ТЕМА : Основи CSS. Підключення шрифтів в CSS

CSS (Cascading Style Sheets), або каскадні таблиці стилів, описують правила відображення та розміщення окремого елемента веб-сторінки. Правила створення стилю складаються з двох основних частин: **селектора** і **блоку оголошення**.

Селектор повідомляє браузеру, який саме елемент форматувати, в блоці оголошення перелічено властивості форматування та їх значення.

Рис. 2.3. Структура оголошення стилю елемента в CSS

Додавання CSS до веб-сторінки

Вбудовані таблиці стилів

Вбудовані стилі знаходяться між тегами <style> ... </ style>, що вставляються всередину елемента <head>. Вбудовані стилі діють тільки на сторінці, на якій вони містяться. На одній сторінці можна розміщувати довільну кількість вбудованих стилів:

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

Внутрішні стилі елементів

Внутрішні стилі елементів не використовують селектори, опис стилю відбувається безпосередньо через атрибут style в початковому тезі елементу:

```
<p style="font-family:"Times New Roman", Georgia, Serif; color:#70d7700;">Зверніть увагу на цей текст.</p>
```

Зовнішні таблиці стилів

Зовнішня таблиця стилів представляє текстовий файл з розширенням .css, в якому знаходиться весь набір стилів CSS. Задані в файлі стилі будуть працювати для всіх сторінок веб-сайту. Під'єднати зовнішній файл зі стилями можна в два способи:

Прикріплення до веб-сторінки за допомогою тега <link>, вкладеного в тег <head>:

```
<head>
    <link rel="stylesheet" type="text/css" href="style1.css">
    <link rel="stylesheet" type="text/css" href="style2.css">
</head>
```

де rel="stylesheet" вказує тип посилання (посилання на таблицю стилів), а type = "text/css" повідомляє браузеру тип даних, в даному випадку це текстовий файл, що містить css-код.

Прикріплення до веб-сторінки за допомогою правила @import

Правило @import дозволяє завантажити зовнішню таблицю стилів. Щоб директива @import працювала, вона повинна розташовуватися всередині тегу <style> перед іншими правилами:

```
<style type="text/css">
    @import url(mobile.css);
```

p {font-size: 0.9em; color: grey;}

</style>

CSS-функція @font-face можна використовувати для завантаження власного шрифту, щоб ви не були повністю залежними від шрифтів, встановлених на комп'ютері користувача.

Функція @font-face дозволяє вам вибирати з більш широкого кола шрифтів, ніж доступно звичайно, спираючись виключно на системні шрифти користувача.

Можливі значення

Шрифти, зазначені за допомогою команди @font-face при правильній версії, виводяться та активуються лише у тому випадку, якщо вони використовуються у документі *HTML*. Це добре для продуктивності, оскільки шрифт не завантажується, якщо цього не потрібно.

Ось приклад використання функції @font-face:

```
@font-face {
  font-family: "Open Sans";
  src: url("opensans.woff2") format("woff2");
}
body {
  font-family: "Open Sans", sans-serif;
}
```

У цьому прикладі за допомогою @font-face відбувається завантаження шрифту. Як сімейство шрифтів використовується "Open Sans". Потім йде звертання до цього сімейства шрифтів, коли він використовується для елемента <body>.

Не використовуючи @font-face, шрифти, які користувач може побачити на сторінці, буде обмежена шрифтами, встановленими на його пристрої.

Отже, якщо у прикладі не використовувалася функція @font-face для підключення шрифту, шрифт "Open Sans" для елемента <body> буде працювати, лише якщо користувач встановив цей шрифт на своєму

пристрої. Якщо ні, його браузер використовуватиме стандартний шрифт з сімейства sans-serif. Багато систем використовують *Helvetica* або *Arial* як стандартний шрифт sans-serif, проте немає гарантії, що навіть ці шрифти будуть встановлені в системі користувача.

Використання декількох форматів файлів

Ви можете надати список форматів шрифтів для завантаження. Якщо система користувача не підтримує один формат, він може підтримувати наступний у своєму списку і так далі.

Ось приклад:

```
@font-face {
  font-family: "Open Sans";
  src: local("Open Sans"),
    url("opensans.woff2") format("woff2"),
    url("opensans.woff") format("woff"),
    url("opensans.ttf") format("truetype");
}
body {
  font-family: "Open Sans", sans-serif;
}
```

У наведеному вище прикладі використовується функція local(), щоб отримати шрифт з локальної системи користувача. Якщо його не знайдено, він буде шукати наступний параметр - у цьому випадку - параметр *.woff2*. Якщо система не підтримує цей формат шрифту, вона буде використовувати опцію *.woff*. Якщо він не підтримує цей параметр, він буде використовувати опцію *.ttf*.

Якщо жоден з цих параметрів не працює, елемент <body> буде в кінцевому підсумку використовувати стандартний шрифт sans-serif у системі користувача.

Можливі значення

У @font-face прописуються такі дескриптори, які визначають місце розташування шрифтів як локально, так і зовні, а також стильові характеристики окремого сімейства шрифтів:

font-family

Це дескриптор сімейства шрифтів, який визначає назву шрифту. Це ім'я, про яке ви посилаєтесь у своїх деклараціях CSS, коли ви хочете послатися на цей шрифт. Цей дескриптор обов'язковий.

src

Визначає адресу за якою розташований шрифт. Його значення - це пріоритетний список, розділений комами, список зовнішніх або локальних посилань. Коли викликається шрифт, браузер проходить список наведених посилань, використовуючи перший, який він може успішно активувати. Шрифти, що містять недійсні дані або не знайдені локальні шрифти, ігноруються, а браузер завантажує наступний шрифт зі списку. Цей дескриптор обов'язковий.

Приклад підключення CSS

Приклад. Опрацювати і проаналізувати даний приклад, який складається з файлів: **index.html і main.css та color.css**

ФАЙЛ : index.html

```
<html>
<head>
   <title>Вивчаємо CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css" />
   <style>
      p {
         font-family: Verdana;
      }
   </style>
</head>
<body>
   <p>Деякий текст...</p>
```

```
    <p>Деякий текст...</p>
  <p style="font-style: italic;">Деякий текст...</p>
  <p> Ваша оцінка: <span style="color: #0c0;">5</span></p>
  <p> Ваша оцінка: <span style="color: #c00;">2</span></p>
</body>
</html>
```

ФАЙЛ : main.css

```
@import url("color.css");

p {
        font-size: 150%;

}
```

ФАЙЛ : color.css

```
p{
  color: #c00;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

Приклад підключення шрифтів

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: **index.html і main.css**

ФАЙЛ : index.html

```
<html>
<head>
   <title>Изучаем CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css" />
```

```
    </head>
    <body>
        <!-- Шрифты: http://allfont.ru/ и https://dafontfree.net -->
        <!-- Конвертер шрифтов: https://onlinefontconverter.com/ -->
        <h1>Заговолок незвичайним шрифтом</h1>
        <p>Текст незвичайним шрифтом</p>
    </body>
</html>
```

ФАЙЛ : main.css

```
    @font-face {
     font-family: BadScript;
     src: url("../fonts/BadScript.woff2") format("woff2"), url
("../fonts/BadScript.woff") format ("woff");
    }

    body {
        font-family: BadScript;
    }

    h1 {
        text-align: center;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

Всі файли і шрифти бажано зберігати в одній папці. Довідник по CSS і HTML https://css.in.ua/

ЗАВДАННЯ:

Завдання 1.

Виберіть вподобаний шрифт в Інтернеті і скачайте його.

Завдання 2.

Конвертує його в необхідні формати.

Завдання 3.

Підключіть шрифт в CSS і переконайтеся, що він працює.

ТЕМА: Одиниці виміру і зовнішній вигляд тексту в CSS

Стрічковий тег - span

При оформленні тексту за допомогою css найчастіше використовують тег . Він позначає «просто певну частину тексту». Тобто власного значення він не має. Також цей тег ніяк не змінює відображення тексту. Однак, додатковий сенс даного тегу додають за допомогою класів. Наприклад:

 </ span>

 </ span>

А вже для класу за допомогою css задають стилі і тим самим змінюють оформлення.

Властивість font-size: задаємо розмір шрифту

Властивість font-size задає розмір шрифту. Розмір шрифту найкраще здавати у відносних одиницях виміру – em. 1em звичайно дорівнює довжині літери M в даному шрифті.

Інші одиниці виміру для завдання розмірів шрифту:

- пікселі: 20px
- пункти: 15pt
- відсотки: 80%

Пікселі і пункти - це абсолютні одиниці виміру, а відсотки - відносні. Також розмір шрифту можна задавати за допомогою ключових слів: small, large і т.д. Але їх зазвичай не використовують.

Властивість font-weight: товщина накреслення

Напівжирний текст можна задавати за допомогою властивості font-weight, що має два основних значення:

1. normal - звичайне написання;

2. bold - напівжирне накреслення.

Насправді ця властивість має багато значень: bold, bolder, lighter, normal, 100, 200, 300, 400, 500, 600, 700, 800, 900. Ці значення задають ступінь товщини шрифту, від самого тонкого, до самого товстого. Але більшість браузерів все одно вміють відображати тільки два варіанти товщини: звичайний і напівжирний. Тому інші значення властивості зазвичай не використовують.

Властивість font-style: курсив

Накреслення тексту можна задавати за допомогою властивості font-style. Її основні значення:

1. normal - звичайне написання;

2. italic - курсивне зображення.

Ця властивість має й інші значення, але їх майже не використовують.

Властивість font-family: шрифт

Задати сімейство шрифту можна за допомогою властивості font-family. Можна задавати конкретну назву шрифту: "Times New Roman". А можна задавати бажаний тип шрифту, наприклад:

- serif - шрифт із зарубками;
- sans-serif - шрифт без зарубок.

Є й інші типи, але вони використовуються рідше.

Зазвичай, як значення властивості задають список шрифтів, перераховуючи їх через кому. На початку списку розташовують найрідкісніший шрифт, потім схожий, але поширений, а в самому кінці списку - бажаний тип шрифту. Приклад:

```
body {
font-family: "PT Sans", "Arial", serif;
}
```

Браузер проходить за списком зліва направо і використовує перший знайдений в системі шрифт.

Властивість color: колір тексту

Колір тексту задається за допомогою властивості color. Значення кольору можна задавати різними способами:

1. Шістнадцятковим кодом, наприклад #FF9900.
2. Ключовим словом: red.
3. У RGB-форматі: rgb(255, 255, 0).

Найчастіше колір задають в шістнадцятковому форматі.

Властивість text-decoration: підкреслення та інші ефекти

Додаткове оформлення тексту можна задати за допомогою властивості text-decoration. Її значення:

1. underline - підкреслення;

2. line-through - закреслення;

3. overline – лінія над текстом;

4. none - прибирає вище перелічені ефекти.

До тексту можна одночасно застосувати кілька ефектів, якщо перерахувати значення через пропуск.

Декоративне підкреслення

Ви, напевно, вже багато разів бачили красиве пунктирне підкреслення. Його використовують для оформлення посилань та інших динамічних елементів. Технологія наступна:

1. Прибираємо звичайне підкреслення за допомогою text-decoration.

2. Задаємо потрібний колір тексту за допомогою color.

3. Додаємо декоративне підкреслення за допомогою властивості border-bottom.

Також можна при наведенні курсору приховувати таке підкреслення за допомогою псевдо класу: hover.

Задаємо регістр символів за допомогою text-transform

За допомогою css можна керувати навіть регістром символів: робити букви малими або великими. Робиться це за допомогою властивості text-transform. Її значення:

1. lowercase - усі малі;

2. uppercase - всі прописні;

3. capitalize - кожне слово починається із прописної;

4. none - Скасовує зміна регістру.

Управляємо пробілами: white-space

Як ви вже знаєте, браузер ігнорує множинні пробіли та переноси рядків в HTML-коді. Змінити цю поведінку можна за допомогою тегу <pre>. Однак, за допомогою CSS управляти пробілами і переносами можна більш гнучкіше. За це відповідає властивість white-space, значення якого:

- nowrap - відображає весь текст одним рядком без переносів;
- pre - зберігає пробіли та переноси стрічки як у вихідному коді аналогічно тегу <pre>;
- pre-wrap - працює як значення pre, але додає автоматичні переноси, якщо текст не поміщається в контейнер;
- normal - режим за замовчуванням.

Вертикальне вирівнювання: vertical-align

Вирівнюванням тексту по вертикалі можна керувати за допомогою властивості vertical-align. Її дія добре помітна в елементах

таблиці, в середині текстового рядка «робота» цієї властивості помітна, якщо в ній є фрагменти різного розміру.

У даної властивості багато значень, але найбільш часто використовувані:

1. top - вирівнювання по верхньому краю рядка;

2. middle - по середині;

3. bottom - по нижньому краю;

4. baseline - за базовою лінії (значення за замовчуванням).

Верхні і нижні індекси на CSS

За допомогою CSS можна імітувати теги <sub> і <sup>, які застосовуються для створення нижніх і верхніх індексів. Робиться це так:

1. використовуємо властивість vertical-align із значенням sub або super.

2. трохи зменшуємо розмір шрифту за допомогою font-size.

Властивість line-height: управляємо висотою рядка

Висотою рядка або, правильніше, міжрядковим інтервалом можна керувати за допомогою властивості line-height. Значення цієї властивості можна задавати наступними способами:

1. множником, наприклад 1.5, 2.

2. у відсотках: 150%.

3. за допомогою будь-яких інших одиниць виміру CSS: 12px, 2em.

4. ключовим словом normal, яке задає автоматичний розрахунок висоти рядка.

Переважно міжрядковий інтервал задається або множником, або у відносних одиницях виміру.

Вертикальний ритм тексту

У веб-дизайні існує поняття «вертикальний ритм тексту». Воно досить складне і ми не будемо глибоко в нього вдаватися. Відзначимо лише, що хороший вертикальний ритм покращує сприйняття тексту.

Щоб зберегти вертикальний ритм і хорошу читабельність тексту при будь-якому розмірі шрифту, потрібно задавати розміри шрифту, міжрядковий інтервал і вертикальні відступи між заголовками і абзацами у відносних одиницях.

Тоді, як би користувач не змінював розмір шрифту, ваш текст залишиться читабельним і зручним для сприйняття.

Приклад. Опрацювати і проаналізувати даний приклад, який складається з двох файлів: **index.html і main.css.**

ФАЙЛ : index.html

```
<html>
<head>
   <title>Вивчаємо CSS</title>
   <link type="text/css" rel="stylesheet" href="css/main.css">
</head>
<body>
   <h1>Заголовок незвичайним шрифтом</h1>
   <h2>Підзаголовок</h2>
   <p>Текст незвичайним шрифтом.</p>
   <div>Текст в <strong>strong</strong></div>
   <div>Текст в <em>em</em></div>
   <p>Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ...
```

Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ... Дуже довгий текст ...</p>

```
        <p class="underline">ТЕКСТ</p>
        <p class="strike">ТЕКСТ</p>
        <a href="#">Посилання</a>
        <a class ="upper" href="#">Посилання 1 </a>
        <a class ="lower" href="#">Посилання 2 </a>
        <p class="shadow">ТЕКСТ з тінню</p>
    </body>
</html>
```

ФАЙЛ main.css.

```
    @font-face {
        font-family: BadScript;
        src: url("../fonts/BadScript.woff2") format("woff2"), url("../fonts/BadScript.woff") format("woff");
    }

    body {
        font-family: BadScript;

    }

    h1,h2 {
            text-align: center;
    }

    h1{
        color:#0c0;
```

```
}
p  {
   font-size: 150%;
   font-weight: lighter;
   line-height: 300%;
   text-indent: 20px;
   text-align:justify;
}

strong {
   font-style:italic:
   font-weight:normal;
}

em{
   font-style:normal;
   color:red;
}

.underline{
   text-decoration:underline;
   text-indent: 0;
}

.strike{
   text-decoration:line-through;
}

a{
   color:#c00;
```

```
        text-decoration:none;
        font-size: 200%;
    }

    a.upper{
        text-transform:uppercase;
    }
    a.lower{
            text-transform:lowercase;
    }
    .shadow{
        text-align:right;
        text-shadow: 5px 4px 8px #555;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

ЗАВДАННЯ:

ЗАВДАННЯ 1

Введіть кілька абзаців тексту.

ЗАВДАННЯ 2

Зробіть текст великого розміру.

ЗАВДАННЯ 3

Зробіть текст зеленим кольором.

ЗАВДАННЯ 4

Зробіть відступи у перших рядків абзаців.

ЗАВДАННЯ 5

Додайте ще текст в тезі div і зробіть його жирним.

ЗАВДАННЯ 6

Вирівняйте текст в div по правому краю.

ЗАВДАННЯ 7

Зробіть два абзаци тексту з тінню без абзацного відступу, тінь другого абзацу повинна бути у протилежному напрямку

ЗАВДАННЯ 8

Продемонструйте створення нижніх і верхніх індексів використовуючи властивість vertical-align

ТЕМА: Селектори

За допомогою селекторів створюються CSS-правила для форматування елементів сторінки. Як селектори можна використовувати елементи, класи, ідентифікатори, а також псевдо класи і псевдо елементи.

Універсальний селектор. Універсальний селектор позначає правила, що стосуються всіх елементів, наприклад, наступний запис обнулить відступи для всіх елементів веб-сайту:

```
* {Margin: 0;}
```

Селектор Елементу. Селектори елементів використовуються для визначення стилів для всіх даних елементів сайту, наприклад, стиль заголовків h1 або загальний стиль абзаців:

```
h1 {font-family: Lobster, cursive;}
p {letter-spacing: 0.1em;}
```

Селектор Класу. Селектори класу використовуються для визначення стилів, які можна застосувати для різних елементів, розміщених в різних частинах або на різних сторінках сайту.

Код HTML	Код CSS
<h1 class="headline">Інструкція	headline {

користування персональним комп'ютером</h1> Примітка. Це важливо для роботи	text-transform: uppercase; color: lightblue; }

Селектор Ідентифікатора. Селектори ідентифікатора використовуються для привласнення стилю одному конкретному елементу. Ідентифікатор id належить унікальному елементу, тому його можна використовувати в документі лише один раз.

#sidebar {text-transform: uppercase; color: lightblue;}

Селектор Нащадку. До нащадків елемента відносяться його дочірні елементи. Селектори нащадків дозволяють стилізувати всі вкладені елементи, наприклад, можна від форматувати зовнішній вигляд всіх елементів маркованого списку:

ul li {text-transform: uppercase;}

Якщо потрібно від форматувати нащадки певного елемента, то можна поставити йому стильовий клас:

p.first a {color: green;} - означає, що потрібно застосувати даний стиль до всіх посилань, нащадків абзацу, що відноситься до класу з назвою first;

p .first a {color: green;} - якщо додати пробіл, то будуть обрані посилання, розташовані всередині будь-якого тегу класу .first, який є нащадком елемента ;

.first a {color: green;} - даний стиль застосовується до любого посилання, що розташоване всередині інших тегів, позначених класом .first.

Дочірній селектор. Дочірній тег є прямим нащадком тегу, що його містить. Тобто, відносини "батьки-діти" існують між елементами і тими елементами, які містяться безпосередньо всередині них. В одного елемента може бути кілька

дочірніх елементів, а батьківський елемент може бути в кожного елемента тільки один.

p> strong - вибирає всі елементи strong, які є дочірніми по відношенню до елементу p.

Сестринський селектор. Сестринські відносини виникають між елементами, що мають загального батька. Селектори сестринських елементів дозволяють вибрати теги з групи елементів одного рівня.

h1 + p - вибере всі перші абзаци, що йдуть безпосередньо за будь-яким тегом <h1>, не зачіпаючи решта абзаців.

h1 ~ p - вибере всі абзаци, які є сестринськими по відношенню до будь-якого заголовку h1 і йдуть після нього.

Селектор Атрибуту. Селектори атрибутів дозволяють форматувати елементи на основі вибірки будь-яких атрибутів, що містяться в них або значень атрибутів, наприклад:

[атрибут] - вибирає всі елементи, для яких задано вказаний атрибут.

img [alt] - вибирає всі картинки, що містять атрибут alt.

img [title = "flower"] - вибирає всі картинки, назва яких містить слово flower.

Псевдо класи

Псевдо класи дозволяють додавати особливі класи до елементів, вибираючи об'єкти, яких немає в структурі веб-сторінки, або які не можна вибрати за допомогою звичайних селекторів, наприклад, перша літера або перший рядок одного абзацу. Псевдо класи добре проілюстровані різними станами посилання, наприклад:

a: link – описує стиль невідвідуваного посилання.

a: visited - описує стиль вже відвіданого посилання.

a: hover - описує вигляд елементу, над яким проходить вказівник мишки.

a: focus - описує вигляд елементу, над яким знаходиться (сфокусований) вказівник.

a: active - описує вигляд елементу, який активовано користувачем.

Структурні псевдо класи. Структурні псевдо класи форматують дочірні елементи відповідно до зазначеного параметра в дужках, наприклад:

:nth-child (odd) - вибирає непарні дочірні елементи.

:nth-child (even) - вибирає парні дочірні елементи.

:nth-child (3n) - вибирає кожен третій елемент серед дочірніх.

Структурні псевдо класи типу. Вказують на конкретний тип дочірнього тегу:

:nth-of-type () - вибирає елементи за аналогією з: nth-child (), при цьому бере до уваги тільки тип елемента.

:first-of-type - дозволяє вибрати перший дочірній елемент.

:last-of-type - вибирає останній дочірній елемент конкретного типу.

:nth-last-of-type () - вибирає елемент заданого типу в списку елементів відповідно до зазначеного місцеположення, починаючи з кінця.

:only-of-type - вибирає єдиний елемент зазначеного типу серед дочірніх елементів батьківського елементу.

Псевдо елементи. Псевдо елементи не є елементами сторінки, їх використовують для додавання вмісту, який генерується за допомогою властивості content, і для зміни зовнішнього вигляду частини елементу:

:before – додає певний вміст перед елементом.

:after - додає певний вміст після елемента.

Комбінації селекторів

Щоб домогтися більш чіткого вибору елементів для форматування, можна не обмежуватися завданням одного типу селектора, а використовувати комбінації селекторів, наприклад:

a [href] [title] - вибере всі посилання, для яких задані атрибути href і title;

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

Угруповання селекторів

Можна застосувати один стиль до кількох елементів, причому обмежень за кількістю елементів немає. Для цього необхідно в лівій частині оголошення помістити через кому потрібні селектори, наприклад:

h1, h2, h3, h4 {color: tomato; background: white;}

Принцип каскадування і специфічність правила

Каскадування представляє процес застосування різних правил до одного і того ж елементу. Більш конкретні правила мають пріоритет над більш загальними. Якщо у відношення одного і того ж елемента визначено кілька стилів, то в результаті до нього буде застосований останній з них.

Для кожного правила браузер обчислює специфічність селектора, і якщо у елемента є конфліктуючі оголошення властивостей, до уваги береться правило, що має найбільшу специфічність.

Значення специфічності складається з чотирьох частин: 0, 0, 0, 0. Специфічність селектора визначається наступним чином:

- для id додається 0, 1, 0, 0;

- для class додається 0, 0, 1, 0;
- для кожного елементу і псевдо елемента додається 0, 0, 0, 1;
- для стилю, який до даного безпосередньо до елементу - 1, 0, 0, 0;
- універсальний селектор не має специфічності.

```
h1 {color: lightblue;} /* специфічність 0, 0, 0, 1 */
em {color: silver;} /* специфічність 0, 0, 0, 1 */
h1 em {color: gold;} /* специфічність: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2 */
#sidebar {color: orange;} /* специфічність 0, 1, 0, 0 */
li # sidebar {color: aqua;} /* специфічність: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0,
1 */
```

В результаті до елементу застосуються ті правила, специфічність яких більше, наприклад, якщо на елемент діють дві специфічності зі значеннями 0, 0, 0, 2 і 0, 1, 0, 1, то виграє друге правило.

Вагу правила також можна задати за допомогою ключового слова **!important;**, яке додається після значення властивості, наприклад, **font-weight: bold!Important;**. Таке оголошення буде мати пріоритет над всіма іншими правилами.

Більше інформації по правилах CSS можна дізнатися з специфікації мови та довідників.

Приклад з найбільш використовуваних селекторів. Опрацювати і проаналізувати даний приклад, який складається з двох файлів**: index.html і main.css.**

Файл index.html

```
<html>
<head>
    <title>Вивчаємо CSS</title>
    <link type="text/css" rel="stylesheet" href="css/main.css" />
</head>
<body>
    <h1 id="title">Заголовок</h1>
    <p>Текст</p>
    <p class="red">Червоний колір</p>
    <p class="red">Червоний колір</p>
    <p class="red">Червоний колір</p>
    <p>Текст: "<span>Зелений span</span>".</p>
    <div><span>Не Зелений</span></div>
    <form name="form" action="#" method="post">
        <p>Загаловок форми</p>
        <div>
            <p>Імя: </p>
            <input type="text" name="name" />
        </div>
        <div>
```

```
        <input type="submit" name="name" />
    </div>
</form>
<ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
</ul>
<table id="table">
    <tr>
        <td class="big">1</td>
        <td>2</td>
        <td>3</td>
        <td>4</td>
    </tr>
    <tr>
        <td>5</td>
        <td>6</td>
```

```
            <td>7</td>
            <td>8</td>
        </tr>
    </table>
    <p>
        <a href="#">Посилання</a>
    </p> </body> </html>
```

Файл main.css.

```
body {
    font-size: 200%;
}

p {
    color: #00c;
}

.red {
    color: #f00;
}
```

```
#title {
    color: #fa3900;
    text-align: center;
}

p span {
    color: #0c0;
}

body p span {
    text-shadow: 2px 3px 5px #000;
}

input[type="text"] {
    height: 40px;
    width: 400px;
}

input[type="text"]:focus {
    border: 2px solid #c00;
```

```
}

input[type="submit"] {
    color: #0099dd;
    cursor: pointer;
    height: 50px;
    width: 300px;
}

form > p {
    font-size: 200%;
}

ul li:first-child {
    color: #c00;
}

ul li:last-child {
    color: #0c0;
}
```

```
ul li:nth-child(3) {
    color: #aaa;
}

table {
    border: 2px solid #000;
    text-align: center;
    width: 30%;
}

td {
    border: 1px solid #000;
}

td:nth-child(2n) {
    color: #00c;
}

td:nth-child(2n + 1) {
```

```
    color: #a00;
}

a {
    color: #c00;
    text-decoration: none;
}

a:hover {
    color: #00c;
    text-decoration: underline;
}

#table tr .big {
    font-size: 200%;
}

#table tr .big:hover {
    color: #000;
}
```

Переглянути і проаналізувати результат цього прикладу в будь-якому браузері.

ЗАВДАННЯ:

Завдання 1.

Створіть меню, зробивши у кожного посилання свій id. Потім для кожного посилання за допомогою селектора ID вкажіть свій колір.

Завдання 2.

Зробіть так, щоб при наведенні курсору миші на посилання, розмір її тексту збільшувався.

Завдання 3.

Використовуючи селектор параметра, збільшіть розмір кнопки submit у форми, збільшіть розмір напису і змініть колір, а при наведенні курсору миші додайте суцільну рамку товщиною 5px сірого кольору.

Завдання 4.

Додайте 3 абзаци тексту з атрибутом class = "red" і 3 абзаци тексту з атрибутом class = "green".

Завдання 5.

Зробіть у всіх елементів class = "red" червоний колір тексту, а у всіх елементів з class = "green" - зелений.