Курс з CSS: Стилізація Вебу

CSS (Cascading Style Sheets) — це мова стилів, яка використовується для опису зовнішнього вигляду та форматування веб-сторінок, написаних за допомогою HTML. Завдяки CSS ти можеш контролювати кольори, шрифти, розташування елементів та багато іншого.

Розділ 1: Вступ до CSS

Що таке CSS і навіщо він потрібен?

* CSS — це інструмент, що дозволяє відділити структуру документа (HTML) від його візуального представлення.
* Переваги CSS:
  * Ефективність: Зміни застосовуються до багатьох сторінок одночасно.
  * Контроль: Повний контроль над дизайном.
  * Швидкість завантаження: Зменшується обсяг коду HTML.

Як підключити CSS до HTML?

Є три основні способи:

* Зовнішній CSS (External CSS): Найкращий і найпоширеніший спосіб. Стилі зберігаються в окремому файлі .css.

```
  <!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Мій Сайт</title>
    <link rel="stylesheet" href="styles.css"> </head>
<body>
    <h1>Привіт, світ!</h1>
</body>
</html>

  /* styles.css */
h1 {
    color: blue;
}
```

* Внутрішній CSS (Internal CSS): Стилі пишуться всередині тегу <style> у секції <head> HTML-документа. Використовується для однієї сторінки, якщо її стилі унікальні.

```
  <!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Мій Сайт</title>
    <style>
        h1 {
            color: green;
        }
    </style>
</head>
```

```
<body>
    <h1>Привіт, світ!</h1>
</body>
</html>
```

* Вбудований CSS (Inline CSS): Стилі застосовуються безпосередньо до HTML-елемента за допомогою атрибута style. Використовується рідко, зазвичай для швидкого тестування або невеликих, унікальних стилів.

```
  <!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Мій Сайт</title>
</head>
<body>
    <h1 style="color: red;">Привіт, світ!</h1>
</body>
</html>
```

Пріоритет стилів: Вбудовані стилі мають найвищий пріоритет, потім внутрішні, а потім зовнішні.

Синтаксис CSS

Правило CSS складається з селектора та блоку оголошень.

* Селектор: Вказує, до яких HTML-елементів будуть застосовані стилі.
* Блок оголошень: Містить одне або кілька оголошень, розділених крапкою з комою.
* Оголошення: Пара властивість: значення;

```
  селектор {
    властивість: значення;
    властивість: значення;
}
```

Приклад:

```
  p { /* Селектор: всі параграфи */
    color: #333; /* Властивість: color, Значення: #333 */
    font-size: 16px; /* Властивість: font-size, Значення: 16px */
    text-align: center;
}
```

Розділ 2: Селектори CSS

Селектори дозволяють точно вибрати елементи для стилізації.

Основні типи селекторів

* Селектор елемента (Type Selector): Вибирає всі екземпляри заданого HTML-елемента.

```
  h1 {
    color: navy;
}
p {
```

```
    line-height: 1.5;
}
```

* Селектор класу (Class Selector): Вибирає елементи з певним атрибутом class. Класи починаються з крапки (.).

```
  <p class="intro">Це вступний текст.</p>
<p class="warning">Це попередження.</p>

  .intro {
    font-style: italic;
}
.warning {
    color: red;
    font-weight: bold;
}
```

* Селектор ідентифікатора (ID Selector): Вибирає унікальний елемент з певним атрибутом id. ID починаються з хешу (#). ID має бути унікальним на сторінці.

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

  #header {
    background-color: lightblue;
    padding: 20px;
}
```

* Універсальний селектор (Universal Selector): Вибирає всі елементи на сторінці. Позначається зірочкою (*).

```
  * {
    box-sizing: border-box; /* Дуже корисна властивість для макетування */
    margin: 0;
    padding: 0;
}
```

Комбінатори селекторів

* Селектор нащадка (Descendant Selector): Вибирає елементи, які є нащадками іншого елемента.

```
  div p { /* Вибирає всі <p>, що знаходяться всередині <div> */
    margin-bottom: 10px;
}
```

* Селектор дочірнього елемента (Child Selector): Вибирає елементи, які є безпосередніми дочірніми елементами іншого елемента.

```
  ul > li { /* Вибирає всі <li>, які є прямими дітьми <ul> */
    list-style-type: square;
}
```

* Селектор суміжного елемента (Adjacent Sibling Selector): Вибирає елемент, який йде одразу за іншим елементом, і вони мають одного батька.

```
  h1 + p { /* Вибирає <p>, що йде одразу за <h1> */
    margin-top: 30px;
}
```

* Селектор загального сусіда (General Sibling Selector): Вибирає всі елементи, які йдуть за вказаним елементом та мають одного батька.

```
  h1 ~ p { /* Вибирає всі <p>, що йдуть за <h1> */
    background-color: #eee;
}
```

Селектори атрибутів

Вибирають елементи на основі їхніх атрибутів.

* [target]
* [target="_blank"]
* [title~="flower"] (містить слово)
* [lang|="en"] (починається з "en" або "en-")
* a[href^="https"] (починається з)
* a[href$=".pdf"] (закінчується на)
* a[href*="example"] (містить)

Псевдокласи та псевдоелементи

Псевдокласи (:) дозволяють стилізувати елементи на основі їхнього стану або позиції.

* :hover (наведення миші)
* :active (активний клік)
* :focus (фокус елемента)
* :visited (відвідане посилання)
* :nth-child(n) (кожен n-й дочірній елемент)
* :first-child, :last-child
* :not(selector) (елементи, які не відповідають селектору)

Псевдоелементи (::) дозволяють стилізувати певні частини елемента.

* ::before (додає вміст перед елементом)
* ::after (додає вміст після елемента)
* ::first-line (перший рядок тексту)
* ::first-letter (перша літера тексту)
* ::selection (виділений текст)

Розділ 3: Кольори, Текст та Шрифти

Кольори

CSS дозволяє використовувати різні формати для задання кольорів:

* За назвою: red, blue, green
* HEX-коди: #RRGGBB (наприклад, #FF0000 для червоного)
* RGB: rgb(red, green, blue) (наприклад, rgb(255, 0, 0))
* RGBA: rgba(red, green, blue, alpha) (додає прозорість, alpha від 0 до 1)
* HSL: hsl(hue, saturation, lightness)
* HSLA: hsla(hue, saturation, lightness, alpha)

Властивості кольорів:

* color: колір тексту
* background-color: колір фону
* border-color: колір рамки

Текст

 * text-align: left, right, center, justify
 * text-decoration: none, underline, overline, line-through
 * text-transform: none, uppercase, lowercase, capitalize
 * text-shadow: h-shadow v-shadow blur-radius color
 * letter-spacing: інтервал між символами
 * word-spacing: інтервал між словами
 * line-height: висота рядка

Шрифти

 * font-family: тип шрифту (наприклад, Arial, sans-serif)
 * font-size: розмір шрифту (px, em, rem, %)
 * font-weight: жирність шрифту (normal, bold, 100-900)
 * font-style: стиль шрифту (normal, italic, oblique)
 * font-variant: normal, small-caps

Приклад:

```
body {
    font-family: 'Open Sans', Arial, sans-serif;
    color: #333;
    font-size: 18px;
    line-height: 1.6;
}

h2 {
    font-weight: bold;
    text-align: center;
    text-transform: uppercase;
}
```

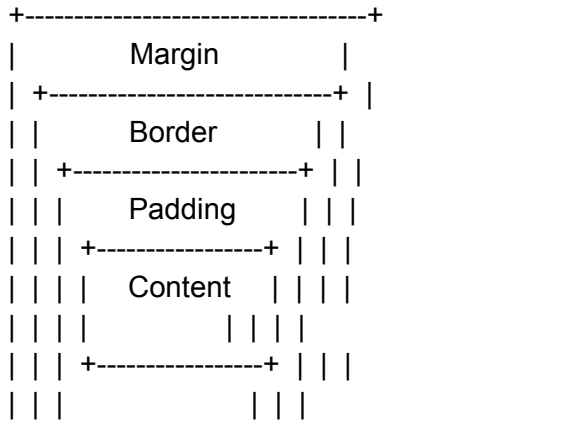
Розділ 4: Модель Коробки (Box Model)

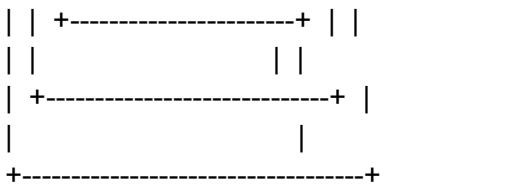
Кожен HTML-елемент на веб-сторінці розглядається браузером як прямокутна "коробка". Розуміння Box Model є ключовим для макетування.

Компоненти Box Model

 * Content (Вміст): Фактичний вміст елемента (текст, зображення тощо).
 * Padding (Внутрішні відступи): Простір між вмістом і рамкою. Прозорий.
 * Border (Рамка): Лінія, яка оточує padding та content.
 * Margin (Зовнішні відступи): Простір навколо рамки. Прозорий. Використовується для створення відстані між елементами.

<!-- end list -->





Властивості Box Model
 * width / height: Ширина та висота вмісту.
 * padding: padding-top, padding-right, padding-bottom, padding-left або скорочено padding: top right bottom left;
   * padding: 10px; (всі сторони 10px)
   * padding: 10px 20px; (верх/низ 10px, ліво/право 20px)
   * padding: 10px 20px 30px; (верх 10px, ліво/право 20px, низ 30px)
 * border: border-width, border-style, border-color або скорочено border: 1px solid black;
   * border-radius: закруглення кутів
 * margin: margin-top, margin-right, margin-bottom, margin-left або скорочено margin: top right bottom left;
   * margin: auto; (для блокових елементів, щоб центрувати їх по горизонталі)
box-sizing
Дуже важлива властивість, що контролює, як width та height враховують padding та border.
 * content-box (за замовчуванням): width та height відносяться лише до вмісту. Padding та border додаються до загальної ширини/висоти.
 * border-box: width та height включають padding та border. Це полегшує розрахунки при макетуванні.
Рекомендація: Завжди використовуй box-sizing: border-box; для всіх елементів:

```
html {
    box-sizing: border-box;
}
*, *::before, *::after {
    box-sizing: inherit;
}
```

Розділ 5: Відображення Елементів та Позиціонування
Властивість display
Властивість display визначає, як елемент відображається в документі.
 * block: Елементи займають всю доступну ширину, починаються з нового рядка. Приклади: div, p, h1.
 * inline: Елементи займають стільки місця, скільки їм потрібно, не починаються з нового рядка. Приклади: span, a, img. width, height, margin-top, margin-bottom не працюють.
 * inline-block: Комбінація inline та block. Елементи відображаються в одному рядку, але можуть мати width, height, margin та padding.
 * none: Елемент повністю видаляється з документа (не відображається і не займає місце).
 * flex: Включає Flexbox-контейнер.
 * grid: Включає Grid-контейнер.
Позиціонування (position)
Властивість position контролює, як елемент позиціонується на сторінці.

* static (за замовчуванням): Елемент знаходиться в нормальному потоці документа. Властивості top, right, bottom, left не діють.
* relative: Елемент позиціонується відносно своєї нормальної позиції. top, right, bottom, left зміщують його, але простір, який він займає, залишається незмінним.
* absolute: Елемент вилучається з нормального потоку та позиціонується відносно найближчого позиціонованого предка (тобто предка з position: relative, absolute, fixed або sticky). Якщо такого предка немає, позиціонується відносно <html>.
* fixed: Елемент вилучається з нормального потоку та позиціонується відносно вікна перегляду (viewport). Залишається на місці навіть при прокручуванні сторінки.
* sticky: Елемент поводиться як relative, доки не досягне певної точки при прокручуванні, після чого стає fixed.

Приклад:

```
.container {
    position: relative; /* Батьківський елемент для absolute */
    width: 300px;
    height: 200px;
    border: 1px solid black;
}

.box {
    position: absolute;
    top: 20px;
    right: 20px;
    width: 50px;
    height: 50px;
    background-color: lightcoral;
}
```

z-index

Контролює порядок накладання позиціонованих елементів. Елементи з більшим z-index знаходяться зверху. Працює тільки для позиціонованих елементів.

Розділ 6: Flexbox

Flexbox (Flexible Box Layout Module) — це одномірна система макетування, що дозволяє легко розташовувати елементи в ряд або стовпець.

Основні концепції Flexbox

* Flex Container (Батьківський елемент): Елемент, до якого застосовуються властивості display: flex;.
* Flex Items (Дочірні елементи): Елементи, що знаходяться всередині Flex Container.

Властивості Flex Container

* display: flex;: Перетворює елемент на flex-контейнер.
* flex-direction: Визначає основну вісь (напрямок) елементів:
  * row (за замовчуванням): зліва направо
  * row-reverse: справа наліво
  * column: зверху вниз
  * column-reverse: знизу вгору
* justify-content: Вирівнювання елементів по основній осі:
  * flex-start
  * flex-end

  * center
  * space-between (рівний простір між елементами)
  * space-around (рівний простір навколо елементів)
  * space-evenly (рівний простір між та навколо елементів)
* align-items: Вирівнювання елементів по поперечній осі:
  * flex-start
  * flex-end
  * center
  * baseline
  * stretch (розтягує елементи, щоб заповнити контейнер)
* flex-wrap: Дозволяє елементам переноситися на новий рядок (для row) або стовпець (для column).
  * nowrap (за замовчуванням)
  * wrap
  * wrap-reverse
* align-content: Вирівнювання рядків (якщо flex-wrap: wrap;) по поперечній осі.

Властивості Flex Items

* flex-grow: Визначає, наскільки елемент може збільшуватися.
* flex-shrink: Визначає, наскільки елемент може зменшуватися.
* flex-basis: Початковий розмір елемента перед розподілом вільного простору.
* flex: Скорочена властивість для flex-grow, flex-shrink, flex-basis. (наприклад, flex: 1 1 auto;)
* order: Змінює візуальний порядок елементів.
* align-self: Перевизначає align-items для окремого елемента.

Приклад Flexbox:

```
<div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
</div>

.flex-container {
    display: flex;
    justify-content: center; /* Центрувати елементи по горизонталі */
    align-items: center; /* Центрувати елементи по вертикалі */
    height: 200px;
    border: 1px solid #ccc;
}

.flex-item {
    background-color: lightgreen;
    padding: 20px;
    margin: 10px;
}
```

Розділ 7: Grid Layout

CSS Grid Layout — це двовимірна система макетування, що дозволяє створювати складні сітки для розміщення елементів.

Основні концепції Grid
 * Grid Container (Батьківський елемент): Елемент, до якого застосовуються властивості display: grid;.
 * Grid Items (Дочірні елементи): Елементи, що знаходяться всередині Grid Container.
 * Grid Lines (Лінії сітки): Горизонтальні та вертикальні лінії, що утворюють сітку.
 * Grid Tracks (Доріжки сітки): Простір між двома паралельними лініями сітки (рядки або стовпці).
 * Grid Cells (Клітинки сітки): Перетин рядка та стовпця.
 * Grid Areas (Області сітки): Іменовані області сітки, що охоплюють кілька клітинок.
Властивості Grid Container
 * display: grid;: Перетворює елемент на grid-контейнер.
 * grid-template-columns: Визначає кількість та ширину стовпців.
   * grid-template-columns: 100px 1fr 2fr; (фіксована ширина, потім відносні)
   * grid-template-columns: repeat(3, 1fr); (3 стовпці однакової ширини)
 * grid-template-rows: Визначає кількість та висоту рядків.
 * gap (або grid-gap): Простір між клітинками.
   * row-gap, column-gap
 * justify-items: Вирівнювання елементів всередині клітинок по горизонталі.
 * align-items: Вирівнювання елементів всередині клітинок по вертикалі.
 * justify-content: Вирівнювання сітки в контейнері по горизонталі.
 * align-content: Вирівнювання сітки в контейнері по вертикалі.
 * grid-template-areas: Дозволяє іменувати області сітки та розміщувати елементи за назвами.
Властивості Grid Items
 * grid-column-start, grid-column-end (або grid-column: start / end;)
 * grid-row-start, grid-row-end (або grid-row: start / end;)
 * grid-area: Для розміщення елементів в іменованих областях.
 * justify-self, align-self: Перевизначають вирівнювання для окремого елемента.
Приклад Grid Layout:

```
<div class="grid-container">
    <header class="grid-item">Header</header>
    <nav class="grid-item">Nav</nav>
    <main class="grid-item">Content</main>
    <aside class="grid-item">Sidebar</aside>
    <footer class="grid-item">Footer</footer>
</div>

.grid-container {
    display: grid;
    grid-template-columns: 1fr 3fr 1fr; /* Три стовпці */
    grid-template-rows: auto 1fr auto; /* Три рядки */
    gap: 10px;
    grid-template-areas:
        "header header header"
        "nav content sidebar"
        "footer footer footer";
    min-height: 100vh;
}
```

```
.grid-item {
    background-color: #f0f0f0;
    padding: 20px;
    border: 1px solid #ddd;
}

header { grid-area: header; }
nav    { grid-area: nav; }
main   { grid-area: content; }
aside  { grid-area: sidebar; }
footer { grid-area: footer; }
```

Розділ 8: Адаптивний Дизайн (Responsive Design)
Адаптивний дизайн дозволяє твоїм веб-сторінкам виглядати добре на різних пристроях та розмірах екрану.
Медіа-запити (@media)
Медіа-запити дозволяють застосовувати CSS-стилі тільки тоді, коли виконуються певні умови (наприклад, розмір екрану).

```
/* Базові стилі для мобільних пристроїв */
body {
    font-size: 16px;
}

/* Стилі для екранів шириною від 768px і вище */
@media screen and (min-width: 768px) {
    body {
        font-size: 18px;
    }
    .grid-container {
        grid-template-columns: 200px 1fr; /* Змінюємо сітку для більших екранів */
        grid-template-areas:
            "header header"
            "nav content"
            "footer footer";
    }
    aside { display: none; } /* Приховуємо сайдбар на деяких брейкпоінтах */
}

/* Стилі для екранів шириною від 1200px і вище */
@media screen and (min-width: 1200px) {
    body {
        font-size: 20px;
    }
    .grid-container {
        grid-template-columns: 1fr 3fr 1fr;
        grid-template-areas:
            "header header header"
```

```
            "nav content sidebar"
            "footer footer footer";
    }
    aside { display: block; }
}
```

Мета-тег Viewport
Обов'язковий для адаптивного дизайну, щоб браузер правильно масштабував сторінку.
Розміщується в <head>:
<meta name="viewport" content="width=device-width, initial-scale=1.0">

Розділ 9: Переходи та Анімації
Переходи (transition)
Дозволяють плавно змінювати значення властивостей CSS.
 * transition-property: властивість, яку анімуємо (наприклад, background-color, all)
 * transition-duration: тривалість переходу
 * transition-timing-function: швидкість переходу (ease, linear, ease-in, ease-out, ease-in-out)
 * transition-delay: затримка перед початком переходу
Приклад:

```
.button {
    background-color: blue;
    color: white;
    padding: 10px 20px;
    transition: background-color 0.3s ease-in-out; /* Плавна зміна кольору */
}

.button:hover {
    background-color: darkblue;
}
```

Анімації (animation)
Дозволяють створювати більш складні, багатоетапні анімації за допомогою @keyframes.
 * Визначення Keyframes:

```
   @keyframes slideIn {
    0% { /* Початковий стан */
        transform: translateX(-100%);
        opacity: 0;
    }
    50% {
        opacity: 0.5;
    }
    100% { /* Кінцевий стан */
        transform: translateX(0);
        opacity: 1;
    }
}
```

* Застосування Анімації:
  * animation-name
  * animation-duration
  * animation-timing-function
  * animation-delay
  * animation-iteration-count: infinite або число повторень
  * animation-direction: normal, reverse, alternate, alternate-reverse
  * animation-fill-mode: forwards, backwards, both, none (що відбувається до/після анімації)

Приклад:

```
.animated-element {
    animation-name: slideIn;
    animation-duration: 2s;
    animation-timing-function: ease-out;
    animation-iteration-count: 1;
    animation-fill-mode: forwards; /* Зберігає кінцевий стан */
}
```

Розділ 10: Поради та Рекомендації

* Організація CSS:
  * Використовуй зовнішні файли CSS.
  * Організуй файли за логічними блоками (наприклад, base.css, layout.css, components.css).
  * Використовуй коментарі для пояснення складних частин коду.
* Змінні CSS (Custom Properties): Дозволяють визначати багаторазові значення (наприклад, кольори, розміри шрифтів) та легко їх змінювати.

```
  :root { /* Глобальні змінні */
    --primary-color: #007bff;
    --font-size-base: 16px;
}

body {
    font-size: var(--font-size-base);
    color: var(--primary-color);
}
```

* Методології CSS:
  * BEM (Block, Element, Modifier): Допомагає створювати модульний та масштабований CSS. (Наприклад, .block__element--modifier)
  * SMACSS, ITCSS
* Препроцесори CSS (Sass/Less): Додають функціонал, якого немає в чистому CSS (змінні, міксини, вкладеність, функції).
* Фреймворки CSS (Bootstrap, Tailwind CSS): Готові набори стилів та компонентів для швидкої розробки.
* Інструменти розробника (Developer Tools): Використовуй інструменти браузера (Chrome DevTools, Firefox Developer Tools) для