують сторінку, будь то текст, зображення, відео, ігри, аудіодоріжки, що програються, або щось ще.

<meta charset="utf-8"> - елемент встановлює utf-8 кодування документа, яке включає більшість символів з усіх відомих мов. Тепер документ може обробляти будь-який текстовий контент, який ви вкладете в нього. Немає причин не встановлювати її, тому що це допоможе уникнути проблем.

<title></title> - встановлює заголовок для сторінки, який є назвою, що з'являється на вкладці браузера сторінки, що завантажується, і використовується для опису сторінки, коли ви додаєте її в обране.

<meta name="viewport" content="width=device-width, initial-scale=1" /> - спеціальний код, який потрібний для правильного відображення сторінки на мобільних пристроях.

ЗОБРАЖЕННЯ

Звернемо увагу на елемент зображення:

```
<img src="images/firefox-icon.png" alt="Моє зображення">
```

Цей код вбудовує зображення на сторінку в потрібному місці. Це робиться за допомогою атрибута src, в якому міститься шлях до файлу зображення.

Ми також включили атрибут alt. У цьому атрибуті ви вказуєте пояснювальний текст для користувачів, які не можуть побачити зображення з таких причин:

1. У них є порушення зору. Користувачі з порушенням зору часто використовують "екранні диктори", які читають їм альтернативний текст.

2. Щось пішло не так, внаслідок чого зображення не відобразилося. Наприклад, спробуйте навмисно змінити шлях в src, зробивши його неправильним. Якщо ви збережете та перезавантажите сторінку, то маєте побачити альтернативний текст замість зображення.

Альтернативний текст – "пояснювальний текст". Він повинен надати достатньо інформації, щоб мати уявлення про те, що передає зображення. У цьому прикладі текст "Моє зображення" не годиться. Найкращою альтернативою для логотипу Firefox буде "Логотип Firefox: вогняний Лис навколо Землі".

Спробуйте вигадати більш відповідний альтернативний текст для зображення.

ЗАГОЛОВКИ

Елементи заголовка дозволяють вказувати певні частини контенту як заголовки або підзаголовки контенту. Так само, як книга має назву, назви розділів і підзаголовків, HTML документ може містити те саме. HTML включає шість рівнів заголовків <h1>–<h6>, хоча зазвичай ви будете використовувати не більше трьох-чотирьох:

```
<h1>Головний заголовок</h1>
<h2>Заголовок верхнього рівня</h2>
<h3>Підзаголовок</h3 >
<h4>Підзаголовок Підзаголовка</h4>
```

Тепер спробуйте додати відповідну назву для HTML сторінки, трохи вище за елемент <img>.

ПАРАГРАФИ

В елементах <p> містяться параграфи тексту. Ви будете використовувати їх регулярно при розмітці тексту:

```
<p>Це параграф</p>
```

Додайте текст до кількох параграфів, розташованих прямо під елементом <img>.

СПИСКИ

Більшість веб-контенту є списками, і HTML має спеціальні елементи для них. Розмітка списку завжди складається щонайменше з двох елементів. Найбільш поширеними типами списків є нумеровані та ненумеровані списки. Ненумеровані списки - це списки, де порядок пунктів не має значення, наприклад, список покупок. Вони обертаються елементом <ul>. Нумеровані списки – списки, де порядок пунктів має значення. Вони обертаються елементом <ol>.

Кожен пункт всередині списків знаходиться всередині елемента <li>. Наприклад, якщо ми хочемо включити частину цьогоого фрагмента параграфа до списку:

```
<p>Mozilla, ми є світовою спільнотою технологів, мислителів та будівельників, що працюють разом</p>
```

Ми могли б змінити розмітку на таку:

```
<p>Mozilla, ми є світовою спільнотою</p>
<ul>
  <li>технологів</li>
  <li>мислителів</li>
  <li>будівельників</li>
</ul>
<p>, що працюють разом</p>
```

Спробуйте додати впорядкований або невпорядкований список на сторінку.

ПОСИЛАННЯ

Для додавання посилання потрібно використовувати простий елемент - <a>. Для того щоб текст у параграфі став посиланням, виберіть текст, наприклад Google. Оберніть текст на елемент <a>:

```
<a>Google</a>
```

Задайте елементу <a> атрибут href:

```
<a href= "">Google</a>
```

Заповніть значення цього атрибуту веб-адресою, на яку ви хочете вказати посилання. Наприклад:

```
<a href= "https://www.google.com">Google</a>
```

Ви можете отримати несподівані результати, якщо на початку веб-адреси ви опустите https:// або http:// частину, яку називають протоколом. Після створення посилання, натисніть на ньому, щоб переконатися, що вона працює.

CSS

ЩО ТАКЕ CSS?

CSS (Cascading Style Sheets) – код, який ви використовуєте для стилізування веб-сторінки. Ми відповімо на такі запитання: Як я можу зробити текст чорним чи червоним? Як я можу зробити так, щоб контент з'являвся у різних місцях екрана? Як прикрасити веб-сторінку фоновими зображеннями та кольорами?

CSS не є мовою програмування. Це мова таблиці стилів, вона дозволяє вибірково застосовувати стилі до елементів в HTML документах. Наприклад, щоб вибрати всі елементи параграфа на HTML сторінці і перетворити їх текст на червоний, ви повинні написати наступне:

```
p {
  color: red;
}
```

Давайте спробуємо: вставити ці три рядки CSS в новий файл у текстовому редакторі, а потім збережемо файл як style.css у папці styles. Нам потрібно застосувати CSS до HTML документа, інакше CSS стиль не вплине на те, як браузер відображає HTML документ. Відкрийте файл index.html і вставте такий рядок у шапку між тегами <head> і </head>:

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

Збережіть index.html і завантажте його у браузері. Якщо текст параграфа тепер червоний, вітаємо. Ви написали перший CSS! CSS можна визначити прямо всередині HTML файлу. Для цього використовуйте тег <style></style>. Всередині цього тегу пишіть так, ніби це CSS файл:

```
<style>
p {
  color: red;
}
</style>
```

Ви також можете визначати стиль прямо всередині будь-якого елемента. Наприклад:

```
<table style="width: 100%; margin-bottom: 0.5em;">
```

АНАТОМІЯ НАБОРУ ПРАВИЛ

Давайте розглянемо CSS докладніше. Уся структура називається набором правил. Зазначимо імена окремих елементів:

1. СЕЛЕКТОР. Ім'я HTML елементу на початку набору правил. Він вибирає елементи для застосування стилю (в даному випадку елементи p). Для стилізування іншого елемента просто змініть селектор.

2. ВИЗНАЧЕННЯ. Одне правило, наприклад, color: red; вкаже, які властивості елементу ви хочете стилізувати.

3. ВЛАСТИВОСТІ. Способи, якими можна стилізувати даний HTML елемент (у разі color є властивістю для елементів p). У CSS ви можете вибрати, які властивості ви хочете вказати у визначенні.

4. ЗНАЧЕННЯ ВЛАСТИВОСТІ. Праворуч від властивості, після двокрапки знаходиться значення властивості, в якому вибирається одне з безлічі можливих значень для даної властивості (у color є безліч значень, крім red).

ВАЖЛИВІ ЧАСТИНИ СИНТАКСИСУ:

Кожен набір правил (крім селектора) повинен бути обернений у фігурні дужки ({}). У кожному оголошенні необхідно використовувати двокрапку (:), щоб відокремити властивість від її значень. У кожному наборі правил ви повинні використовувати точку з комою (;), щоб відокремити кожне оголошення від наступного.

Щоб змінити кілька значень властивостей відразу, вам потрібно написати їх, розділяючи крапкою з комою:

```
p {
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

Ви також можете вибрати безліч елементів різних типів і застосувати єдиний набір правил для них всіх. Додайте кілька селекторів, розділених комами.

```
p,li,h1 {
  color: red;
}
```

Все в CSS документі між /* і */ є коментарем, який браузер ігнорує, коли він обробляє код. Це місце, де ви пишете нотатки про те, що робите.

ТИПИ СЕЛЕКТОРІВ

Існує безліч різних типів селектора. Вище ми розглядали лише селектор елементів, який вибирає всі елементи цього типу у HTML документі. Але ми можемо зробити вибір конкретнішим. Ви можете знайти докладніший список у посібнику селекторів.

СЕЛЕКТОР ЕЛЕМЕНТУ:

Іноді називається селектором тега чи типу. Вибирає всі HTML елементи вказаного у селекторі типу. Всі елементи <p> будуть вирівняні по центру з червоним текстом:

```
p {
  text-align: center;
  color: red;
}
```

ID СЕЛЕКТОР:

Вибирає елемент із зазначеним ID (на сторінці, може бути лише один елемент із будь-яким ID). Це правило буде застосовано до елементу з id="para1":

```
#para1 {
  text-align: center;
  color: red;
}
```

СЕЛЕКТОР КЛАСУ:

Вибирає елементи на сторінці із зазначеним класом (безліч екземплярів класу може бути на сторінці). Всі елементи HTML з class="center" будуть червоними та вирівняні по центру:

```
.center {
  text-align: center;
  color: red;
}
```

Елементи HTML також можуть відноситися до більш ніж одного класу. У цьому прикладі елемент <p> буде оформлений відповідно до стилів class="center" і class="large":

```
<p class="center large">Цей параграф має два класи.</p>
```

УНІВЕРСАЛЬНИЙ СЕЛЕКТОР:

Універсальний селектор (*) вибирає всі елементи HTML на сторінці. Наведене нижче правило CSS вплине на кожен елемент HTML на сторінці:

```
* {
  text-align: center;
  color: blue;
}
```

СЕЛЕКТОР ГРУПУВАННЯ:

Селектор групування вибирає всі елементи HTML з однаковими визначеннями стилю. У наступному коді елементи h1, h2 та p мають однакові визначення стилю:

```
h1 {
  text-align: center;
  color: red;
}
h2 {
  text-align: center;
  color: red;
}
p {
  text-align: center;
  color: red;
}
```

Краще згрупувати селектори, щоб мінімізувати код. Щоб згрупувати селектори, розділіть кожен селектор комою. У цьому прикладі ми згрупували селектори з наведеного вище коду:

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

СЕЛЕКТОР АТРИБУТА:

Використовується для вибору елементів із зазначеним атрибутом. У цьому прикладі вибираються всі елементи <a> з атрибутом “target”:

```
a[target] {
  background-color: yellow;
}
```

СЕЛЕКТОР АТРИБУТА ЗІ ЗНАЧЕННЯМ:

Селектор [attribute="value"] використовується для вибору елементів із зазначеним атрибутом та значенням. У цьому прикладі вибираються всі елементи <a> з атрибутом target="_blank":

```
a[target="_blank"] {
  background-color: yellow;
}
```

СЕЛЕКТОР ПСЕВДОКЛАСУ:

Псевдоклас використовується для визначення особливого стану елемента. Наприклад, його можна використовувати, щоб вказати:

- Стиль елемента, коли користувач наводить курсор мишки.
- Стиль відвіданих та невідвіданих посилань по-різному.
- Стиль елементу, коли він отримує фокус.

Синтаксис псевдокласів:

```
selector:pseudo-class {
  property: value;
}
```

Посилання можуть відображатися по-різному:

```
/* не відвідане посилання */
a:link {
  color: #FF0000;
}
/* відвідане посилання */
a:visited {
  color: #00FF00;
}
/* мишка над посиланням*/
a:hover {
  color: #FF00FF;
}
/* вибране посилання */
a:active {
  color: #0000FF;
}
```

Селектор a:hover ПОВИНЕН йти після a:link та a:visited у визначенні CSS, щоб бути ефективним! Селектор a:active ПОВИНЕН йти після a:hover визначення CSS, щоб бути ефективним! Імена псевдокласів не чутливі до регістру.

Псевдокласи можна поєднувати з класами HTML. Коли ви наводите курсор на посилання з класом “highlight”, воно змінює колір:

```
a.highlight:hover {
  color: #ff0000;
}
```

ШРИФТИ І ТЕКСТ

Давайте зробимо шрифти та текст трохи кращими. Перш за все, поверніться і знайдіть текст з Google Fonts, який ви десь зберегли. Додайте елемент <link ... > усередині шапки index.html (у будь-якому місці між <head> і </head>). Це буде виглядати так (все в одному рядку коду):

```
<link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
```

Потім видаліть існуюче правило у файлі style.css. Це був добрий тест, але червоний текст насправді виглядає не дуже добре. Додайте наступні рядки в потрібне місце, замінивши рядок placeholder фактичною font-family рядком, який ви отримали від Google Fonts (font-family просто означає, який шрифт ви хочете використовувати для тексту).

Це правило спочатку встановлює глобальний базовий шрифт і розмір шрифту для всієї сторінки (оскільки <html> є батьківським елементом для всієї сторінки, і всі елементи всередині нього успадковують такий самий font-size і font-family):

```
html {
  font-size: 30px; /* Шрифт буде розміром 30 пікселів*/
  font-family: [placeholder]
}
```

Все разом:

```
<head>
...
<link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
<style>
 html {
      font-family: 'Roboto', sans-serif;
      font-size: 30px;
 }
</style>
...
</head>
```

Значення sans-serif додатково вказано у font-family на випадок, якщо буде якась помилка при завантаженні шрифта Roboto.

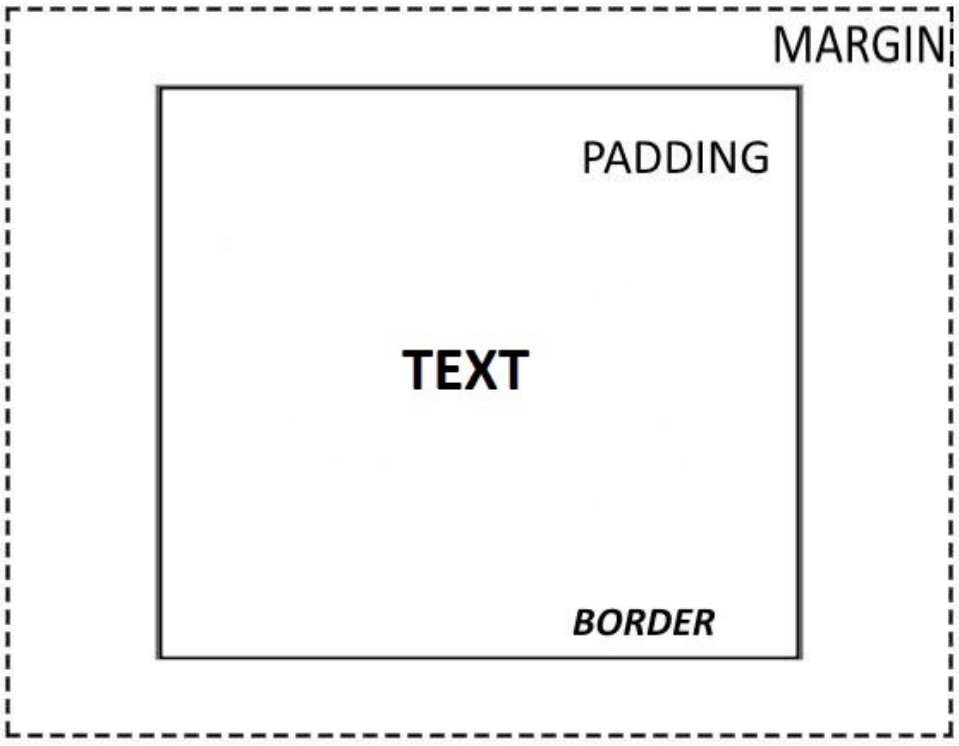
Коли вибиратимете шрифт, майте на увазі, що не в усіх шрифтах є українські символи, оскільки багато шрифтів спроектовані лише для англійського алфавіту.

Тепер ми встановимо розмір шрифту для елементів, що містять текст усередині HTML тіла (<h1>, <li> та <p>). Ми також відцентруємо текст заголовка та встановимо висоту рядка та відстань між літерами у тілі документа, щоб зробити його зручним для читання:

```
h1 {
  font-size: 60px;
  text-align: центр;
}
p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}
```

БЛОКИ

У написанні CSS постійно використовуються блоки - встановлення їх розміру, кольору та розташування. Більшість елементів HTML на сторінці можуть бути представлені як блоки, розташовані один в одному. Кожен із блоків має багато різних властивостей.



padding – простір лише навколо контенту.

border - суцільна лінія, розташована поряд із padding.

margin – простір навколо зовнішньої сторони елементу.

width – ширина елементу.

background-color – колір позаду контенту.

color – колір контенту елементу (зазвичай тексту).

text-shadow – встановлює тінь тексту всередині елементу.

display – встановлює режим відображення елементу.

КОЛІР СТОРІНКИ

Це правило встановлює колір фону для всієї сторінки.

```
html {
  background-color: #00539F;
}
```

Змініть код кольору зверху, на колір, який ви вибрали під час планування сайту.

Вибрати кольори [https://htmlcolorcodes.com/].

Браузер розуміє англійські назви понад 100 кольорів. Наприклад, Green (зелений), Blue (синій), Orange (помаранчевий), Red (червоний), Yellow (жовтий), Purple (фіолетовий), White (білий), Black (чорний), Pink (рожевий), Turquoise (бірюзовий)), Violet (фіолетовий), SkyBlue (небесно-блакитний), PaleGreen (світло-зелений).

Повний список назв кольорів можна знайти на сайті CSS-Tricks:
[https://tinyurl.com/grpk652]

СТИЛІЗАЦІЯ ТІЛА СТОРІНКИ

```
body {
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}
```

width: 600px;
Задає ширину тіла сторінки у 600 пікселів.

margin: 0 auto;
Коли ви встановите два значення для таких властивостей як margin або padding, перше значення елементу впливає на верхню та нижню сторону (зробивши їх 0 у даному випадку), а друге значення для лівої та правої сторони. Значення auto є особливим, воно ділить простір по горизонталі порівну зліва та справа. Ви можете використовувати один, два, три чи чотири значення.

background-color: #FF9500;
Встановлює колір фону елементу. Ми використовували червоно-оранжевий для тіла сторінки, на відміну від темно-синього кольору для елементу.

padding: 0 20px 20px 20px;
Ми маємо чотири значення, встановлені для padding, щоб зробити трохи простору навколо контенту. Цього разу ми не встановлюємо padding на верхній частині тіла, але робимо 20 пікселів зліва, знизу та праворуч. Значення встановлюються у такому порядку: зверху, праворуч, знизу, ліворуч.

border: 5px solid black;
Встановлює суцільну чорну рамку шириною 5 пікселів.

СТИЛІЗАЦІЯ ЗАГОЛОВКА

Ми маємо жахливий розрив у верхній частині. Це відбувається тому, що браузери використовують певний стиль за замовчуванням для <h1> елементу. Щоб позбутися від розриву, ми встановлюємо margin=0.

```
h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}
```

Потім ми встановили для заголовка верхній та нижній padding на 20 пікселів, і зробили текст заголовка того ж кольору, як і колір фону html.

Ми використовували властивість text-shadow, яка застосовує текстову тінь для текстового контенту елементу. Він має такі чотири значення:

- Перше значення пікселів задає горизонтальне усунення тіні від тексту - як далеко вона рухається впоперек: негативне значення має рухати її вліво.

- Друге значення задає вертикальне усунення тіні від тексту - як далеко вона рухається вниз, у цьому прикладі негативне значення має перемістити її вгору.

- Третє значення пікселів задає радіус розмиття тіні - велике значення означатиме більш розмиту тінь.

- Четверте значення задає основний колір тіні.

ЦЕНТРУВАННЯ ЗОБРАЖЕННЯ

На закінчення ми відцентруємо зображення, щоб зробити його краще.

```
img {
  display: block;
  margin: 0 auto;
}
```

Ми можемо використовувати margin: 0 auto хитрість знову, тому що ми це робили раніше для body, але ми повинні зробити ще щось. Елемент body є блоковим, це означає, що він займає багато місця на сторінці і може мати margin та інші значення відступів, що застосовуються до нього.

Зображення, навпаки, є малими елементами, тобто вони цього не можуть. Таким чином, щоб застосовувати margin до зображення, ми повинні дати зображенню блокову поведінку за допомогою display: block;.

ІНСТРУМЕНТИ РОЗРОБНИКА

Кожен браузер оснащений інструментами для веб-розробника. Ці інструменти дозволяють робити різні речі - від вивчення завантажених HTML, CSS і JavaScript до відображення того, яких ресурсів потребує сторінка і як довго вона буде завантажуватися.

ЯК ВІДКРИТИ ІНСТРУМЕНТИ?

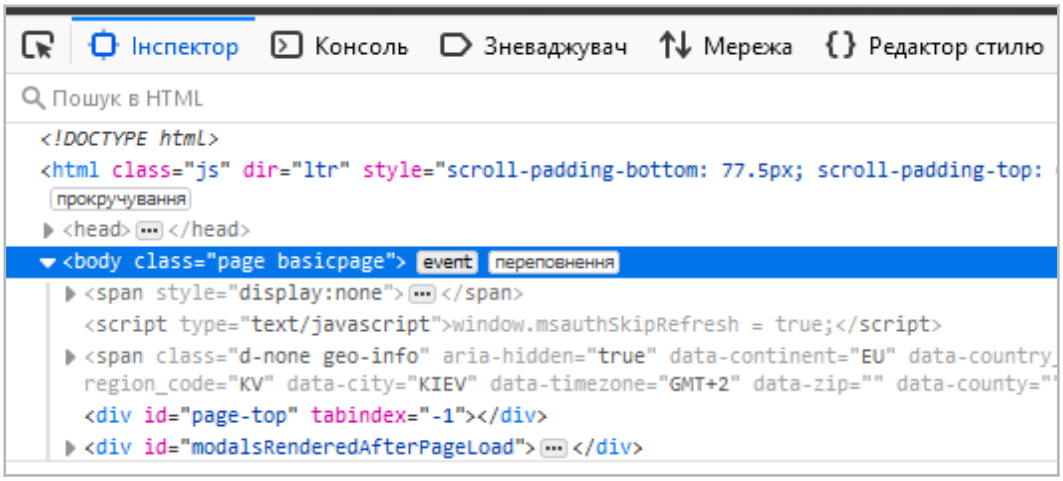
Панель розробника знаходиться у нижній або правій частині браузера. Як її відобразити? Є три варіанти:

- КЛАВІАТУРА: Ctrl+Shift+I.
- ПАНЕЛЬ МЕНЮ. FIREFOX:
  Відкрити меню > Інші інструменти > Інструменти веброзробника.
- ПАНЕЛЬ МЕНЮ. CHROME:
  Відкрити меню > Додаткові інструменти > Інструменти розробника.

КОНТЕКСТНЕ МЕНЮ. Натисніть правою кнопкою миші на будь-якій ділянці веб-сторінки. З'явиться контекстне меню, в якому потрібно вибрати "Дослідити Елемент" (або "Дослідити"). Цей спосіб відобразить код елементу, на якому ви клацнули правою кнопкою.

ІНСПЕКТОР DOM І СТИЛЮ

За замовчуванням на панелі відкривається вкладка Інспектора.



Цей інструмент дозволяє побачити, як HTML-код виглядає на сторінці в даний час, а також CSS, який застосований до кожного елемента на сторінці. Він також дозволяє в реальному часі редагувати HTML та CSS. Зміни можна побачити у вікні браузера. Якщо ви не бачите Inspector, натисніть на вкладку.

Для початку спробуйте натиснути правою кнопкою миші (Ctrl+клік) по елементу HTML у DOM inspector і подивіться на контекстне меню.

Пункти меню можуть відрізнятися в браузерах, але важливими з них є ті самі.

- Видалити вузол (іноді Видалити елемент). Видаляє поточний елемент.

- Правити як HTML (іноді додати атрибут/редагувати текст). Дозволяє редагувати HTML та бачити результат. Корисно для налагодження та тестування.

- :hover/:active/:focus. Примушує елементи переключити свій стан на той, до якого застосовано стиль.

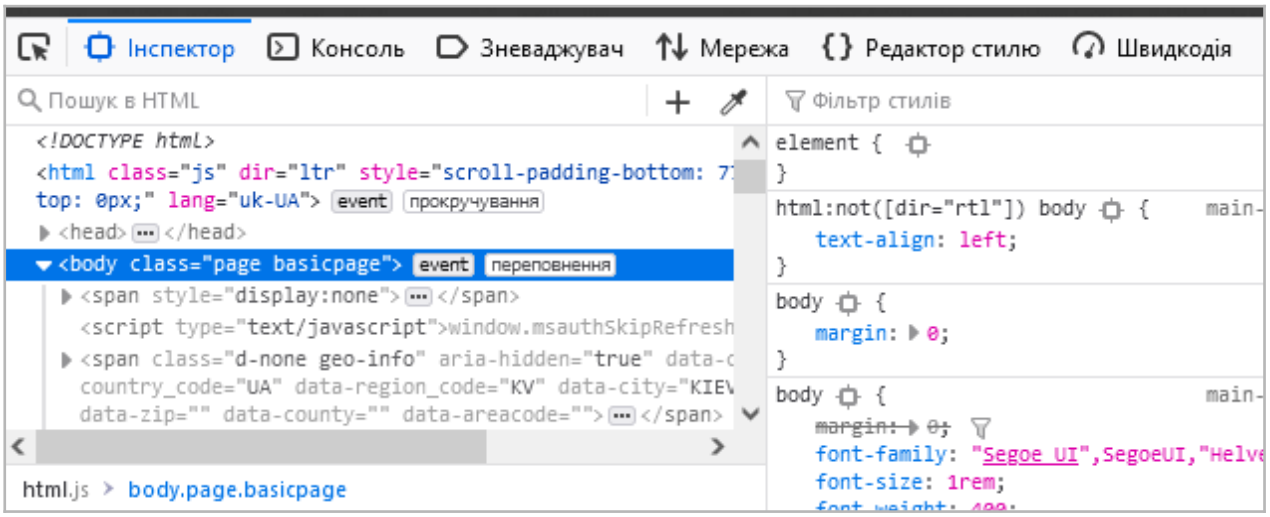
- Копіювати/Копіювати як HTML. Копіює поточний виділений HTML.

Спробуйте змінити щось через вікно Inspector на сторінці прямо зараз. Двічі клацніть на елементі, або натисніть правою кнопкою миші та виберіть "Правити як HTML" з контекстного меню. Ви можете зробити будь-які зміни, які ви захочете, але не зможете їх зберегти.

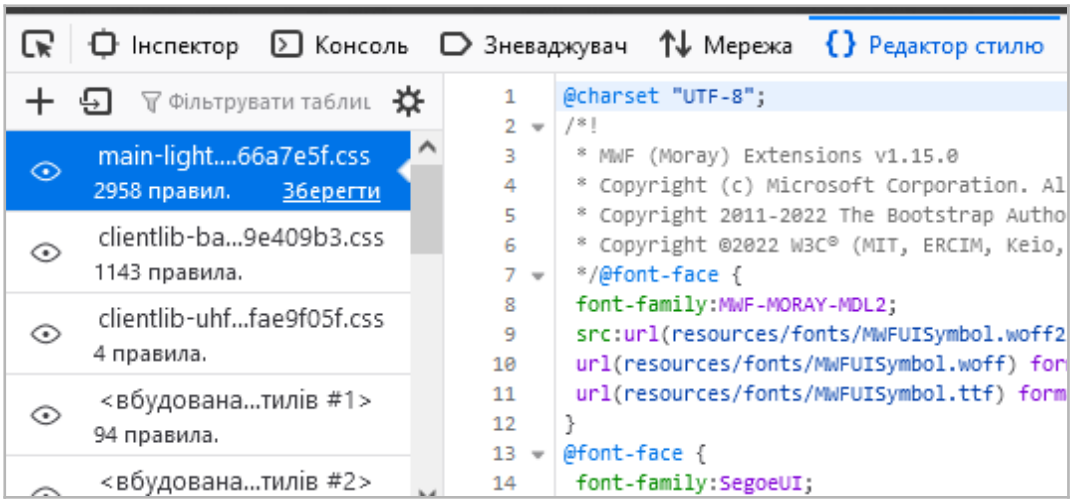
РЕДАКТОР СТИЛЮ

За замовчуванням редактор стилю відображає CSS властивості, застосовані до поточного вибраного елементу. Властивості, застосовані до поточного елементу, відображаються у порядку зменшення пріоритету.

Можна прибирати галочки навпроти властивостей для того, щоб бачити, що вийде, якщо їх видалити. Натисніть на маленьку стрілку поруч із властивістю, щоб побачити всі її еквіваленти. Натисніть на ім'я властивості або значення, щоб відкрити текстове віконце, в якому ви можете задати нові значення і побачити, як зміниться елемент.



Поряд з кожною властивістю вказано ім'я файлу та номер рядка, де розташовується ця властивість. Якщо клацнути, браузер перенесе вас у вікно, де можна редагувати цей CSS і зберегти.



Ви можете також натиснути на фігурну закриваючу дужку будь-якої властивості, щоб вивести текст на новий рядок, де зможете написати абсолютно нову декларацію CSS для сторінки.

Ви могли помітити інші вкладки:

Макет: Відображає блокову модель виділеного елементу. Тут ви можете побачити зовнішні та внутрішні відступи, а також межі застосовані до елементу, тут також вказано їхній розмір.

Розраховано: Тут вказані всі обчислення властивостей виділеного елементу (остаточні значення застосовані браузером).

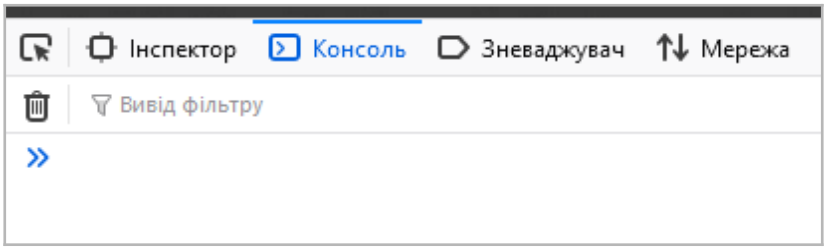
Анімації: На цій вкладці показані анімації застосовані до виділеного елементу.

Сумісність: Показує сумісність властивостей елементу з різними браузерами.

Шрифти: Показує шрифти застосовані для виділеного елементу.

КОНСОЛЬ JAVASCRIPT

Консоль JavaScript - корисний інструмент для налагодження. Вона дозволяє завантажувати JavaScript всупереч порядку завантаження скрипта в браузері і повідомляє про помилки, як тільки браузер намагається виконати код.



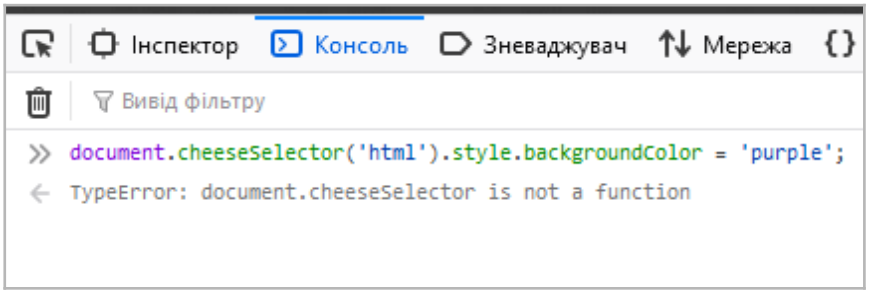
Щоб отримати доступ до консолі, натисніть кнопку Консоль (або Console). Щоб зрозуміти, що відбувається, введіть фрагменти коду в консоль один за одним (а потім натисніть Enter):

```
alert('Привіт');
document.querySelector('html').style.backgroundColor = 'purple';
```

Тепер введіть неправильну версію коду та подивіться, що вийде.

```
document.cheeseSelector('html').style.backgroundColor = 'purple';
```

Ви побачите помилки, які повідомить браузер. Найчастіше ці помилки виглядають загадково, але вони мають бути досить простими, щоб зрозуміти проблему.



JAVASCRIPT

ЩО ТАКЕ JAVASCRIPT?

JavaScript – мова програмування, яка додає інтерактивність на веб-сайт (наприклад: ігри, відгук на натискання кнопки або під час введення даних у форми, динамічні стилі, анімація).

JavaScript ("JS" для стислості) - динамічна мова програмування, яка застосовується до HTML документа. Він може забезпечити інтерактивність на веб-сайтах. Його розробив Brendan Eich, співзасновник проекту Mozilla.

JavaScript – мова програмування, яка дає можливість реалізовувати складну поведінку сайту. Щоразу, коли ви бачите веб-сторінку, яка не тільки відображає статичний вміст, але й робить більше (своєчасно відображає оновлення контенту, виводить інтерактивні карти, 2D/3D анімацію або прокручує відео), будьте певні, що тут не обійшлося без JavaScript.

Вважається, що JavaScript важче вивчити, ніж пов'язані з ним технології, на зразок HTML і CSS. Перед вивченням JavaScript рекомендуємо спочатку ознайомитися з ними.

Необхідно багато працювати, щоб створити професійний веб-сайт. Якщо ви новачок у веб-розробці, ми рекомендуємо почати з малого. Ви не створюватиме великий і складний сайт прямо зараз, проте створити простий веб-сайт нескладно.

Ви можете зробити дуже багато з JavaScript. Ви можете почати з малого, з простих функцій, таких як каруселі, галереї зображень, макети, що змінюються, і відгук на натискання кнопок. Коли ви станете більш досвідченим, ви зможете створювати ігри, анімовану графіку, програми з базами даних та багато іншого!

ДОДАТКИ:

Програмні Інтерфейси Програми (API) вбудовані в браузери, що забезпечують різні функціональні можливості, такі як динамічне створення HTML та встановлення CSS стилів, захоплення та маніпуляція відеопотоком, робота з веб-камерою користувача або генерація 3D графіки та аудіо семплів.

Сторонні API дозволяють розробникам впроваджувати функціональність своїх сайтів від інших розробників. Ви можете застосувати до HTML сторонні фреймворки та бібліотеки, що дозволить прискорити створення сайтів.

ПРИВІТ СВІТ

JavaScript є однією з найперспективніших веб-технологій, і коли ви освоїте і почнете використовувати його, ваші веб-сайти перейдуть у новий вимір потужності та креативності.

З JavaScript трохи складніше освоїтися, ніж з HTML та CSS, і вам доведеться почати з малого, продовжуючи вивчення невеликими кроками.

Ми покажемо, як додати деякі основи JavaScript на сторінку, щоб створити "Привіт світ!" приклад. Для початку перейдіть на тестовий сайт і створіть новий файл з ім'ям main.js. Збережіть його у папці scripts. Далі перейдіть в index.html і введіть наступний тег у новий рядок прямо перед тегом, що закриває </body>:

```
<script src="scripts/main.js"></script>
```

Він робить те ж, що й елемент <link> для CSS – додає JavaScript на сторінку, дозволяючи йому працювати з нею.

Тепер додайте такий код файл main.js:

```
var myHeading = document.querySelector('h1');
myHeading.textContent = 'Привіт Світ!';
```

Переконайтеся, що файли HTML та JavaScript збережені, і завантажте index.html у браузері. Причиною, через яку ми поставили елемент <script> у нижній частині HTML файлу, є те, що HTML завантажується у порядку появи його у файлі.

Якщо JavaScript завантажується першим і йому потрібно взаємодіяти з HTML нижче за нього, він не зможе працювати, бо JavaScript буде завантажений раніше, ніж HTML, з яким потрібно працювати. Тому розташовувати JavaScript в нижній частині HTML сторінки вважається гарною практикою.

JavaScript код можна визначити прямо всередині HTML файлу. Для цього використовуйте тег <script></script> без атрибуту src. Всередині цього тегу пишіть так, ніби це JavaScript файл:

```
<script>
var myHeading = document.querySelector('h1');
myHeading.textContent = 'Привіт, Світ!';
</script>
```

ЩО СТАЛОСЯ?

Отже, заголовок тексту було змінено на "Привіт, Світ!" за допомогою JavaScript. Ми зробили це за допомогою виклику функції querySelector(), захопивши посилання на заголовок і зберігши його в змінній myHeading.

Це дуже схоже на те, що ми робили у CSS за допомогою селекторів. Якщо ви хочете щось зробити з елементом, то спочатку вам потрібно його вибрати. Після цього, ми встановлюємо значення змінної myHeading в textContent властивість (що являє собою контент заголовка) "Привіт, Світ!"

Спробуйте вводити JavaScript код у консолі, щоб побачити, що відбувається.

ЗМІННІ

Змінні - це контейнери, всередині яких можна зберігати значення. Ви починаєте оголошувати змінну з ключовим словом var, за яким слідує будь-яке ім'я, яким ви захочете її назвати:

```
var myVariable;
```

ВСІ ІНСТРУКЦІЇ В JAVASCRIPT ПОВИННІ ЗАКІНЧУВАТИСЯ ТОЧКОЮ З КОМОЮ, щоб вказати, де закінчується ця інструкція. Якщо ви не додасте її, то можете отримати несподівані результати.

Ім'я змінної може містити лише літери, цифри або символи $ (символ долара) та _ (символ підкреслення). Перший символ не може бути цифрою. Називайте змінні англійськими літерами, без пропусків.

JavaScript чутливий до регістру - myVariable відрізняється від змінної myvariable.

Після оголошення змінної, ви можете надати їй значення:

```
myVariable = 'Bob';
```

Ви можете зробити обидві ці операції в одному рядку:

```
var myVariable = 'Bob';
```

Ви можете отримати значення, просто викликавши змінну на ім'я:

```
alert(myVariable);
```

Після встановлення значення змінної ви можете змінити його пізніше:

```
var myVariable = 'Bob';
myVariable = 'Steve';
```

Змінні мають різні типи даних:

ТИП ДАНИХ: STRING (рядок)

Щоб показати, що змінна є рядком, помістіть її в лапки.

```
var myVariable = 'Bob';
```

Замість одинарних можна використовувати подвійні лапки.

ТИП ДАНИХ: NUMBER (число)

Числа не мають лапок навколо себе. Числа можна записувати як із десятковими знаками, так і без них.

```
var x = 3.14;
var y = 3;
```

Дуже великі чи дуже малі числа можуть бути записані в експоненційному поданні:

```
var x = 123e5; // 12300000
var y = 123e-5; // 0.00123
```

ТИП ДАНИХ: BOOLEAN (логічний)

Значення true (правда) / false (брехня). Слова true і false - спеціальні ключові слова, і не потребують лапок.

```
var myVariable = true;
```

ТИП ДАНИХ: ARRAY (масив)

Дозволяє зберігати кілька значень в одному посиланні.

```
var myVariable = [1,'Bob','Steve',10];
```

Звернутися до елемента масиву можна так: myVariable[0]. У квадратних дужках пишуть номер (індекс) елементу масиву. Перший елемент має індекс = 0.

ТИП ДАНИХ: OBJECT (об'єкт)

У принципі, що завгодно. Все в JavaScript є об'єктом, і може зберігатися в змінній.

```
var myVar = document.querySelector('h1');
```

КОМЕНТАРІ

Ви можете помістити коментарі в JavaScript код:

```
/*
Все, що знаходиться тут коментар. */
```

Якщо коментар однорядковий:

```
// Це коментар
```

ОПЕРАТОРИ

Оператор - математичний символ, який обчислює результат, заснований на двох значеннях (або змінних).

ОПЕРАТОР: +

Використовується для складання чисел або склеювання рядків.

```
alert(6 + 9); // 15
alert ("Hello" + ", world!"); // Hello, world!
```

Якщо ви додасте два числа, результатом буде число:

```
var x = 10;
var y = 20;
var z = x + y; //30
```

Якщо ви додасте два рядки, результатом буде конкатенація рядків:

```
var x = "10";
var y = "20";
var z = x + y; //1020
```

Якщо додасте число і рядок, результатом буде конкатенація рядків:

```
var x = 10;
var y = "20";
var z = x + y; //1020
```

Якщо ви додасте рядок і число, результатом буде конкатенація рядків:

```
var x = "10";
var y = 20;
var z = x + y; // 1020
```

ОПЕРАТОР: ВІДНІМАННЯ (-), МНОЖЕННЯ (*), ДІЛЕННЯ (/)

Вони роблять те, що ви очікуєте від них в математиці.

```
alert(9 - 3);
alert(8 * 2); // множення
alert(9 / 3) ; // ділення
```

ОПЕРАТОР ПРИСВОЮВАННЯ: =

Надає нове значення змінній.

```
var myVariable = 'Bob';
```

ОПЕРАТОР ПОРІВНЯННЯ: ===

Результатом є ІСТИНА (true), якщо дві величини рівні та мають однакові типи даних. Результатом є БРЕХНЯ (false), якщо дві величини не рівні або мають різні типи даних.

```
alert(myVariable === 4);
```

В Javascript також є оператор порівняння ==, який перевіряє рівність величин без перевірки їх типів даних.

```
alert(myVariable == 4);
```

Приклад:

```
var x = 5;
alert(x == 8); // false
alert(x == 5); // true
alert(x == "5"); // true
alert(x === 5); // true
alert(x === "5"); // false
```

ОПЕРАТОР МЕНШЕ: <

Результатом є ІСТИНА (true), якщо перше значення є меншим ніж друге. В іншому випадку результатом є БРЕХНЯ (false). Приклади:

Число 40<50, результатом є ІСТИНА (true).
Число 60<50, результатом є БРЕХНЯ (false).

ОПЕРАТОР БІЛЬШЕ: >

Результатом є ІСТИНА (true), якщо перша величина більша ніж друга.
Результатом є БРЕХНЯ (false) в іншому випадку.

ОПЕРАТОР МЕНШЕ ЧИ РІВНО: <=

Результатом є ІСТИНА (true), якщо перше значення є меншим ніж друге або рівним йому. В іншому випадку результатом є БРЕХНЯ (false). Приклади:

Число 40<=41, результатом є ІСТИНА (true).
Число 60<=50, результатом є БРЕХНЯ (false).

ОПЕРАТОР БІЛЬШЕ ЧИ РІВНО: >=

Результатом є ІСТИНА (true), якщо перша величина більша ніж друга або дорівнює ій. Результатом є БРЕХНЯ (false) в іншому випадку.

ОПЕРАТОР: НЕ (!), НЕРІВНО (!==)

Повертає логічно протилежне значення, яке цьому передує; перетворює true на false. Коли використовується разом із оператором рівності, перевіряє, чи є два значення нерівними.

ПРИКЛАД 1. Основний вираз true, але порівняння повертає false, тому що ми заперечуємо його:

```
var myVariable = 3;
alert(!(myVariable === 3));
```

ПРИКЛАД 2. Тут ми перевіряємо "myVariable НЕ дорівнює 3". Це повертає false, тому що myVariable дорівнює 3.

```
var myVariable = 3;
alert(myVariable !== 3);
```

ЛОГІЧНИЙ ОПЕРАТОР І: &&

Результатом є ІСТИНА (true), якщо обидва вирази є ІСТИНА. Результатом є БРЕХНЯ (false), якщо хоча б один з виразів є БРЕХНЯ.

X	**Y**	**X && Y**
істина	істина	істина
істина	брехня	брехня
брехня	істина	брехня
брехня	брехня	брехня

Приклад: «Якщо батьки придбають мені кішку І собаку, я буду щасливий». Оператор «І» робить істинним твердження «буду щасливий» тільки за умови, якщо батьки придбають кішку і собаку (обох тварин одночасно).

ЛОГІЧНИЙ ОПЕРАТОР АБО: ||

Результатом є ІСТИНА (true), якщо хоча б один з виразів є ІСТИНА. Результатом є БРЕХНЯ (false), якщо обидва вирази є БРЕХНЯ.

X	**Y**	**X || Y**
істина	істина	істина
істина	брехня	істина
брехня	істина	істина
брехня	брехня	брехня

Приклад: «Якщо батьки придбають мені кішку АБО собаку, то я буду щасливий». Оператор АБО робить істинним твердження «буду щасливий» тільки за умови, якщо батьки придбають кішку АБО собаку (лише одну тварину).

ЩО ЩЕ ТРЕБА ЗНАТИ:

Змішування типів даних може призвести до несподіваних результатів під час виконання обчислень, тому будьте обережні. Коли ви посилаєтеся на змінні правильно, ви отримуєте результати, на які ви очікуєте.

Наприклад, введіть "35" + "25" у консоль. Чому ви не отримали результат, на який ви очікували? Тому що лапки перетворили числа на рядки, тож у результаті вийшла конкатенація рядків, а не додавання чисел.

Якщо ви введете 35+25, то отримаєте правильний результат.

Існує набагато більше операторів для вивчення.

УМОВИ

Умова - структура, яка дозволяє перевірити, чи вираз справжній, а потім виконати різний код залежно від результату. Найпоширеніша форма умови if...else:

```
var iceCream = 'шоколадне';
if (iceCream === 'шоколадне') {
  alert('Так, я люблю шоколадне морозиво!');
} else {
  alert ('Шкода, моє улюблене - шоколадне...');
}
```

Вираз всередині if (...) - це перевірка, яка використовує тотожний оператор, щоб порівняти змінну iceCream з рядком 'шоколадне' і побачити, чи вони рівні. Якщо це порівняння повертає true, виконається перший блок коду. Якщо ні, цей код пропуститься і виконається другий блок коду після else.

If може використовуватися без частини else:

```
if (iceCream === 'шоколадне') {
  alert('Так, я люблю шоколадне морозиво!');
}
```

ФУНКЦІЇ

Функції є способом упаковки коду, який потрібно використовувати повторно. Щоразу, коли ви хочете зробити щось, ви можете просто викликати функцію, а не постійно переписувати весь код. Ви вже бачили деякі види використання функцій, наприклад:

```
var myVariable = document.querySelector('h1');
alert('hello!');
```

Функції, document.querySelector та alert, вбудовані в браузер для того, щоб ви використовували їх, коли це потрібно.

Якщо ви бачите щось, що виглядає як ім'я змінної, але має дужки - () - після нього, швидше за все, це функція. Функції часто приймають аргументи – дані, з якими вони мають виконати свою роботу. Вони передаються в дужки і поділяються комами, якщо існує більше одного аргументу.

Наприклад, функція alert() показує користувачу вікно попередження, але треба задати рядок як аргумент, щоб сказати функції, який текст показувати.

Хорошою новиною є те, що ви можете визначити свої власні функції. Проста функція, яка приймає два числа як аргументи і множить їх:

```
function multiply(num1, num2) {
  var result = num1 * num2;
  return result;
}
```

Спробуйте запустити цю функцію в консолі, потім спробуйте використати нову функцію кілька разів:

```
multiply(4,7);
multiply(20,20);
multiply(0.5,3);
```

Інструкція return повідомляє браузеру повернути змінну result з функції, яку можна використовувати пізніше. Це необхідно, тому що певні змінні всередині функцій доступні тільки всередині цих функцій. Це називається областю видимості змінної.

ПОДІЇ

Для інтерактивних веб-сайтів потрібні події. Події – структура, яка слухає те, що відбувається у браузері, а потім дозволяє запускати код у відповідь на це.

Найбільш очевидною є подія кліка, яка викликається браузером, коли ми клацаємо по чомусь мишею. Для демонстрації цього, спробуйте ввести таку команду в консолі, а потім натисніть на поточній веб-сторінці:

```
document.querySelector('html').onclick = function(){
  alert('Ой! Не натискай на мене!');
}
```

Існує безліч способів прикріпити подію до елементу. Тут ми вибираємо HTML елемент і встановлюємо йому обробник властивості onclick анонімною функцією, яка містить код, що запускається, коли відбувається подія кліка.

```
document.querySelector('html').onclick = function() {};
```

еквівалентно

```
var myHTML = document.querySelector('html');
myHTML.onclick = function() {};
```

ЗМІНА ЗОБРАЖЕННЯ

Давайте додамо ще одне зображення на сайт і створимо код для перемикання між зображеннями, коли на них клацнули. Знайдіть інше зображення, яке ви хотіли б показати на сайті. Переконайтеся, що воно має такий самий розмір, як перше зображення, або близький до нього. Збережіть зображення у папку images. Перейдіть до файлу main.js і введіть JavaScript:

```
var myImage = document.querySelector('img');
myImage.onclick = function() {
  var mySrc = myImage.getAttribute('src');
  if(mySrc === 'images/firefox-icon.png') {
    myImage.setAttribute('src','images/firefox2.png');
  } else {
    myImage.setAttribute('src','images/firefox-icon.png');
  }
```

```
}
```

Збережіть файли та завантажте index.html у браузері. Тепер, коли ви клацнули на картинці, вона повинна змінитися!

Отже, ми зберігаємо посилання на елемент зображення змінної myImage. Далі ми створюємо цей змінний обробник події onclick з анонімною функцією. Тепер кожен раз, коли на цей елемент зображення клацнуть:

Ми отримуємо значення з атрибута src зображення.

Ми використовуємо умову для перевірки значення src, чи дорівнює шлях до вихідного зображення:

> Якщо це так, ми змінюємо значення src на шлях до зображення 2, змушуючи інше зображення завантажуватися всередині елемента <image>.
>
> Якщо це не так (означає, воно мало вже змінитися), ми змінюємо значення src, повертаючись до початкового зображення.

ПРИВІТАННЯ

Далі додамо код, щоб змінити заголовок сторінки та включити персональне вітальне повідомлення, коли користувач вперше заходить на сайт. Це вітальне повідомлення буде зберігатися, коли користувач йде з сайту, а потім повертається. Ми також додамо можливість змінювати користувача.

У index.html додайте рядок перед елементом <script>:

```
<button>Змінити користувача</button>
```

У main.js, додайте такий код в кінець файлу точно так, як написано - він захопить посилання на нову кнопку та заголовок і збереже в змінні:

```
var myButton = document.querySelector('button');
var myHeading = document.querySelector('h1');
```

Тепер додайте функцію для встановлення персоналізованого привітання - вона нічого не робитиме, але ми будемо використовувати її пізніше:

```
function setUserName() {
  var myName = prompt('Введіть ваше прізвище');
  localStorage.setItem('name', myName);
  myHeading.innerHTML = 'Привіт, ' + myName;
}
```

Ця функція містить функцію prompt() , яка викликає діалогове вікно, трохи схоже на alert(), prompt() просить ввести дані та зберігає їх у змінній після того, як користувач натискає OK.

У цьому випадку ми просимо користувача ввести його ім'я. Далі ми викликаємо API localStorage, яке дозволяє зберігати дані у браузері та витягувати їх пізніше.

Ми використовуємо функцію setItem() з localStorage для створення та зберігання даних у властивості під назвою 'name', і встановлюємо це значення у змінну myName, яка містить ім'я, введене користувачем. Наприкінці ми встановлюємо заголовок у вигляді рядка та імені користувача.

Потім додаємо блок if ... else, щоб ми могли викликати код ініціалізації, програма виконує його, коли він вперше завантажується:

```
if(!localStorage.getItem('name')) {
  setUserName();
} else {
  var storedName = localStorage.getItem('name');
  myHeading.innerHTML = 'Привіт, ' + storedName;
}
```

Цей перший блок використовує оператор заперечення (логічне НЕ), щоб перевірити, чи існують дані в пункті name. Якщо ні, то функція setUserName() запускається для створення.

Якщо це так (тобто користувач встановив його під час попереднього відвідування), ми отримуємо збережене ім'я, використовуючи getItem(), і встановлюємо заголовок у вигляді рядка плюс ім'я користувача так само, як ми робили всередині setUserName().

На закінчення, встановимо обробник події onclick на кнопку, щоб при натисканні по ній запускалася функція setUserName(). Це дозволяє користувачеві задавати нове ім'я, натиснувши кнопку:

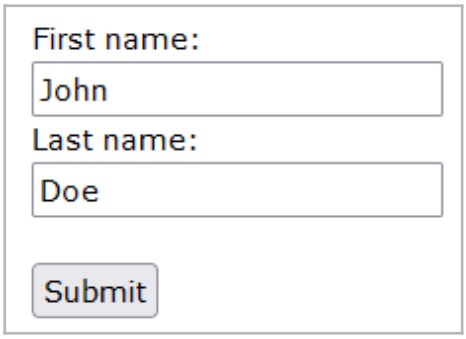
```
myButton.onclick = function() {
  setUserName();
}
```

Тепер, коли ви вперше відвідаєте сайт, ми просимо вас ввести ім'я користувача, а потім надамо персональне повідомлення. Потім ви можете змінити ім'я в будь-який момент, натиснувши кнопку. Ім'я збережеться після закриття сайту, тому персональне повідомлення все ще буде там, коли ви відкриєте сайт знову!

HTML 2

ФОРМИ

Форма HTML використовується для збору даних, введених користувачем. Введені дані найчастіше надсилаються на сервер для обробки.



ЕЛЕМЕНТ <form>

Елемент <form> використовується для створення форми для введення даних:

```
<form>
.
елементи форми
.
</form>
```

Елемент <form> є контейнером для різних типів елементів введення, таких як: текстові поля, прапорці, перемикачі, кнопки надсилання тощо.

ЕЛЕМЕНТ <input>

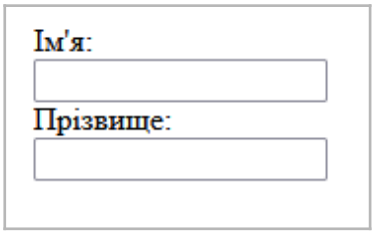
Елемент <input> є найбільш використовуваним елементом форми. Елемент <input> можна відобразити різними способами, залежно від атрибута type. Приклади:

Тип	Опис
<input type="text">	Відображає однорядкове поле введення тексту
<input type="radio">	Відображає перемикач для вибору одного з багатьох варіантів
<input type="checkbox">	Відображає прапорець
<input type="submit">	Відображає кнопку надсилання форми
<input type="button">	Відображає кнопку, яку можна натиснути

ТЕКСТОВІ ПОЛЯ

<input type="text"> визначає однорядкове поле для введення тексту. Форма з полями введення тексту:

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```



Сама форма не відображається. Також зауважте, що стандартна ширина поля введення становить 20 символів.

ЕЛЕМЕНТ <label>

Зверніть увагу на використання елементу <label> у прикладі вище. Тег <label> визначає мітку для багатьох елементів форми. Елемент <label> корисний для користувачів програм зчитування з екрана, оскільки програма вголос читатиме мітку, коли користувач фокусується на елементі введення.

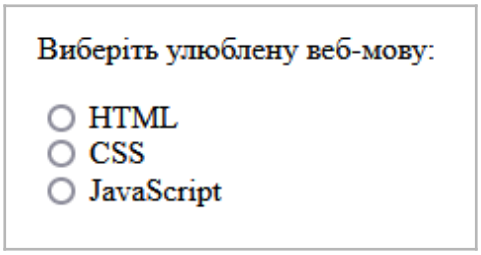
Елемент <label> також допомагає користувачам, яким важко клацати на дуже маленькі області (наприклад, перемикачі чи прапорці), оскільки коли користувач клацає текст всередині елемента <label>, він перемикає перемикач/прапорець.

Атрибут for тегу <label> має дорівнювати атрибуту id елемента <input>, щоб зв’язати їх разом.

ПЕРЕМИКАЧІ

<input type="radio"> визначає перемикач. Перемикачі дозволяють користувачеві вибрати ОДИН із обмеженої кількості варіантів. Форма з перемикачами:

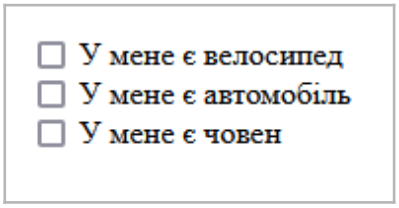
```
<p>Виберіть улюблену веб-мову:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```



ПРАПОРЦІ

<input type="checkbox"> визначає прапорець. Прапорці дозволяють користувачеві вибрати НУЛЬ або БІЛЬШЕ варіантів з обмеженої кількості варіантів. Приклад:

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1">У мене є велосипед</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2">У мене є автомобіль</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3">У мене є човен</label>
</form>
```

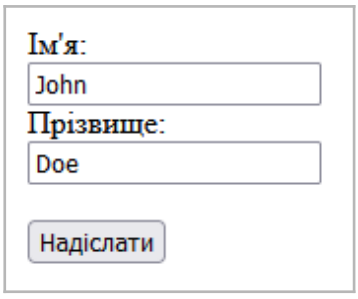


КНОПКА «НАДІСЛАТИ»

<input type="submit"> визначає кнопку для надсилання даних форми обробнику форми. Обробник форми зазвичай є файлом на сервері зі сценарієм для обробки вхідних даних. Обробник форми вказується в атрибуті дії форми.

Форма з кнопкою надсилання:

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Надіслати">
</form>
```



АТРИБУТ name ДЛЯ <input>

Зауважте, що кожне поле введення повинно мати атрибут name для надсилання. Якщо атрибут name пропущено, значення поля введення не надсилатиметься взагалі. У цьому прикладі не надсилатиметься значення поля введення «Ім’я»:

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" value="John"><br><br>
  <input type="submit" value="Submit">
</form>
```

АТРИБУТИ ФОРМИ

У цьому розділі описуються різні атрибути для елементу <form>.

АТРИБУТ action

Атрибут action визначає дію, яка буде виконана під час надсилання форми. Зазвичай дані форми надсилаються у файл на сервері, коли користувач натискає кнопку відправки. У наведеному прикладі дані форми надсилаються у файл під назвою «action_page.php». Цей файл містить сценарій, який обробляє дані форми.

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

Якщо атрибут action пропущено, він дорівнює адресі поточної сторінки.

АТРИБУТ target

Атрибут target визначає, де відображати відповідь, отриману після надсилання форми. Цільовий атрибут може мати одне з таких значень:

_blank – відповідь відображається в новому вікні чи вкладці.

_self - відповідь відображається в поточному вікні.

_parent – відповідь відображається в батьківському фреймі.

_top - Відповідь відображається у всьому тілі вікна.

framename - відповідь відображається в іменованому iframe.

Значення за замовчуванням - _self. Воно означає, що відповідь відкриється в поточному вікні. Тут надісланий результат відкриється в новій вкладці веб-переглядача:

```
<form action="/action_page.php" target="_blank">
```

АТРИБУТ method

Атрибут method визначає метод HTTP, який буде використовуватися під час надсилання даних форми. Дані форми можна надіслати як змінні URL-адреси (з method="get") або як пост-транзакцію HTTP (з method="post"). Метод HTTP за замовчуванням під час надсилання даних форми - GET. У цьому прикладі використовується метод GET під час надсилання даних форми:

```
<form action="/action_page.php" method="get">
```

У цьому прикладі використовується метод POST під час надсилання даних форми:

```
<form action="/action_page.php" method="post">
```

Примітки щодо GET:

- Додає дані форми до URL-адреси в парах ім’я/значення.
- НІКОЛИ не використовуйте GET для надсилання конфіденційних даних. (надіслані дані форми видно в URL).
- Довжина URL-адреси обмежена (2048 символів).
- Корисно для надсилання форм, коли треба додати результат до закладок.
- GET добре підходить для незахищених даних.

Примітки щодо POST:

- Додає дані форми в тіло запиту HTTP (надіслані дані форми не відображаються в URL-адресі).
- POST не має обмежень за розміром і може використовуватися для надсилання великих обсягів даних.
- Надсилання форм за допомогою POST не можна додати до закладок.

Завжди використовуйте POST, якщо дані форми містять конфіденційну інформацію!

АТРИБУТ autocomplete

Атрибут autocomplete визначає, увімкнути чи вимкнути автозаповнення у формі. Коли автозаповнення ввімкнено, браузер автоматично доповнює значення на основі значень, які користувач ввів раніше. Форма з увімкненим автозаповненням:

```
<form action="/action_page.php" autocomplete="on">
```

АТРИБУТ novalidate

Атрибут novalidate є булевим атрибутом. Якщо він присутній, він визначає, що дані форми (вхідні дані) не повинні перевірятися під час надсилання. Приклад:

```
<form action="/action_page.php" novalidate>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
</form>
```

ІНШІ АТРИБУТИ

Атрибут	Опис
accept-charset	Визначає кодування символів, що використовуються для надсилання форми.
enctype	Вказує, як мають бути закодовані дані форми під час надсилання на сервер (тільки для method="post").
name	Вказує назву форми.
rel	Вказує зв'язок між пов'язаним ресурсом і поточним документом.

ЕЛЕМЕНТИ ФОРМИ

Елемент <form> може містити один або кілька таких елементів форми:

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>

ЕЛЕМЕНТ <input>

Одним із найбільш використовуваних елементів форми є елемент <input>. Елемент <input> можна відобразити кількома способами, залежно від атрибута type.

```
<label for="fname">Ім'я:</label>
<input type="text" id="fname" name="fname">
```

ЕЛЕМЕНТ <label>

Елемент <label> визначає мітку для кількох елементів форми. Елемент <label> корисний для користувачів програм зчитування з екрана, оскільки програма зчитування з екрана вголос читатиме мітку, коли користувач фокусується на елементі введення. Елемент <label> також допомагає користувачам, яким важко клацати дуже маленькі області (наприклад, перемикачі або прапорці), оскільки коли користувач клацає текст всередині елемента <label>, він перемикає перемикач/прапорець. Атрибут for тегу <label> має дорівнювати атрибуту id елемента <input>, щоб зв’язати їх разом.

ЕЛЕМЕНТ <select>

Елемент <select> визначає розкривний список:

```
<label for="cars">Виберіть автомобіль:</label>
<select id="cars" name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Елемент <option> визначає параметр, який можна вибрати. За замовчуванням вибрано перший пункт у розкривному списку. Щоб визначити попередньо вибрану опцію, додайте вибраний атрибут до опції:

```
<option value="fiat" selected>Fiat</option>
```

Використовуйте атрибут size, щоб вказати кількість видимих значень:

```
<label for="cars">Виберіть автомобіль:</label>
<select id="cars" name="cars" size="3">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Використовуйте атрибут multiple, щоб дозволити користувачеві вибрати більше одного значення:

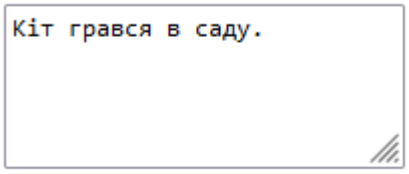
```
<label for="cars">Виберіть автомобіль:</label>
<select id="cars" name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

ЕЛЕМЕНТ <textarea>

Елемент <textarea> визначає багаторядкове поле введення (текстова область):

```
<textarea name="message" rows="10" cols="30">
Кіт грався в саду.
</textarea>
```

Атрибут rows визначає видиму кількість рядків у текстовій області. Атрибут cols визначає видиму ширину текстової області. Ось як наведений вище HTML-код відображатиметься у браузері:



Ви також можете визначити розмір текстової області за допомогою CSS:

```
<textarea name="message" style="width:200px; height:600px;">
Котик грався в саду.
</textarea>
```

ТИПИ ВВЕДЕННЯ

Ось різні типи введення, які можна використовувати в HTML:

- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date" >
- <input type="datetime-local">
- <input type="email">
- <input type="file">
- <input type="hidden">
- <input type="image">
- <input type="month" >
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text">
- <input type="time">
- <input type="url">
- <input type="week">

Значення за умовчанням атрибута типу є "text".

ТИП ВВЕДЕННЯ text (текст)

<input type="text"> визначає однорядкове поле введення тексту:

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

ТИП ВВЕДЕННЯ password (пароль)

<input type="password"> визначає поле пароля. Символи в полі пароля замасковані (відображаються у вигляді зірочок або кружечків).

```
<form>
  <label for="username">Ім'я користувача:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Пароль:</label><br>
  <input type="password" id="pwd" name="pwd">
</form>
```

ТИП ВВЕДЕННЯ submit (кнопка для надсилання даних)

<input type="submit"> визначає кнопку для надсилання даних форми обробнику форми. Обробник форми зазвичай є серверною сторінкою зі сценарієм для обробки вхідних даних. Обробник форми вказується в атрибуті дії форми:

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Надіслати">
</form>
```

Якщо ви опустите атрибут value кнопки надсилання, кнопка отримає текст за замовчуванням:

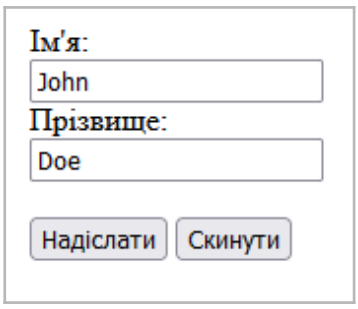
```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit">
</form>
```

ТИП ВВЕДЕННЯ reset (скидає всі значення форми)

<input type="reset"> визначає кнопку скидання, яка змінює всі значення форми на значення за замовчуванням:

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
```

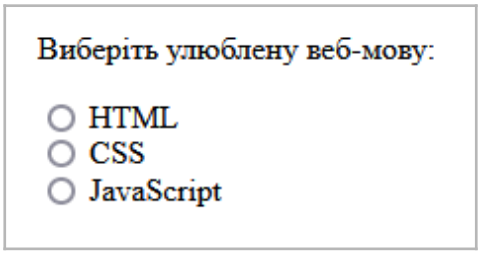
```
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Надіслати">
  <input type="reset" value="Скинути">
</form>
```



Якщо змінити введені значення, а потім натиснути кнопку «Скинути», дані форми буде скинуто до значень за замовчуванням.

ТИП ВВЕДЕННЯ radio (перемикач)

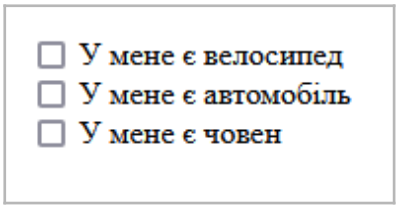
<input type="radio"> визначає перемикач. Перемикачі дозволяють користувачеві вибрати ЛИШЕ ОДИН з обмеженої кількості варіантів:




```
<p>Виберіть улюблену веб-мову:</p>
<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

ТИП ВВЕДЕННЯ checkbox (прапорець)

<input type="checkbox"> визначає прапорець. Прапорці дозволяють користувачеві вибрати НУЛЬ або БІЛЬШЕ варіантів з обмеженої кількості варіантів.




```
<form>
```

```
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1">У мене є велосипед</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2">У мене є автомобіль</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3">У мене є човен</label>
</form>
```

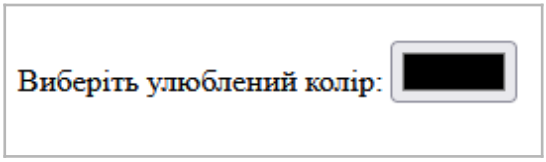
ТИП ВВЕДЕННЯ button (кнопка)

<input type="button"> визначає кнопку:

```
<input type="button" onclick="alert('Привіт')" value="Натисни мене">
```

ТИП ВВЕДЕННЯ color (колір)

<input type="color"> використовується для полів введення, які мають містити колір.

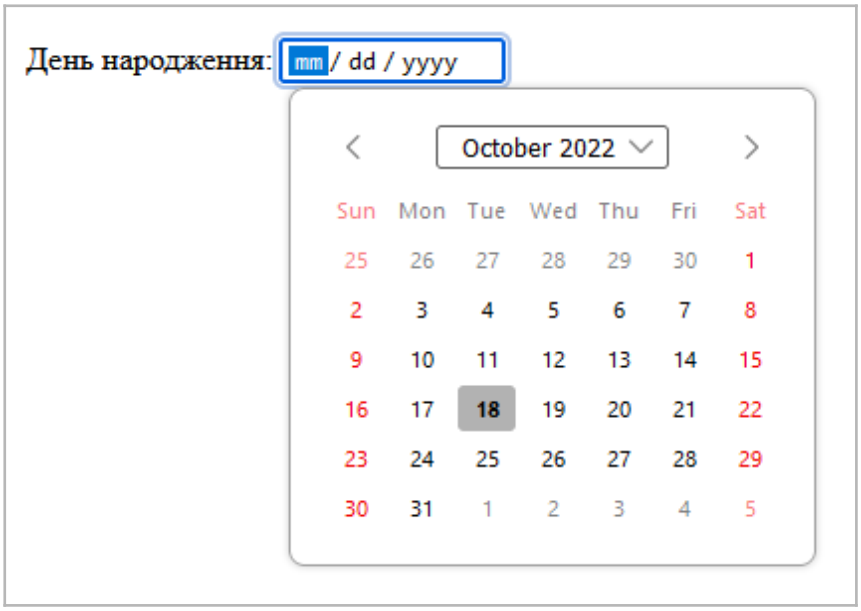



```
<form>
  <label for="favcolor">Виберіть улюблений колір:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

ТИП ВВЕДЕННЯ date (дата)

<input type="date"> використовується для полів введення, які мають містити дату.

```
<form>
  <label for="birthday">День народження:</label>
  <input type="date" id="birthday" name="birthday">
</form>
```



Можна використовувати атрибути min та max, щоб додати обмеження на дати:

```
<form>
  <label for="datemax">Введіть дату до 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>
  <label for="datemin">Введіть дату після 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02">
</form>
```

ТИП ВВЕДЕННЯ datetime-local

<input type="datetime-local"> визначає поле введення дати і часу без часового поясу. Залежно від підтримки веб-переглядача у полі введення може відображатися засіб вибору дати.


```
<form>
  <label for="birthdaytime">День народження (дата й час):</label>
  <input type="datetime-local" id="birthdaytime" name="birthdaytime">
</form>
```

ТИП ВВЕДЕННЯ email (електронна адреса)

<input type="email"> використовується для полів введення, які мають містити адресу електронної пошти.

```
<form>
  <label for="email">Введіть свою електронну адресу:</label>
  <input type="email" id="email" name="email">
</form>
```

ТИП ВВЕДЕННЯ image (зображення)

<input type="image"> визначає зображення як кнопку надсилання. Шлях до зображення вказується в атрибуті src.

```
<form>
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```

ТИП ВВЕДЕННЯ file (файл)

<input type="file"> визначає поле вибору та кнопку для завантаження файлів.

```
<form>
  <label for="myfile">Виберіть файл:</label>
  <input type="file" id="myfile" name="myfile">
</form>
```

ТИП ВВЕДЕННЯ hidden (приховане поле)

<input type="hidden"> визначає приховане поле введення (невидиме для користувача). Приховане поле дозволяє веб-розробникам включати дані, які користувачі не можуть побачити або змінити під час надсилання форми.

У прихованому полі часто зберігається запис бази даних, який потрібно оновити під час надсилання форми. Хоча значення не відображається у вмісті сторінки, воно видиме (і може бути відредаговано) за допомогою інструментів розробника будь-якого браузера.

```
<form>
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
```

ТИП ВВЕДЕННЯ month (місяць)

<input type="month"> дозволяє користувачеві вибрати місяць і рік.

```
<form>
  <label for="bdaymonth">День народження (місяць і рік):</label>
  <input type="month" id="bdaymonth" name="bdaymonth">
</form>
```

ТИП ВВЕДЕННЯ number (число)

<input type="number"> визначає числове поле введення. Ви також можете встановити обмеження на те, які номери приймаються. У наведеному прикладі показане числове поле введення, куди можна ввести значення від 1 до 5:

```
<form>
  <label for="quantity">Кількість (від 1 до 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

ОБМЕЖЕННЯ ВВЕДЕННЯ

Ось список деяких поширених обмежень на введення:

Атрибут	Опис
checked	Вказує, що поле введення має бути попередньо вибрано під час завантаження сторінки (для type="checkbox" або type="radio")
disabled	Вказує, що поле введення має бути вимкнено
max	Вказує максимальне значення для поля введення
maxlength	Вказує максимальну кількість символів для поля введення
min	Вказує мінімальне значення для поля введення
pattern	Вказує регулярний вираз для перевірки вхідного значення
readonly	Вказує, що поле введення неможливо змінити
required	Вказує, що поле введення є обов’язковим (має бути заповненим)
size	Визначає ширину (в символах) поля введення
step	Вказує допустимі інтервали чисел для значення поля введення
value	Вказує значення за замовчуванням для поля введення

У наступному прикладі показано числове поле введення, куди можна ввести значення від 0 до 100 із кроком 10. Значення за замовчуванням - 30:

```
<form>
  <label for="quantity">Кількість:</label>
  <input type="number" id="quantity" name="quantity" min="0" max="100" step="10"
value="30">
</form>
```

ТИП ВВЕДЕННЯ range (діапазон)

<input type="range"> визначає елемент керування для введення числа, точне значення якого не є важливим (наприклад, повзунок). Діапазон за замовчуванням становить від 0 до 100. Однак ви можете встановити обмеження на те, які числа приймаються за допомогою атрибутів min, max і step:

```
<form>
  <label for="vol">Гучність (від 0 до 50):</label>
  <input type="range" id="vol" name="vol" min="0" max="50">
</form>
```

ТИП ВВЕДЕННЯ search (пошук)

<input type="search"> використовується для полів пошуку (поле поводиться як звичайне текстове поле).

```
<form>
  <label for="gsearch">Пошук у Google:</label>
  <input type="search" id="gsearch" name="gsearch">
</form>
```

ТИП ВВЕДЕННЯ tel (номер телефону)

<input type="tel"> використовується для введення номера телефону.

```
<form>
  <label for="phone">Введіть свій номер телефону:</label>
  <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]
{3}">
</ form>
```

ТИП ВВЕДЕННЯ time (час без часового поясу)

<input type="time"> дозволяє користувачеві вибрати час (без часового поясу).

```
<form>
  <label for="appt">Виберіть час:</label>
  <input type="time" id="appt" name="appt">
</form>
```

ТИП ВВЕДЕННЯ url (URL-адреса)

<input type="url"> використовується для полів введення, які мають містити URL-адресу. Залежно від підтримки браузера, поле URL-адреси може автоматично перевірятися під час надсилання.

```
<form>
  <label for="homepage">Додайте свою домашню сторінку:</label>
  <input type="url" id="homepage" name="homepage">
</form>
```

ТИП ВВЕДЕННЯ week (тиждень)

<input type="week"> дозволяє користувачеві вибрати тиждень і рік.

```
<form>
  <label for="week">Виберіть тиждень:</label>
```

```
  <input type="week" id="week" name="week">
</form>
```

АТРИБУТИ ДЛЯ <input>

АТРИБУТ value

Атрибут value визначає початкове значення для поля введення:

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

АТРИБУТ readonly

Атрибут readonly визначає, що поле введення є лише для читання. Поле введення лише для читання не можна змінити (однак користувач може перейти до нього табуляцією, виділити його та скопіювати з нього текст). Значення поля введення лише для читання буде надіслано під час надсилання форми.

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John" readonly><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

АТРИБУТ disabled

Атрибут disabled визначає, що поле введення має бути вимкнено. Поле введення буде непридатним для використання. Значення вимкненого поля введення не буде надіслано під час надсилання форми.

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

АТРИБУТ size

Атрибут size визначає видиму ширину поля введення в символах. Значення розміру за замовчуванням - 20. Атрибут size працює з такими типами введення: текст, пошук, tel, url, email і пароль.

```
<form>
```

```
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN-код:</label><br>
  <input type="text" id="pin" name="pin" size="4">
</form>
```

АТРИБУТ maxlength

Атрибут maxlength визначає максимальну кількість символів, дозволену в полі введення. Якщо встановлено максимальну довжину, поле введення не прийматиме більше за вказану кількість символів. Цей атрибут не забезпечує зворотного зв'язку. Якщо ви хочете попередити користувача, треба написати код JavaScript.

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN-код:</label><br>
  <input type="text" id="pin" name="pin" maxlength="4" size="4">
</form>
```

АТРИБУТИ min і max

Атрибути min і max визначають мінімальне та максимальне значення для введення в поле. Атрибути min і max працюють з такими типами введення: число, діапазон, дата, дата-час-локальний, місяць, час і тиждень. Використовуйте атрибути max і min разом, щоб створити діапазон допустимих значень.

```
<form>
  <label for="datemax">Введіть дату до 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>

  <label for="datemin">Введіть дату після 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02"><br><br>

  <label for="quantity">Кількість (від 1 до 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

АТРИБУТ multiple

Атрибут multiple визначає, що користувачеві дозволено вводити більше одного значення в поле введення. Атрибут multiple працює з такими типами введення: електронна пошта та файл.

```
<form>
  <label for="files">Виберіть файли:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```

АТРИБУТ pattern

Атрибут шаблону введення визначає регулярний вираз, за яким перевіряється значення поля введення під час надсилання форми. Атрибут шаблону працює з такими типами введення: текст, дата, пошук, URL-адреса, телефон, електронна пошта та пароль. Використовуйте глобальний **title**, щоб описати шаблон, щоб допомогти користувачеві. Поле введення, яке може містити лише три літери (без цифр або спеціальних символів):

```
<form>
  <label for="country_code">Код країни:</label>
  <input type="text" id="country_code" name="country_code"
  pattern="[A-Za-z]{3}" title="Трибуквений код країни">
</form>
```

АТРИБУТ placeholder

Атрибут заповнювача введення визначає коротка підказка, яка описує очікуване значення поля введення (приклад значення або короткий опис очікуваного формату). Коротка підказка відображається в полі введення перед тим, як користувач введе значення. Атрибут заповнювача працює з такими типами введення: текст, пошук, URL-адреса, телефон, електронна пошта та пароль.

```
<form>
  <label for="phone">Введіть номер телефону:</label>
  <input type="tel" id="phone" name="phone"
  placeholder="123-45-678"
  pattern="[0-9]{3}-[0-9]{2}-[0- 9]{3}">
</form>
```

АТРИБУТ required

Атрибут required визначає, що поле введення має бути заповнене перед надсиланням форми. Необхідний атрибут працює з такими типами введення: текст, пошук, URL-адреса, телефон, електронна пошта, пароль, засоби вибору дати, номер, прапорець, радіо та файл.

```
<form>
  <label for="username">Ім'я користувача:</label>
  <input type="text" id="username" name="username" required>
</form>
```

АТРИБУТ step

Атрибут step вказує допустимі інтервали чисел для поля введення. Приклад: якщо step="3", дозволені числа можуть бути -3, 0, 3, 6 тощо. Цей атрибут можна використовувати разом із атрибутами max і min для створення діапазону допустимих значень. Атрибут step працює з такими типами введення: число, діапазон, дата, дата-час-локальний, місяць, час і тиждень.

```
<form>
  <label for="points">Бали:</label>
  <input type="number" id="points" name="points" step="3">
</form>
```

Обмеження введення не є безпомилковими, і JavaScript надає багато способів додавати незаконні введення. Щоб безпечно обмежити введення, його також має перевірити отримувач (сервер).

АТРИБУТ autofocus

Атрибут autofocus вказує, що поле введення має автоматично отримувати фокус під час завантаження сторінки. Нехай поле введення «Ім’я» автоматично отримує фокус під час завантаження сторінки:

```
<form>
  <label for="fname">Ім'я:</label><br>
  <input type="text" id="fname" name="fname" autofocus><br>
  <label for="lname">Прізвище:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

АТРИБУТИ height і width

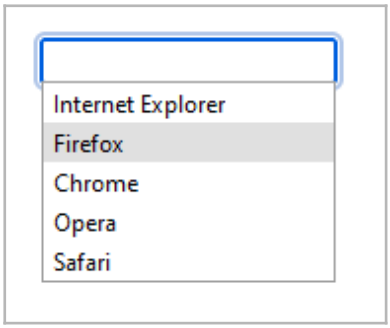
Атрибути визначають висоту і ширину елемента <input type="image">. Завжди вказуйте атрибути як висоти, так і ширини для зображень. Якщо встановлено висоту та ширину, то місце, необхідне для зображення, резервується під час завантаження сторінки. Без цих атрибутів браузер не знає розміру зображення та не може зарезервувати для нього відповідний простір. В результаті макет сторінки буде змінюватися під час завантаження.

Визначте зображення як кнопку надсилання з атрибутами висоти та ширини:

```
<form>
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Прізвище:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
</form>
```

АТРИБУТ list

Атрибут list відноситься до елементу <datalist>, який містить попередньо визначені параметри для елементу <input>. Елемент <input> із попередньо визначеними значеннями в <datalist>:




```
<form>
  <input list="браузери">
  <datalist id="браузери">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

АТРИБУТ autocomplete

Атрибут autocomplete вказує, увімкнути чи вимкнути автозаповнення для форми чи поля введення. Автозаповнення дозволяє браузеру передбачити значення. Коли користувач починає вводити текст у поле, браузер має відобразити варіанти заповнення поля на основі раніше введених значень. Атрибут autocomplete працює з <form> і такими типами <input>: текст, пошук, url, tel, електронна пошта, пароль, засоби вибору дати, діапазон і колір.

HTML-форма з увімкненим і вимкненим автозаповненням для одного поля введення:

```
<form action="/action_page.php" autocomplete="on">
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Прізвище:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <label for="email">Електронна пошта:</label>
  <input type="email" id="email" name="email" autocomplete="off"><br><br>
  <input type="submit" value="Submit">
</form>
```

У деяких браузерах вам може знадобитися активувати функцію автозаповнення, щоб це працювало (перегляньте розділ «Налаштування» в меню браузера).

АТРИБУТ form

Атрибут form визначає форму, до якої належить елемент <input>. Значення цього атрибута має дорівнювати атрибуту id елемента <form>, якому він належить. Поле введення, розташоване поза формою HTML (але все ще є частиною форми):

```
<form action="/action_page.php" id="form1">
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>

<label for="lname">Прізвище:</label>
<input type="text" id="lname" name="lname" form ="form1">
```

АТРИБУТ formaction

Атрибут formaction визначає URL-адресу файлу, який оброблятиме вхідні дані після надсилання форми. Цей атрибут перевизначає атрибут дії елементу <form>.

Атрибут formation працює з такими типами введення: submit і image. HTML-форма з двома кнопками надсилання з різними діями:

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Прізвище:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Надіслати">
  <input type="submit" formaction="/action_page2.php"
value="Надіслати як адміністратор">
</form>
```

АТРИБУТ formenctype

Атрибут formenctype визначає спосіб кодування даних форми під час надсилання (лише для форм із method= "post"). Цей атрибут перевизначає атрибут enctype елементу <form>. Атрибут formenctype працює з типами введення: submit і image.

Форма з двома кнопками надсилання. Перший надсилає дані форми з кодуванням за замовчуванням, другий надсилає дані форми, закодовані як "multipart/form-data":

```
<form action="/action_page_binary.asp" method="post">
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

АТРИБУТ formmethod

Атрибут formmethod визначає метод HTTP для надсилання даних форми до URL-адреси вказаної в атрибуті action. Атрибут formmethod перевизначає атрибут method елементу <form>.

Атрибут formmethod працює з такими типами введення: submit і image. Дані форми можна надіслати як змінні URL-адреси (method="get") або як пост-транзакцію HTTP (method="post").

Примітки щодо методу "get":

- Цей метод додає дані форми до URL-адреси в парах ім’я/значення.
- Цей метод корисний для надсилання форм, коли користувач хоче додати результат до закладок.
- Існує обмеження на кількість даних, які ви можете розмістити в URL-адресі (залежить від браузера), тому ви не можете бути впевнені, що всі дані форми будуть правильно передані.
- Ніколи не використовуйте метод "get" для передачі конфіденційної інформації! (пароль або іншу конфіденційну інформацію буде видно в адресному рядку браузера).

Примітки щодо методу "post":

- Цей метод надсилає дані форми як транзакцію HTTP post.
- Надсилання форм за допомогою "post" не можна додавати в закладки.
- Метод "post" надійніший і безпечніший, ніж "get", а "post" не має обмежень щодо розміру.

Форма з двома кнопками надсилання. Перша надсилає дані форми за допомогою method="get". Друга надсилає дані форми з method="post":

```
<form action="/action_page.php" method="get">
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Прізвище:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Надіслати через GET">
  <input type="submit" formmethod="post" value="Надіслати через POST">
</form>
```

АТРИБУТ formtarget

Атрибут formtarget визначає назву або ключове слово, яке вказує, де відображати відповідь, отриману після надсилання форми. Цей атрибут замінює цільовий атрибут елементу <form>. Атрибут formtarget працює з типами введення: submit і image.

Форма з двома кнопками надсилання з різними цільовими вікнами:

```
<form action="/action_page.php">
  <label for="fname">Ім'я:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Прізвище:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Надіслати">
  <input type="submit" formtarget="_blank"
value="Надіслати та відкрити нове вікно">
</form>
```

АТРИБУТ formnovalidate

Атрибут formnovalidate визначає, що елемент <input> не повинен перевірятися під час надсилання. Цей атрибут замінює атрибут novalidate елементу <form>. Атрибут formnovalidate працює з такими типами введення: submit.

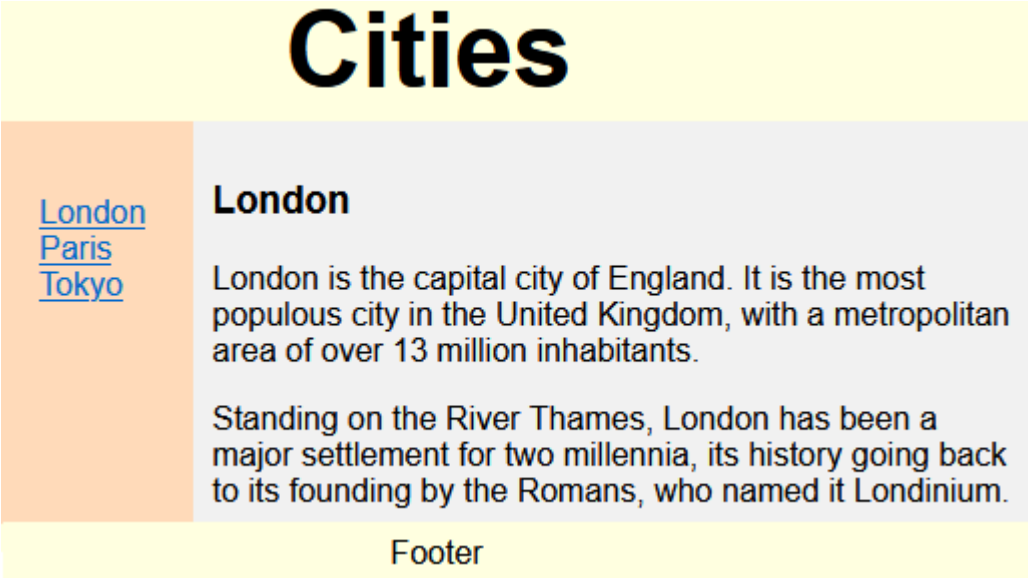
Форма з двома кнопками надсилання (з перевіркою та без неї):

```
<form action="/action_page.php">
  <label for="email">Введіть свою електронну адресу:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formnovalidate="formnovalidate"
  value="Submit without validation">
</form>
```

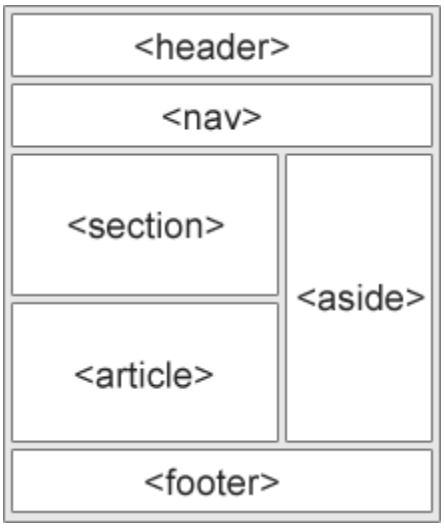
HTML 3

ЕЛЕМЕНТИ ТА МЕТОДИ РОЗМІТКИ

Веб-сайти часто відображають контент у кількох стовпцях.



HTML має кілька семантичних елементів, які визначають різні частини веб-сторінки:



- <header> - Визначає заголовок документа чи розділу.
- <nav> - Визначає набір навігаційних посилань.
- <section> - Визначає розділ у документі.
- <article> - Визначає незалежний, автономний контент.
- <aside> - Визначає бічну панель.
- <footer> - Визначає нижній колонтитул документа або розділу.
- <details> - Визначає додаткові деталі, які користувач може відкривати та закривати на вимогу.
- <summary> - Визначає заголовок для <details> елементу.

Існують різні методи створення багатоколонних макетів.

- CSS-ФРЕЙМВОРК. Якщо ви бажаєте швидко створити свій макет, ви можете використовувати фреймворк CSS, такий як W3.CSS або Bootstrap.
- ПЛАВАЮЧИЙ МАКЕТ CSS. Загально прийнято робити цілі веб-макети, використовуючи float властивість CSS. Float легко освоїти, вам просто потрібно пам'ятати, як працюють властивості float.
- CSS ФЛЕКСБОКС. Використання flexbox гарантує, що елементи поводяться передбачувано, коли макет сторінки повинен відповідати різним розмірам екрана та різним пристроям відображення. Недоліки: Flexbox не підтримується в Internet Explorer 10 та ранніх версіях.
- CSS-сітка. Модуль CSS Grid Layout пропонує систему компонування на основі сітки з рядками та стовпцями, що спрощує розробку веб-сторінок без використання плаваючих елементів та позиціонування.

Багато прикладів розмітки сайтів:

https://www.w3schools.com/w3css/w3css_templates.asp

ПРИКЛАД LAYOUT FLOAT

Джерело:
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_float

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Template</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

body {
  font-family: Arial, Helvetica, sans-serif;
}

/* Style the header */
header {
  background-color: #666;
  padding: 30px;
  text-align: center;
  font-size: 35px;
  color: white;
}
```

```
/* Create two columns/boxes that floats next to each other */
nav {
  float: left;
  width: 30%;
  height: 300px; /* only for demonstration, should be removed */
  background: #ccc;
  padding: 20px;
}

/* Style the list inside the menu */
nav ul {
  list-style-type: none;
  padding: 0;
}

article {
  float: left;
  padding: 20px;
  width: 70%;
  background-color: #f1f1f1;
  height: 300px; /* only for demonstration, should be removed */
}

/* Clear floats after the columns */
section::after {
  content: "";
  display: table;
  clear: both;
}

/* Style the footer */
footer {
  background-color: #777;
  padding: 10px;
  text-align: center;
  color: white;
}

/* Responsive layout - makes the two columns/boxes stack on top of each other
instead of next to each other, on small screens */
@media (max-width: 600px) {
  nav, article {
        width: 100%;
        height: auto;
  }
}
</style>
```

```
</head>
<body>

<h2>CSS Layout Float</h2>
<p>In this example, we have created a header, two columns/boxes and a footer. On smaller screens, the columns will stack on top of each other.</p>
<p>Resize the browser window to see the responsive effect (you will learn more about this in our next chapter - HTML Responsive.)</p>

<header>
  <h2>Cities</h2>
</header>

<section>
  <nav>
        <ul>
        <li><a href="#">London</a></li>
        <li><a href="#">Paris</a></li>
        <li><a href="#">Tokyo</a></li>
        </ul>
  </nav>

  <article>
        <h1>London</h1>
        <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
        <p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>
  </article>
</section>

<footer>
  <p>Footer</p>
</footer>

</body>
</html>
```

ПРИКЛАД LAYOUT FLEXBOX

Джерело:
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_flexbox

```
<!DOCTYPE html>
```

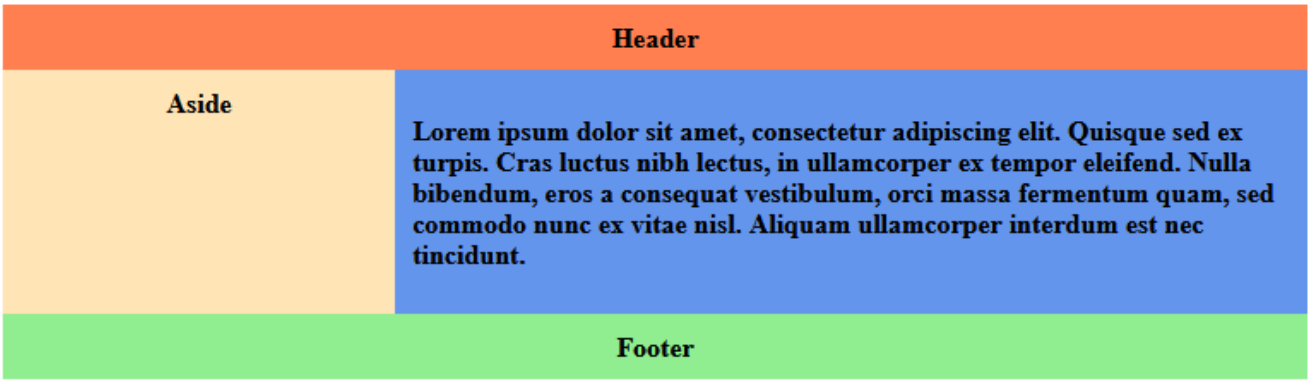
```
<html lang="en">
<head>
<title>CSS Template</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

body {
  font-family: Arial, Helvetica, sans-serif;
}

/* Style the header */
header {
  background-color: #666;
  padding: 30px;
  text-align: center;
  font-size: 35px;
  color: white;
}

/* Container for flexboxes */
section {
  display: -webkit-flex;
  display: flex;
}

/* Style the navigation menu */
nav {
  -webkit-flex: 1;
  -ms-flex: 1;
  flex: 1;
  background: #ccc;
  padding: 20px;
}

/* Style the list inside the menu */
nav ul {
  list-style-type: none;
  padding: 0;
}

/* Style the content */
article {
  -webkit-flex: 3;
```

```
  -ms-flex: 3;
  flex: 3;
  background-color: #f1f1f1;
  padding: 10px;
}

/* Style the footer */
footer {
  background-color: #777;
  padding: 10px;
  text-align: center;
  color: white;
}

/* Responsive layout - makes the menu and the content (inside the section) sit on top of each other instead of next to each other */
@media (max-width: 600px) {
  section {
        -webkit-flex-direction: column;
        flex-direction: column;
  }
}
</style>
</head>
<body>

<h2>CSS Layout Flexbox</h2>
<p>In this example, we have created a header, two columns/boxes and a footer. On smaller screens, the columns will stack on top of each other.</p>
<p>Resize the browser window to see the responsive effect.</p>
<p><strong>Note:</strong> Flexbox is not supported in Internet Explorer 10 and earlier versions.</p>

<header>
  <h2>Cities</h2>
</header>

<section>
  <nav>
        <ul>
        <li><a href="#">London</a></li>
        <li><a href="#">Paris</a></li>
        <li><a href="#">Tokyo</a></li>
        </ul>
  </nav>

  <article>
```

```
        <h1>London</h1>
        <p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
        <p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>
  </article>
</section>

<footer>
  <p>Footer</p>
</footer>

</body>
</html>
```

ПРИКЛАД LAYOUT FLEXBOX 2

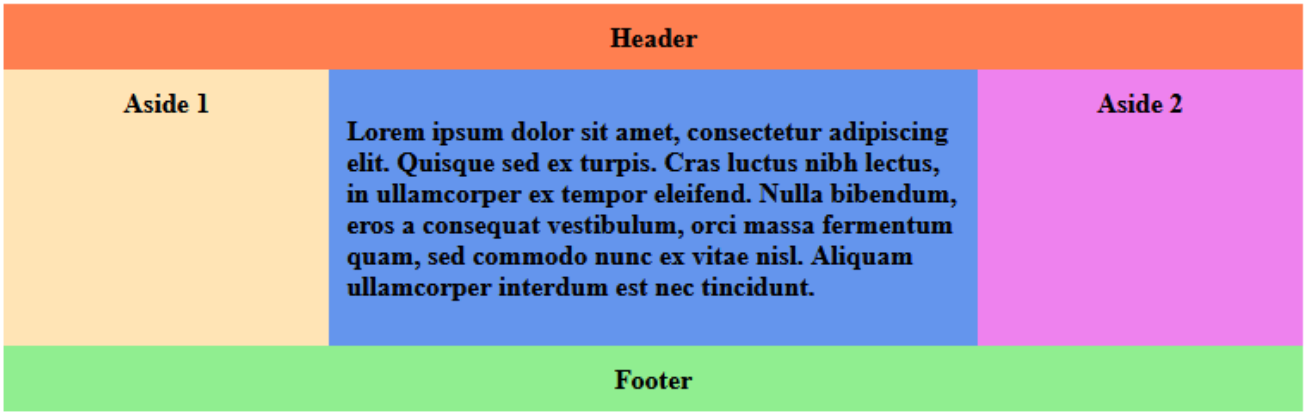
Джерело:
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_flexbox2




```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
  font-weight: bold;
  text-align: center;
}

.flex-container > * {
  padding: 10px;
```

```
  flex: 1 100%;
}

.main {
  text-align: left;
  background: cornflowerblue;
}

.header {background: coral;}
.footer {background: lightgreen;}
.aside {background: moccasin;}

@media all and (min-width: 768px) {
  .aside { flex: 1 auto; }
  .main        { flex: 3 0px; }
  .aside { order: 1; }
  .main        { order: 2; }
  .footer  { order: 4; }
}
</style>
</head>
<body>

<div class="flex-container">
  <header class="header">Header</header>
  <aside class="aside">Aside</aside>
  <article class="main">
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed ex turpis. Cras luctus nibh lectus, in ullamcorper ex tempor eleifend. Nulla bibendum, eros a consequat vestibulum, orci massa fermentum quam, sed commodo nunc ex vitae nisl. Aliquam ullamcorper interdum est nec tincidunt.</p>
  </article>
  <footer class="footer">Footer</footer>
</div>

</body>
</html>
```

ПРИКЛАД LAYOUT FLEXBOX 3



Джерело:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_flexbox3

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
  font-weight: bold;
  text-align: center;
}

.flex-container > * {
  padding: 10px;
  flex: 1 100%;
}

.main {
  text-align: left;
  background: cornflowerblue;
}

.header {background: coral;}
.footer {background: lightgreen;}
.aside1 {background: moccasin;}
.aside2 {background: violet;}

@media all and (min-width: 768px) {
  .aside { flex: 1 auto; }
```

```
  .main       { flex: 3 0px; }
  .aside1 { order: 1; }
  .main       { order: 2; }
  .aside2 { order: 3; }
  .footer  { order: 4; }
}
</style>
</head>
<body>

<div class="flex-container">
  <header class="header">Header</header>
  <aside class="aside aside1">Aside 1</aside>
  <article class="main">
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed ex turpis. Cras luctus nibh lectus, in ullamcorper ex tempor eleifend. Nulla bibendum, eros a consequat vestibulum, orci massa fermentum quam, sed commodo nunc ex vitae nisl. Aliquam ullamcorper interdum est nec tincidunt.</p>
  </article>
  <aside class="aside aside2">Aside 2</aside>
  <footer class="footer">Footer</footer>
</div>

</body>
</html>
```

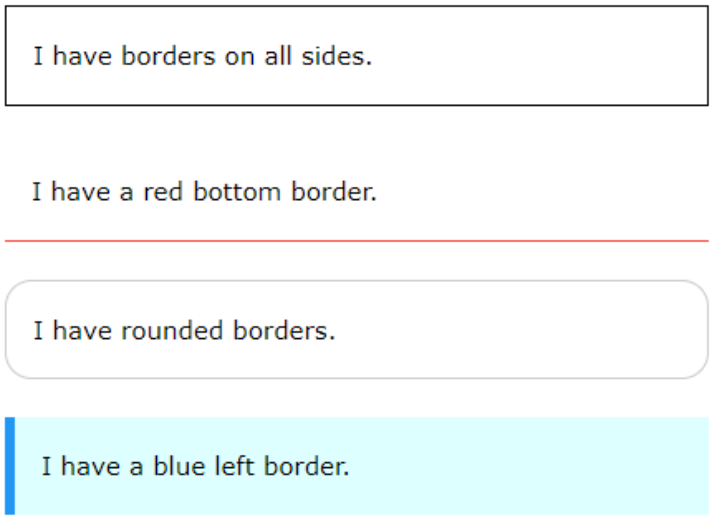
CSS 2

CSS редактор онлайн:

https://www.w3schools.com/css/tryit.asp?filename=trycss_default

РАМКА

Властивості border дозволяють вказати стиль, ширину та колір рамки елементу.



ТИП РАМКИ:

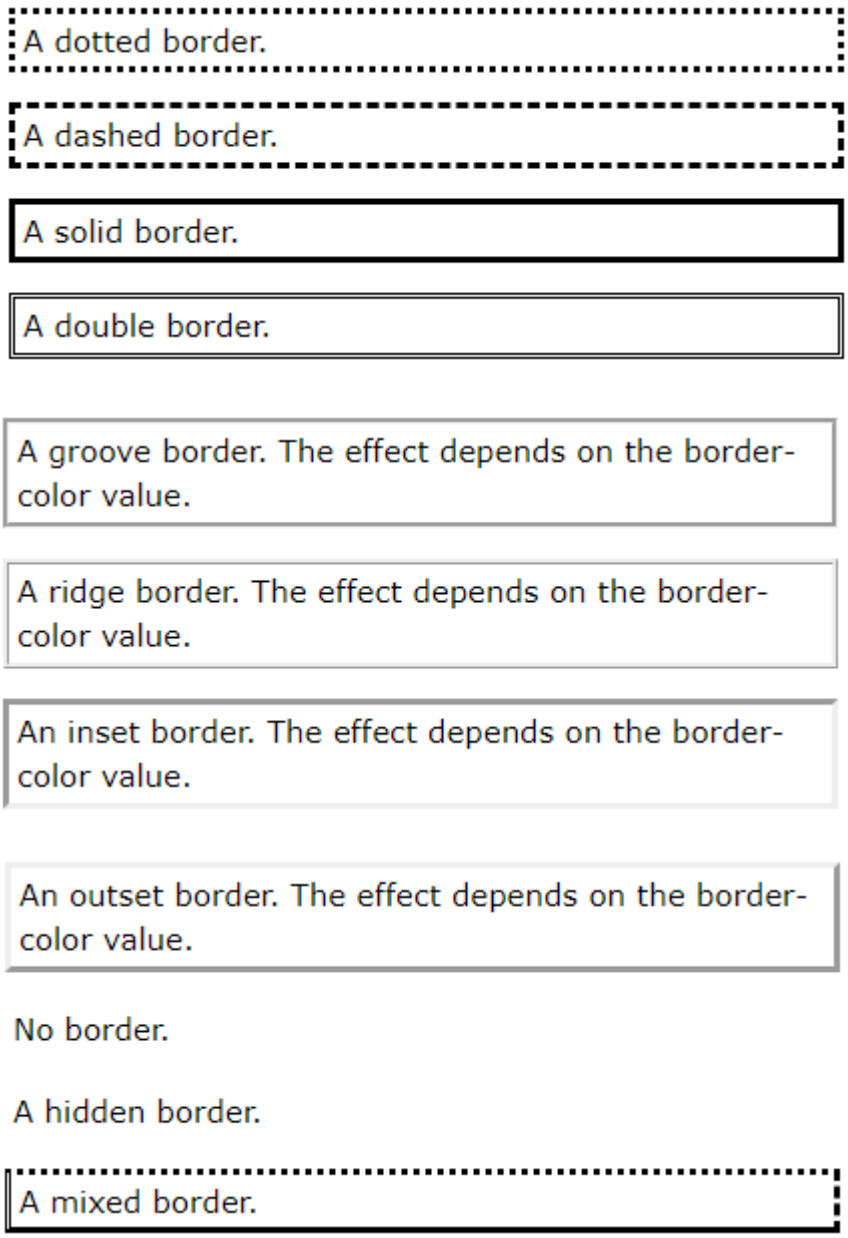
Властивість border-style визначає тип рамки для відображення. Значення:

- dotted – Визначає пунктирну рамку.
- dashed – Визначає пунктирну рамку.
- solid - Визначає суцільну рамку.
- double – Визначає подвійну рамку.
- groove – Тривимірна рифлена рамка. Ефект залежить від кольору рамки.
- ridge - Визначає 3D-ребристу рамку. Ефект залежить від кольору рамки.
- inset – Визначає рамку 3D-inset. Ефект залежить від кольору рамки.
- outset - Визначає рамку 3D outset. Ефект залежить від кольору рамки.
- none - без рамки.
- hidden – визначає приховану рамку.

Властивість border-style може мати від одного до чотирьох значень (для верхньої рамки, правої рамки, нижньої рамки та лівої рамки). Різни стилі рамки:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: суцільна;}
p.double {border-style: double; }
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border -style: none;}
```

```
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

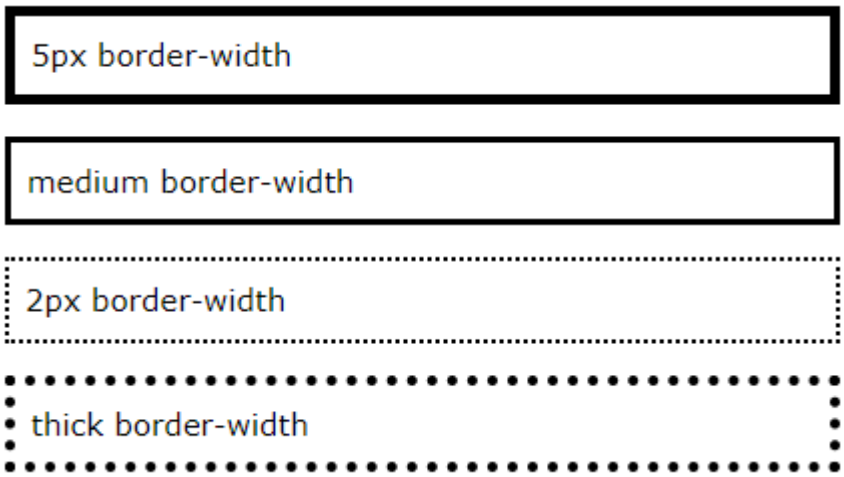


ШИРИНА РАМКИ:

Властивість border-width визначає ширину рамки. Ширину можна встановити як певний розмір (px, pt, cm, em) або за допомогою одного з попередньо визначених значень: thin (тонкий), medium (середній), або thick (товстий). Приклад:

```
p.one {
  border-style: solid;
  border-width: 5px;
}
p.two {
  border-style: solid;
  border-width: medium;
}
p.three {
  border-style: dotted;
```

```
  border-width: 2px;
}
p.four {
  border-style: dotted;
  border-width: thick;
}
```



СТОРОНИ РАМКИ:

Властивість border-width може мати від одного до чотирьох значень (для верхньої рамки, правої рамки, нижньої рамки та лівої рамки):

```
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px зверху та знизу, 20px з боків */
}
p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20 пікселів зверху та знизу, 5 пікселів з боків */
}
p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25 пікселів зверху, 10 пікселів праворуч, 4
пікселів внизу та 35 пікселів ліворуч */
}
```

КОЛІР МЕЖІ:

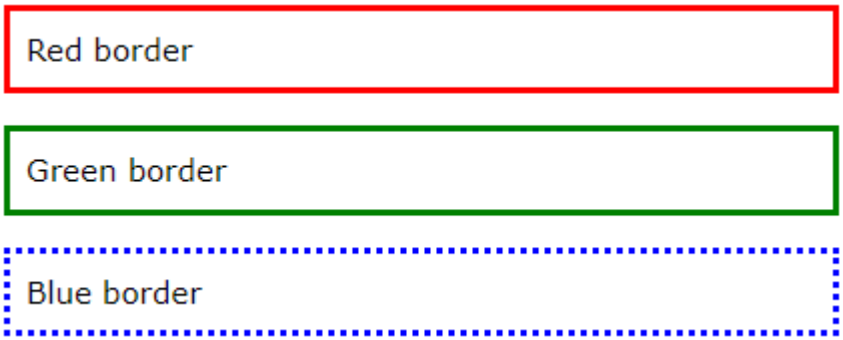
Властивість border-color використовується для встановлення кольору чотирьох рамок. Колір можна встановити за:

- Назвою - вкажіть назву кольору, наприклад «red».
- HEX - вкажіть значення HEX, наприклад "#ff0000".
- RGB – вкажіть значення RGB, наприклад «rgb(255,0,0)».
- HSL – укажіть значення HSL, наприклад «hsl(0, 100%, 50%)».
- transparent (прозорий).

Якщо border-color не встановлено, він успадковує колір елементу. Приклад:

```
p.one {
  border-style: solid;
```

```
  border-color: red;
}
p.two {
  border-style: solid;
  border-color: green;
}
p.three {
  border-style: dotted;
  border-color: blue;
}
```



СПЕЦИФІЧНІ КОЛЬОРИ СТОРІН:

Властивість border-color може мати від одного до чотирьох значень (для верхньої сторони, правої сторони, нижньої сторони та лівої сторони).

```
p.one {
  border-style: solid;
  border-color: red green blue yellow; /* червоний верх, зелений праворуч, синій низ і жовтий ліворуч */
}
```

СТОРОНИ РАМКИ:

У CSS є властивості для визначення кожної сторони рамки (верхньої, правої, нижньої та лівої):

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

Якщо властивість border-style має чотири значення:

border-style: dotted solid double dashed;

- верхня сторона - **dotted**.
- права сторона - **solid**.
- нижня сторона - **double**.

- ліва сторона - **dashed**.

Якщо властивість border-style має три значення: **border-style: dotted solid double;**

- верхня сторона - **dotted**.
- права та ліва сторони - **solid**.
- нижня сторона - **double**.

Якщо властивість border-style має два значення: **border-style: dotted solid;**

- верхня і нижня сторони - **dotted**.
- права та ліва сторони - **solid**.

Якщо властивість border-style має одне значення: **border-style: dotted;**

- чотири сторони рамки - **dotted**.

```
/* Чотири значення */
p {
  border-style: dotted solid double dashed;
}
/* Три значення */
p {
  border-style: dotted solid double;
}
/* Два значення */
p {
  border-style: dotted solid;
}
/* Одне значення */
p {
  border-style: dotted;
}
```

СКОРОЧЕННЯ ВЛАСТИВОСТІ:

Щоб скоротити код, також можна вказати всі окремі властивості в одній. Властивість border є скороченою властивістю для таких окремих властивостей:

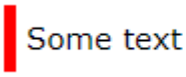
- border-width
- border-style (обов’язково)
- border-color

```
p {
  border: 5px solid red;
}
```

Some text

Можливо вказати окремі властивості рамки лише для однієї сторони. Ліва сторона:

```
p {
  border-left: 6px solid red;
}
```



Нижня сторона рамки:

```
p {
  border-bottom: 6px solid red;
}
```

Властивість border-radius використовується для додавання заокруглених рамок до елементу та для зміни форми елементу:

```
p {
  border: 2px solid red;
  border-radius: 5px;
}
```

ФОН

Властивості фону використовуються для додавання фонових ефектів для елементів.

- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background (вказати всі властивості разом)

КОЛІР ФОНУ:

Властивість background-color визначає колір фону елементу. Колір фону сторінки:

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

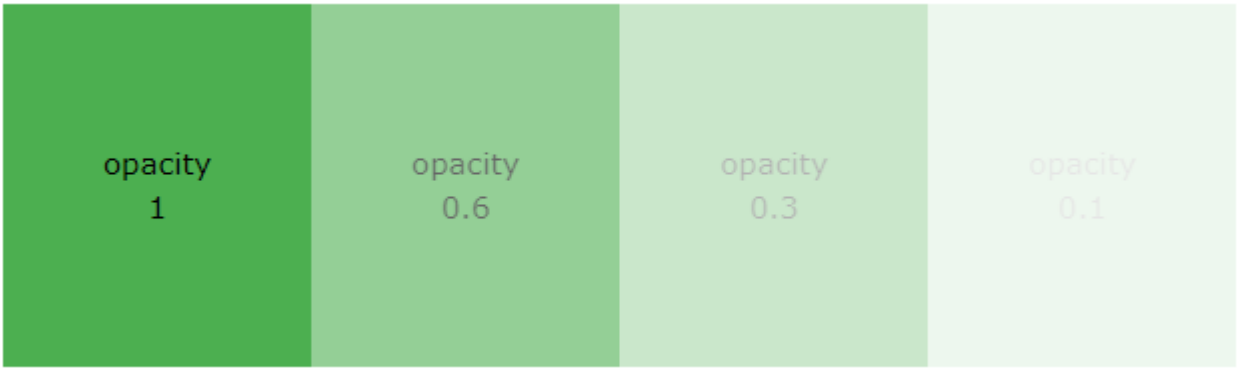
У CSS колір найчастіше вказується:

- правильною назвою кольору, як-от "red".
- шістнадцяткове значення, наприклад "#ff0000".
- значення RGB, наприклад "rgb(255,0,0)".

ПРОЗОРІСТЬ:

Властивість opacity визначає прозорість елементу. Вона може приймати значення від 0 до 1. Чим менше значення, тим прозоріше:

```
div {
  background-color: green;
  opacity: 0.3;
}
```



При використанні властивості opacity для додавання прозорості фону елементу всі його дочірні елементи успадковують однакову прозорість. Це може ускладнити читання тексту всередині повністю прозорого елементу.

ФОНОВЕ ЗОБРАЖЕННЯ:

Властивість background-image визначає зображення, яке буде використовуватися як фон елементу. За замовчуванням зображення повторюється, тому воно покриває весь елемент. Встановити фонове зображення для сторінки:

```
body {
  background-image: url("paper.gif");
}
```

Використовуйте зображення, які не заважають тексту. Фонове зображення також можна встановити для певних елементів, наприклад елементу <p>:

```
p {
  background-image: url("paper.gif");
}
```

ПОВТОР ФОНУ:

За замовчуванням властивість background-image повторює зображення як горизонтально, так і вертикально. Деякі зображення слід повторювати лише горизонтально або вертикально, інакше вони виглядатимуть дивно:

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

Щоб повторити зображення вертикально, установіть background-repeat: repeat-y;

БЕЗ ПОВТОРЕННЯ ФОНУ:

Показ фонового зображення лише один раз теж визначається властивістю background-repeat. Показувати фонове зображення лише один раз:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

ПОЛОЖЕННЯ ФОНОВОГО ЗОБРАЖЕННЯ:

Властивість background-position використовується для визначення положення фонового зображення. Розмістіть фонове зображення у верхньому правому куті:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

ПРОКРУЧУВАТИ АБО ФІКСУВАТИ ЗОБРАЖЕННЯ:

Властивість background-attachment визначає, чи має фонове зображення прокручуватись чи бути фіксованим (не прокручуватиметься разом із рештою сторінки). Вкажіть, що фонове зображення має бути фіксованим:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Вкажіть, що фонове зображення має прокручуватись разом із рештою сторінки:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

ДЛЯ СКОРОЧЕННЯ КОДА:

Щоб скоротити код, можна вказати всі властивості фону в одній. Замість написання:

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

Використовуйте background, щоб встановити властивості фону в одній декларації:

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

Під час використання скороченої властивості порядок значень властивостей такий:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

Неважливо, якщо одне зі значень властивостей відсутнє, коли інші вказані. Зауважте, що ми не використовуємо властивість background-attachment у наведених вище прикладах, оскільки вона не має значення.

ЗОВНІШНІ ВІДСТУПИ

Властивості margin використовуються для створення зовнішніх відступів навколо елементів. Існують властивості для встановлення зовнішніх відступів для кожної сторони елементу (верхньої, правої, нижньої та лівої).

ОКРЕМІ СТОРОНИ:

CSS має властивості для визначення відступів для кожної сторони елементу:

- margin-top
- margin-right
- margin-bottom
- margin-left

Усі властивості можуть мати такі значення:

- auto – браузер обчислює поле.
- length - визначає поле в пікселях, pt, см тощо.
- % - визначає поле в % від ширини батьківського елементу.
- inherit - значення має бути успадковане від батьківського елементу.

Можливі негативні значення. Встановіть різні поля для всіх сторін елементу <p>:

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

ВНУТРІШНІ ВІДСТУПИ

Властивості padding використовуються для створення внутрішніх відступів навколо вмісту елементу всередині будь-яких визначених меж. Існують властивості для встановлення відступів для кожної сторони (верхньої, правої, нижньої та лівої).

ІНДИВІДУАЛЬНІ СТОРОНИ:

CSS має властивості для визначення відступу для кожної сторони елементу:

- padding-top

- padding-right
- padding-bottom
- padding-left

Усі властивості padding можуть мати такі значення:

- length – визначає відступ у px, pt, см і т. д.
- % - вказує відступ у % від наявної ширини батьківського елементу.
- inherit - заповнення має бути успадковане від батьківського елементу.

Негативні значення не допускаються. Різні відступи для всіх сторін елементу:

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

ВСЕ РАЗОМ:

Щоб скоротити код, можна вказати всі властивості в одній. Властивість padding є скороченою властивістю для таких окремих властивостей:

- padding-top
- padding-right
- padding-bottom
- padding-left

Якщо властивість padding має чотири значення:

padding: 25px 50px 75px 100px;

- верхній відступ становить 25 пікселів.
- правий відступ становить 50 пікселів.
- нижній відступ становить 75 пікселів.
- лівий відступ становить 100 пікселів.

```
div {
  padding: 25px 50px 75px 100px;
}
```

Якщо властивість padding має три значення: **padding: 25px 50px 75px;**

- верхній відступ становить 25 пікселів.
- правий і лівий відступи становлять 50 пікселів.
- нижній відступ становить 75 пікселів.

```
div {
  padding: 25px 50px 75px;
}
```

Якщо властивість padding має два значення: **padding: 25px 50px;**

- верхній і нижній відступи становлять 25 пікселів.
- правий і лівий відступи становлять 50 пікселів.

```
div {
  padding: 25px 50px;
}
```

Якщо властивість padding має одне значення: **padding: 25px;**

- усі чотири відступи мають 25 пікселів.

```
div {
  padding: 25px;
}
```

ШИРИНА ЕЛЕМЕНТУ:

Властивість width визначає ширину області вмісту елементу. Область вмісту - це частина всередині заповнення, рамки та поля елементу.

Якщо елемент має задану ширину, відступ, доданий до цього елементу, буде додано до загальної ширини елементу. Це часто є небажаним результатом.

Тут елементу надається ширина 300 пікселів. Проте фактична ширина елементу становить 350 пікселів (300px + 25 пікселів відступу зліва + 25 пікселів відступу справа):

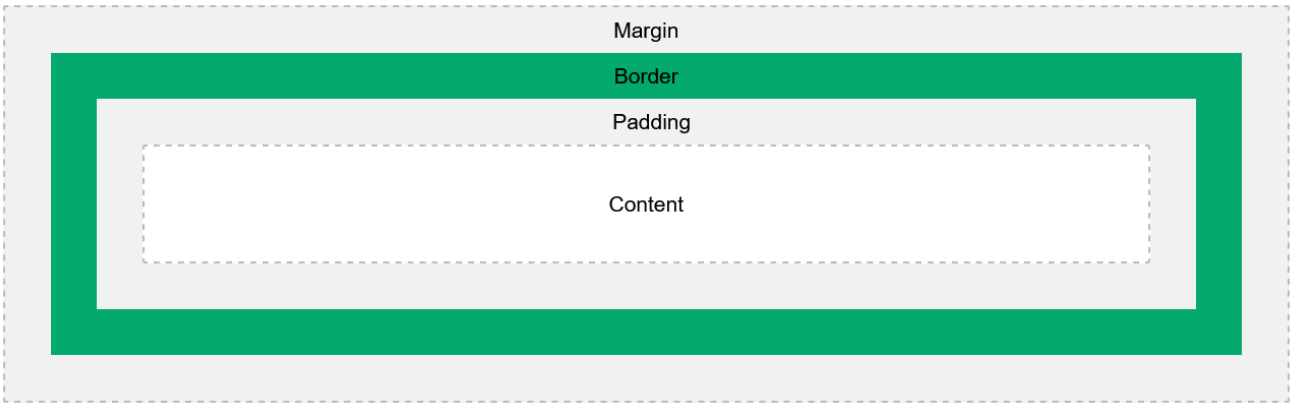
```
div {
  width: 300px;
  padding: 25px;
}
```

Щоб зберегти ширину на рівні 300 пікселів, незалежно від розміру відступу, ви можете використовувати властивість box-sizing. Це змушує елемент зберігати свою фактичну ширину. Якщо ви збільшите відступ, доступний простір вмісту зменшиться. Використовуйте box-sizing, щоб зберегти ширину на рівні 300 пікселів, незалежно від кількості відступів:

```
div {
  width: 300px;
  padding: 25px;
  box-sizing: border-box;
}
```

КОРОБКОВА МОДЕЛЬ:

Термін «коробкова модель» використовується, коли йдеться про дизайн і макет. Модель коробки CSS – це, по суті, коробка, яка обертається навколо кожного елемента HTML. Він складається з полів, рамок, відступів і фактичного вмісту. Модель коробки:



Пояснення різних частин:

- **Content** - вміст коробки, де з'являються текст і зображення.
- **Padding** - очищає область навколо вмісту. Відступ прозорий.
- **Border** - рамка, яка розташована навколо відступу та вмісту.
- **Margin** - очищає область за межами. Прозоре.

Коробкова модель дозволяє додавати рамки навколо елементів і визначати простір між елементами. Демонстрація коробкової моделі:

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

ШИРИНА І ВИСОТА ЕЛЕМЕНТУ:

Щоб правильно встановити ширину і висоту елементу в усіх браузерах, вам потрібно знати, як працює модель коробки. Коли ви встановлюєте властивості ширини та висоти елементу за допомогою CSS, ви просто встановлюєте ширину та висоту області **вмісту**. Щоб обчислити повний розмір елементу, ви також повинні додати внутрішні та зовнішні відступи та рамки.

Цей елемент матиме загальну ширину 350 пікселів:

```
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

Ось розрахунок: 320 пікселів (ширина) + 20 пікселів (лівий + правий padding) + 10 пікселів (ліва + права border) + 0 пікселів (ліва + права margin) **= 350 пікселів.**

Загальна ширина елементу = ширина + лівий padding + правий padding + ліва border + права border + ліве margin + праве margin.

Загальна висота елементу = висота + верхній padding + нижній padding + верхня border + нижня border + верхня margin + нижня margin.

ОДИНИЦІ ВИМІРЮВАННЯ ДОВЖИНИ

CSS має кілька різних одиниць для вираження довжини. Багато властивостей CSS приймають значення «довжини», такі як ширина, поле, відступ, розмір шрифту тощо. **Довжина** - це число, за яким слідує одиниця вимірювання довжини, наприклад 10px, 2em тощо.

```
h1 {
  font-size: 60px;
}
p {
  font-size: 25px;
  line-height: 50px;
}
```

Між числом і одиницею не може бути пропусків. Проте, якщо значення дорівнює 0, одиницю можна опустити. Для деяких властивостей CSS дозволена від'ємна довжина. Є два типи одиниць вимірювання довжини: **абсолютні** та **відносні**.

АБСОЛЮТНА ДОВЖИНА:

Одиниці абсолютної довжини є фіксованими. Довжина, виражена в абсолютних одиницях, відображатиметься як саме такий розмір. Одиниці абсолютної довжини не рекомендуються для використання на екрані, оскільки розміри екрана бувають різні. Однак їх можна використовувати, якщо відомий носій виводу, наприклад для друку.

Одиниця	Опис
cm	сантиметри
mm	міліметри
in	дюйми (1 дюйм = 96 пікселів = 2.54 сантиметра)
px	пікселі (1 піксель = 1/96 від 1 дюйма)
pt	1 пункт = 1/72 від 1 дюйма
pc	1 піса = 12 пунктів

Пікселі (px) завжди залежать від пристрою перегляду. Для пристроїв із низькою роздільною здатністю 1 піксель - це один піксель (точка) дисплея. Для принтерів і екранів з високою роздільною здатністю 1 піксель означає кілька пікселів пристрою.

ВІДНОСНІ ДОВЖИНИ:

Одиниці відносної довжини визначають довжину відносно іншої властивості довжини. Одиниці відносної довжини краще масштабуються між різними середовищами візуалізації.

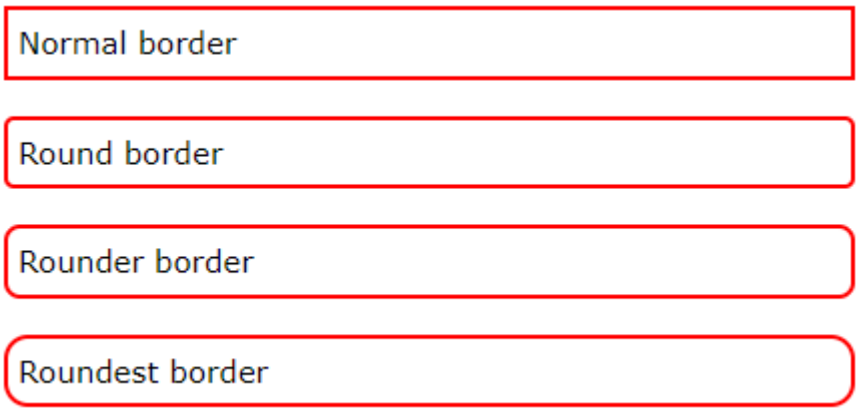
Одиниця	Опис
em	Відносно розміру шрифту елементу (2em означає 2-кратний розмір поточного шрифту)

ex	Відносно x-висоти поточного шрифту (використовується рідко)
ch	Відносно ширини символу «0»
rem	Відносно розміру шрифту кореневого елементу
vw	Відносно 1% ширини вікна браузера. Якщо вікно браузера має ширину 50 см, 1vw = 0.5 см.
vh	Відносно 1% висоти вікна браузера
vmin	Відносно 1% найменшої сторони вікна браузера
vmax	Відносно 1% найбільшої сторони вікна браузера
%	Відносно батьківського елементу

Одиниці em і rem є практичними у створенні ідеально масштабованого макета.

РАДІУС РАМКИ

Властивість CSS border-radius визначає радіус кутів рамки та границі елементу. Ця властивість дозволяє додавати заокруглені кути до елементів.




```
#rcorners1 {
  border-radius: 25px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners2 {
  border-radius: 25px;
  border: 2px solid #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners3 {
  border-radius: 25px;
  background: url(paper.gif);
  background-position: left top;
```

```
  background-repeat: repeat;
  padding: 20px;
  width: 200px;
  height: 150px;
}
```

Властивість border-radius насправді є скороченою властивістю border-top-left-radius, border-top-right-radius, border-bottom-right-radius і border-bottom-left-radius.

Властивість border-radius може мати від одного до чотирьох значень. Правила:

Чотири значення - border-radius: 15px 50px 30px 5px;

Перше значення стосується верхнього лівого кута, друге значення стосується верхнього правого кута, третє значення стосується нижнього правого кута, а четверте значення застосовується до нижнього лівого кута:

Три значення - border-radius: 15px 50px 30px;

Перше значення стосується верхнього лівого кута, друге значення стосується верхнього правого та нижнього лівого кутів, а третє значення стосується нижнього правого кута:

Два значення - border-radius: 15px 50px;

Перше значення стосується верхнього лівого та нижнього правого кутів, а друге значення стосується верхнього правого та нижнього лівого кутів:

Одне значення - border-radius: 15px; (стосується всіх чотирьох кутів):

```
#rcorners1 {
  border-radius: 15px 50px 30px 5px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners2 {
  border-radius: 15px 50px 30px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners3 {
  border-radius: 15px 50px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners4 {
  border-radius: 15px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
```

Ви також можете створити еліптичні кути:

```
#rcorners1 {
  border-radius: 50px / 15px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
```

```
  height: 150px;
}
#rcorners2 {
  border-radius: 15px / 50px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners3 {
  border-radius: 50%;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
```

Elliptical border - border-radius: 50px / 15px:

Elliptical border - border-radius: 15px / 50px:

Ellipse border - border-radius: 50%:

ЛІНІЙНІ ГРАДІЄНТИ

Градієнти дозволяють відображати плавні переходи між двома або більше заданими кольорами. Типи градієнтів:

- Лінійні градієнти (іде вниз/вгору/ліворуч/праворуч/діагонально).
- Радіальні градієнти (визначені центром).
- Конічні градієнти (обертаються навколо центральної точки).

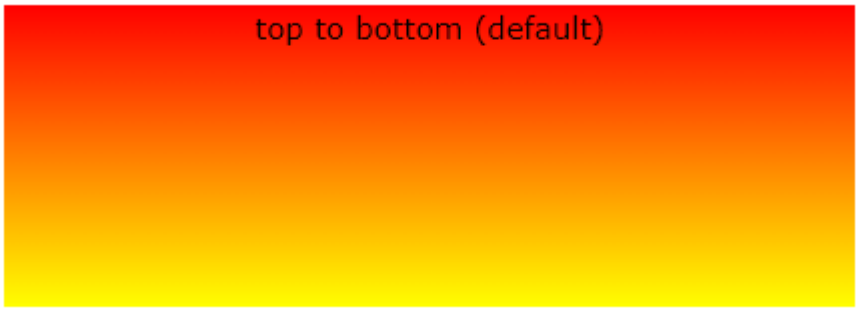
Щоб створити лінійний градієнт, необхідно визначити принаймні дві кольорові зупинки. Кольорові зупинки – це кольори, між якими ви хочете зробити плавні переходи. Ви можете встановити початкову точку та напрямок (або кут) разом із ефектом градієнта. Синтаксис:

background-image: linear-gradient(direction, color-stop1, color-stop2, ...);

НАПРЯМ – ВВЕРХ ДОНИЗУ (ЗА ЗАМОВЧАННЯМ):

У наступному прикладі показано лінійний градієнт, який починається зверху. Він починається з червоного, переходить у жовтий:

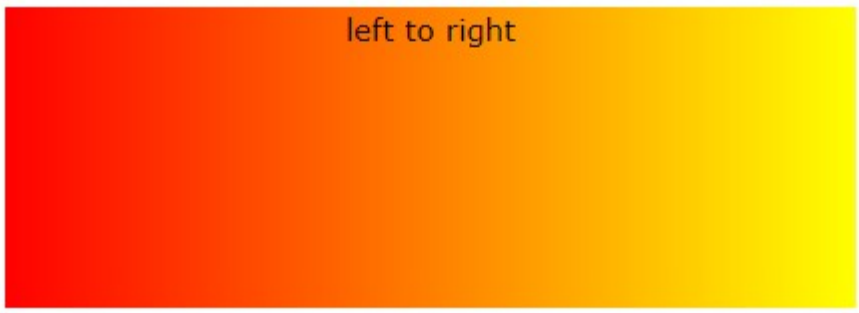
```
#grad {
  background-image: linear-gradient(red, yellow);
}
```



НАПРЯМ – ЗЛІВА НАПРАВО:

У наступному прикладі показано лінійний градієнт, який починається зліва. Він починається з червоного, переходить у жовтий:

```
#grad {
  background-image: linear-gradient(to right, red , yellow);
}
```



НАПРЯМ – ДІАГОНАЛЬ:

Ви можете створити градієнт по діагоналі, вказавши горизонтальну та вертикальну початкові позиції. У наступному прикладі показано лінійний градієнт, який починається у верхньому лівому куті (та йде вниз і праворуч). Він починається з червоного, переходить у жовтий:

```
#grad {
  background-image: linear-gradient(to bottom right, red, yellow);
}
```

ВИКОРИСТАННЯ КУТІВ:

Якщо вам потрібен більший контроль над напрямком градієнта, ви можете визначити кут замість попередньо визначених напрямків.

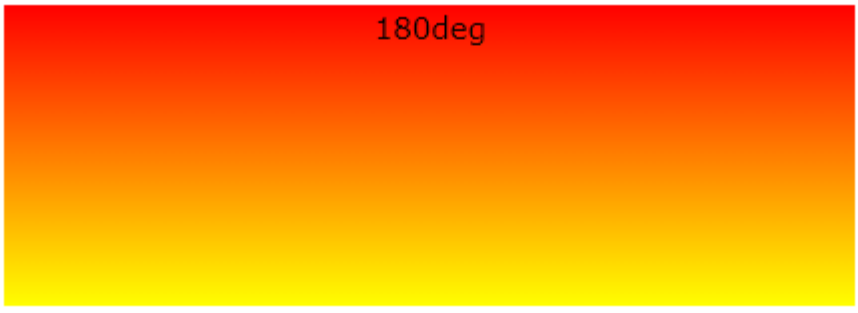
- Значення 0deg еквівалентно to top (доверху).
- Значення 90deg еквівалентно to right (праворуч).
- Значення 180deg еквівалентно to bottom (донизу).

Синтаксис:

background-image: linear-gradient(angle, color-stop1, color-stop2);

У наступному прикладі показано, як використовувати кути на лінійних градієнтах:

```
#grad {
  background-image: linear-gradient(180deg, red, yellow);
}
```



КІЛЬКА КОЛІРНИХ ПОЗИЦІЙ:

Лінійний градієнт (зверху вниз) із кількома кольоровими межами:

```
#grad {
  background-image: linear-gradient(red, yellow, green);
}
```

Лінійний градієнт (зліва направо) з кольорами веселки:

```
#grad {
  background-image: linear-gradient(to right, red, orange, yellow, green, blue,
indigo, violet);
}
```

ПРОЗОРІСТЬ:

Градієнти підтримують прозорість, яку можна використовувати для створення ефектів вицвітання. Додати прозорість можна, використовуючи функцію rgba(), щоб визначити колірні точки. Останній параметр у функції - значення від 0 до 1, яке визначає прозорість кольору: 0 - повна прозорість, 1 - повний колір без прозорості.

У наступному прикладі показано лінійний градієнт, який починається зліва. Він починається повністю прозорим, переходячи до повного червоного кольору:

```
#grad {
  background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
}
```

ФУНКЦІЯ ПОВТОРЕННЯ ГРАДІЄНТА:

repeating-linear-gradient() використовується для повторення лінійних градієнтів.

```
#grad {
  background-image: repeating-linear-gradient(red, yellow 10%, green 20%);
}
```

РАДІАЛЬНІ ГРАДІЄНТИ

Градієнти дозволяють відображати плавні переходи між двома або більше вказаними кольорами. Радіальний градієнт визначається його центром. Щоб створити радіальний градієнт, треба визначити принаймні дві кольорові точки. Синтаксис:

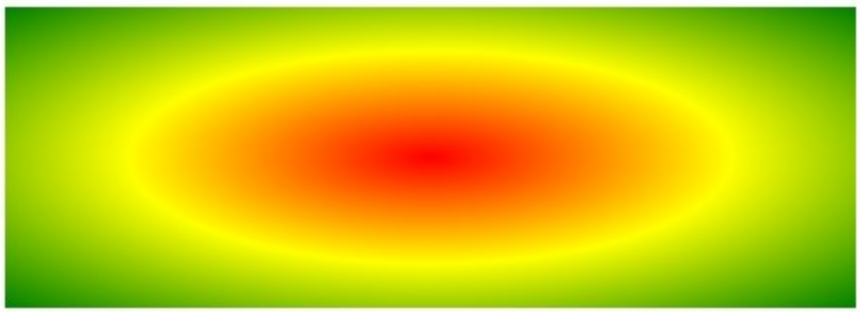
background-image: radial-gradient(розмір форми **at** позиція, початковий колір, ..., останній колір);

За замовчуванням форма – еліпс, розмір – найдальший кут, а позиція – центр.

РІВНОМІРНО РОЗДІЛЕНІ КОЛЬОРОВІ ПОЗИЦІЇ:

Радіальний градієнт з рівномірно розташованими кольоровими межами:

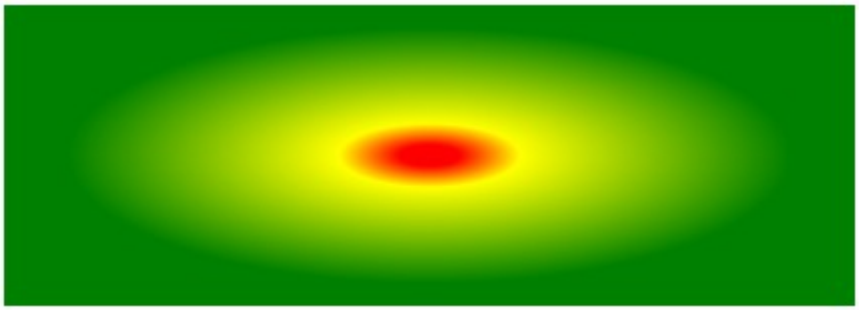
```
#grad {
  background-image: radial-gradient(red, yellow, green);
}
```



РІЗНО РОЗМІЩЕНІ КОЛЬОРОВІ ПОЗИЦІЇ:

Радіальний градієнт із різнорознесеними кольоровими позначками:

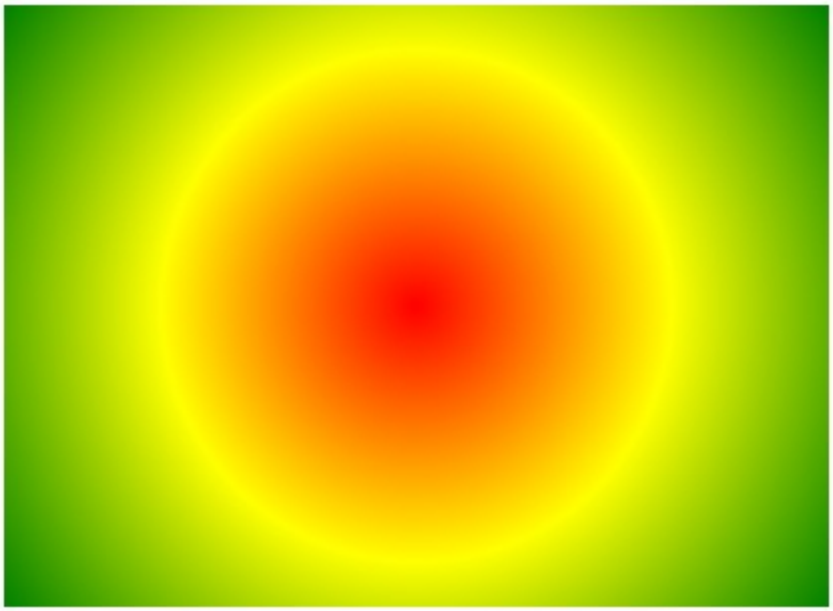
```
#grad {
  background-image: radial-gradient(red 5%, yellow 15%, green 60%);
}
```



ВСТАНОВИТИ ФОРМУ:

Параметр shape визначає форму. Він може приймати значення кола або еліпса. Значенням за замовчуванням є еліпс. Радіальний градієнт із формою кола:

```
#grad {
  background-image: radial-gradient(circle, red, yellow, green);
}
```



КОНІЧНІ ГРАДІЄНТИ

Градієнти дозволяють відображати плавні переходи між двома або більше вказаними кольорами. Конічний градієнт — це градієнт з переходами кольорів. В конічних градієнтах точки зупинки кольору розташовуються навколо градієнтної дуги, вздовж довжини кола. Щоб створити конічний градієнт, необхідно визначити принаймні два кольори. Синтаксис:

background-image: conic-gradient([from angle] [**at** позиція,] колір [градус], колір [градус], ...);

За замовчуванням кут дорівнює 0 градусів, а позиція – центр. Якщо градус не вказано, кольори розподілятимуться рівномірно навколо центральної точки.

У наступному прикладі показано конічний градієнт із трьома кольорами:

```
#grad {
  background-image: conic-gradient(red, yellow, green);
}
```

Ще приклади:

```
/* Конічний градієнт, повернутий на 45 градусів,
    починаючи з синього і закінчуючи червоним
*/
conic-gradient(from 45deg, blue, red);

/* Синювато-фіолетовий квадрат: градієнт переходить від синього до червоного,
    але видно лише нижній правий квадрант, тому що
    центр конічного градієнта знаходиться у верхньому лівому куті
*/
conic-gradient(from 90deg at 0 0, blue, red);

/* Колірне коло */
background: conic-gradient(
    hsl(360, 100%, 50%),
    hsl(315, 100%, 50%),
    hsl(270, 100%, 50%),
    hsl(225, 100%, 50%),
    hsl(180, 100%, 50%),
    hsl(135, 100%, 50%),
    hsl(90, 100%, 50%),
    hsl(45, 100%, 50%),
    hsl(0, 100%, 50%)
);
```

СЕЛЕКТОРИ

У CSS селектори - це шаблони, які використовуються для вибору елементів, до яких потрібно додати стиль.

Селектор	**Приклад**	**Опис**
.class	.intro	Вибирає всі елементи за допомогою class="intro"
.class1.class2	.name1.name2	Вибирає всі елементи з *name1* і *name2*, встановленими в атрибуті class

.class1 .class2	.name1 .name2	Вибирає всі елементи з class=*name2*, які є всередині елементу з class=*name1*
#id	#firstname	Вибирає елемент з id="firstname"
*	*	Вибирає всі елементи
element	p	Вибирає всі <p> елементи
element.class	p.intro	Вибирає всі <p> елементи з class="intro"
element,element	div, p	і всі елементи <p>
element element	div p	Вибирає всі елементи <p> всередині елементів
element>element	div > p	Вибирає всі елементи <p>, де батьківським є елемент element
element+element	div + p	Вибирає перший елемент <p>, який розміщено одразу після елементів
element1~element2	p ~ ul	Вибирає кожен елемент , якому передує елемент <p>
[*attribute*]	[target]	Вибирає всі елементи з target атрибутом
[*attribute=value*]	[target=_blank]	Вибирає всі елементи з target="_blank"
[*attribute~=value*]	[title~=flower]	Вибирає всі елементи з атрибутом title, що містить слово "flower"
[*attribute*|=*value*]	[lang|=en]	Вибирає всі елементи зі значенням атрибута lang, що дорівнює "en" або починається з "en-"
[*attribute^=value*]	a[href^="https"]	Вибирає кожен елемент <a>, значення атрибута href якого починається з "https"
[*attribute$=value*]	a[href$=".pdf"]	Вибирає кожен елемент <a>, значення атрибута href якого закінчується на ".pdf"
[*attribute**=*value*]	a[href*="google"]	Вибирає кожен елемент <a>, значення атрибута href якого містить підрядок "google"
:active	a:active	Вибирає активне посилання
::after	p::after	Вставити щось після вмісту кожного елементу <p>
::before	p::before	Вставити щось перед вмістом кожного елементу <p>
:checked	input:checked	Вибирає кожен позначений елемент <input>
:default	input:default	Вибирає елемент <input> за

		замовчуванням
:disabled	input:disabled	Вибирає кожен вимкнений елемент <input>
:empty	p:empty	Вибирає кожен елемент <p>, який не має дочірніх елементів
:enabled	input:enabled	Вибирає кожен увімкнений елемент <input>
:first-child	p:first-child	Вибирає кожен елемент <p>, який є першим дочірнім елементом свого батька
::first-letter	p::first-letter	Вибирає першу літеру кожного елементу <p>
::first-line	p::first-line	Вибирає перший рядок кожного елементу <p>
:first-of-type	p:first-of-type	Вибирає кожен елемент <p>, який є першим елементом <p> свого батька
:focus	input:focus	Вибирає елемент введення, який має фокус
:fullscreen	:fullscreen	Вибирає елемент, який знаходиться в повноекранному режимі
:hover	a:hover	Вибирає посилання при наведенні миші
:in-range	input:in-range	Вибирає вхідні елементи зі значенням у вказаному діапазоні
:indeterminate	input:indeterminate	Вибирає вхідні елементи, які перебувають у невизначеному стані
:invalid	input:invalid	Вибирає всі елементи введення з недійсним значенням
:lang(*language*)	p:lang(it)	Вибирає кожен елемент <p> з атрибутом lang, що дорівнює "it" (італійський)
:last-child	p:last-child	Вибирає кожен елемент <p>, який є останнім дочірнім елементом його батьківського
:last-of-type	p:last-of-type	Вибирає кожен елемент <p>, який є останнім елементом <p> свого батька
:link	a:link	Вибирає всі невідвідані посилання
::marker	::marker	Вибирає маркери елементів списку
:not(*selector*)	:not(p)	Вибирає кожен елемент, який не є елементом <p>
:nth-child(*n*)	p:nth-child(2)	Вибирає кожен елемент <p>, який є другим дочірнім елементом свого батька

:nth-last-child(*n*)	p:nth-last-child(2)	Вибирає кожен елемент <p>, який є другою дитиною свого батька, рахуючи від останнього дочірнього
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Вибирає кожен елемент <p>, який є другим елементом <p> свого батька, рахуючи від останнього дочірнього
:nth-of-type(*n*)	p:nth-of-type(2)	Вибирає кожен елемент <p>, який є другим елементом <p> свого батька
:only-of-type	p:only-of-type	Вибирає кожен елемент <p>, який є єдиним елементом <p> його батьківського
:only-child	p:only-child	Вибирає кожен елемент <p>, який є єдиним дочірнім елементом його батьківського
:optional	input:optional	Вибирає вхідні елементи без "required" атрибута
:out-of-range	input:out-of-range	Вибирає вхідні елементи зі значенням поза вказаним діапазоном
::placeholder	input::placeholder	Вибирає вхідні елементи з указаним атрибутом "placeholder"
:read-only	input:read-only	Вибирає елементи введення з указаним атрибутом "readonly"
:read-write	input:read-write	Вибирає елементи вводу, в яких атрибут "readonly" НЕ вказано
:required	input:required	Вибирає елементи введення з указаним "required" атрибутом
:root	:root	Вибирає кореневий елемент документа
::selection	::selection	Вибирає частину елементу, вибраного користувачем
:target	#news:target	Вибирає поточний активний елемент #news (клацнувши на URL-адресі, що містить це ім'я)
:valid	input:valid	Вибирає всі елементи введення з дійсним значенням
:visited	a:visited	Вибирає всі відвідані посилання

HTML CANVAS

ЩО ТАКЕ CANVAS?

Елемент HTML <canvas> використовується для малювання графіки за допомогою JavaScript. Елемент <canvas> є лише контейнером для графіки. Ви повинні використовувати JavaScript, щоб фактично намалювати графіку. Canvas має кілька методів малювання контурів, прямокутників, кіл, тексту та додавання зображень.

Internet Explorer 9, Firefox, Opera, Chrome і Safari підтримують <canvas> і його властивості та методи. IE 8 і попередні версії не підтримують <canvas>.

КРОК 1: РОЗМІТКА HTML

Полотно канваса - це прямокутна область на сторінці HTML. За замовчуванням полотно не має рамок і вмісту. Розмітка виглядає так:

```
<canvas id="myCanvas" width="800" height="600"></canvas>
```

Завжди вказуйте атрибут id (на який потрібно посилатися в сценарії), а також ширину та висоту для визначення розміру полотна. Щоб додати рамку, використовуйте style.

Ось приклад базового порожнього полотна:

```
<canvas id="myCanvas" width="800" height="600"
style="border:1px solid #000000;">
</canvas>
```

КРОК 2: ЗНАЙДІТЬ ЕЛЕМЕНТ CANVAS

Перш за все, ви повинні знайти елемент <canvas>. Це робиться за допомогою методу HTML DOM getElementById().

```
<script>
var canvas = document.getElementById("myCanvas");
</script>
```

КРОК 2: СТВОРІТЬ ОБ'ЄКТ МАЛЮВАННЯ

Вам потрібен об'єкт малювання для полотна. Метод getContext() повертає об'єкт HTML із властивостями та методами для малювання:

```
var ctx = canvas.getContext("2d");
```

КРОК 3: МАЛЮЙТЕ НА ПОЛОТНІ

Нарешті ви можете малювати на полотні. Встановіть стиль заливки об'єкта малювання на червоний колір:

```
ctx.fillStyle = "#FF0000";
```

Властивість fillStyle може бути кольором CSS, градієнтом або візерунком. За замовчуванням fillStyle - чорний. Метод fillRect(*x,y,width,height*) малює на полотні прямокутник, заповнений стилем заливки:

```
ctx.fillRect(0, 0, 150, 75);
```

КОД РАЗОМ:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Canvas</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  </head>
  <body>
      <canvas id="myCanvas" width="800" height="600" style="border:1px solid
#000000;">
      </canvas>
      <script>
      var canvas = document.getElementById("myCanvas");
      var ctx = canvas.getContext("2d");
      ctx.fillStyle = "#FF0000";
      ctx.fillRect(0, 0, 150, 75);
      </script>
  </body>
</html>
```

КООРДИНАТИ ПОЛОТНА:

Полотно HTML - двовимірна сітка. Верхній лівий кут полотна має координати (0,0). Метод fillRect(0,0,150,75) означає: почніть з верхнього лівого кута (0,0) і намалюйте прямокутник 150x75 пікселів.

МАЛЮЄМО ЛІНІЮ

Щоб намалювати пряму лінію на полотні, скористайтеся такими методами:

- moveTo(*x,y*) – визначає початкову точку лінії.
- lineTo(*x, y*) – визначає кінцеву точку лінії.

Щоб фактично намалювати лінію, ви повинні використати один із методів «чорнила», наприклад stroke().

Визначте початкову точку в позиції (0,0) і кінцеву точку в позиції (200,100). Потім використовуйте метод stroke(), щоб фактично намалювати лінію:

```
ctx.moveTo(0,0);
ctx.lineTo(200, 100);
ctx.stroke();
```

МАЛЮЄМО КОЛО

Щоб намалювати коло на полотні, скористайтеся такими методами:

- beginPath() - починає шлях.
- arc(x,y,r,startangle,endangle) - створює дугу/криву.

Щоб створити коло за допомогою arc(): встановіть початковий кут на 0 і кінцевий кут на 2*Math.PI. Параметри x і y визначають координати x і y центру кола. Параметр r визначає радіус кола.

Визначте коло за допомогою методу arc(). Потім використовуйте метод stroke(), щоб фактично намалювати коло:

```
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

ЗАПОВНЕННЯ ГРАДІЄНТОМ

Градієнти можна використовувати для заповнення прямокутників, кіл, ліній, тексту тощо. Фігури на полотні не обмежуються суцільними кольорами. Існує два різних типи градієнтів:

- createLinearGradient(*x,y,x1,y1*) – створює лінійний градієнт.
- createRadialGradient(*x,y,r,x1,y1,r1*) – створює радіальний/круговий градієнт.

Коли ми маємо об’єкт градієнта, ми повинні додати дві або більше кольорових зупинок. Метод addColorStop() визначає колірні точки та їхнє положення вздовж градієнта. Позиції градієнта можуть бути будь-якими від 0 до 1.

Щоб використовувати градієнт, установіть властивість fillStyle або strokeStyle для градієнта, а потім намалюйте фігуру (прямокутник, текст або лінію).

Використання лінійного градієнта:

```
// Створення градієнта
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Заливка градієнтом
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
```

Використання радіального/кругового градієнта:

```
// Створення градієнта
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Заливка градієнтом
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
```

МАЛЮВАННЯ ТЕКСТУ

Щоб намалювати текст на полотні, найважливішими властивостями та методами є:

- font - визначає властивості шрифту для тексту.
- fillText(*text,x,y*) - малює заповнений текст на полотні.
- strokeText(*text, x,y*) - малює текст на полотні без заливки.

За допомогою fillText():

```
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
```

Використовуючи strokeText():

```
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
```

МАЛЮВАННЯ ЗОБРАЖЕННЯ

Щоб намалювати зображення на полотні, скористайтеся таким методом:

drawImage(*image,x,y*)

```
<img src="image1.png" id="image1" alt="">
<script>
window.onload = function() {
  var canvas = document.getElementById("myCanvas");
  var ctx = canvas.getContext("2d");
  var img = document.getElementById("image1");
  ctx.drawImage(img, 10, 10);
};
</script>
```

ЗМІСТ