

Лабораторна робота №5. Основи CSS.

Мета: опанувати основні можливості стилізації веб-сторінок за допомогою CSS.

Теоретичні відомості

CSS (Cascading Style Sheets — Каскадні таблиці стилів) — спеціальна мова, що відповідає за зовнішній вигляд HTML-сторінок.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3.

Профіль CSS — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

Синтаксис мови досить простий: він складається з селекторів та властивостей.

Стилі складаються зі списку правил. Кожне правило має один або більше селекторів та блок визначення. Блок визначення складається із оточеного фігурними дужками списку властивостей.

Найпростіші селектори — це селектори по іменах тегів. З їх допомогою можна задати стилі для всіх абзаців на сторінці, для всіх посилань, заголовків першого рівня і так далі. Таги селектори містять ім'я тега без символів < та >. Наприклад:

```
/* Визначаємо червоний колір та розмір шрифту для всіх тегів  
<p> на сторінці */  
p {  
  color: red;  
  font-size:  
}
```

До більш складних селекторів можна віднести селектори класів, ідентифікаторів, псевдокласів, контекстні селектори, селектори псевдоелементів та ідентифікаторів.

Селектори класів

Клас дозволяє об'єднувати різні елементи в смислові групи і застосовувати до них однакове оформлення. Наприклад, можна створити клас «елементи з помилкою» і задати йому червоний колір тексту. Потім можна додавати цей клас до будь-якого HTML-тегу: абзацу, заголовку, елементу списку і так далі. Клас тега можна задати за допомогою атрибуту `class`, який містить ім'я класу (або імена класів через пробіл). Наприклад:

```
<p class = "help">...</ p>
<p class = "help error">...</p>
```

Селектор з використанням класу задається так: `.ім'я_класа`. Наприклад:

```
.help {...}
.error {...}
```

Контекстні селектори

Селектор може складатися з декількох частин, розділених пропуском, наприклад:

```
p strong {...}
ul .hit {...}
.footer .menu a {...}
```

Такі селектори називають **контекстними** або **вкладеними**. Їх використовують для того, щоб застосувати стилі до елемента, тільки якщо він вкладений в потрібний елемент.

Наприклад, селектор `.menu a` спрацює для посилання `a` тільки в тому випадку, якщо вона розташована всередині елемента з класом `.menu`.

Читати їх найпростіше справа наліво:

```
/* Вибрати все теги strong всередині тегів p */
p strong {...}
```

```
/* Вибрати всі елементи з класом .hit всередині тегів ul */
ul .hit {...}
```

```
/* Вибрати всі посилання всередині елементів з класом .menu,  
   які лежать всередині елементів з класом .footer */  
.footer .menu a {...}
```

Таким чином, можна задавати елементам різні стилі в залежності від їх контексту. Якщо посилання розташована всередині меню, зробити її більше, а якщо всередині основного тексту, то задати їй потрібний колір.

Сусідні селектори

Контекстні селектори використовуються для вкладених один в одного елементів, а **сусідні** – для розташованих поруч. Наприклад, теги `` в списку є сусідніми по відношенню один до одного і вкладеними в тег ``.

Сусідні селектори записуються за допомогою знаку `+`. Наприклад:

```
селектор1 + селектор2
```

Стилі застосуються до елемента, відповідному до селектор2, тільки якщо відразу перед ним розташований елемент, який підходить до селектор1. Наприклад є два елементи списку:

```
<li class = "hit"> </ li>  
<li class = "miss"> </ li>
```

Селектор `.hit + .miss` застосує стилі до елемента з класом `miss`, так як перед ним є елемент з класом `hit`.

Селектор `.hit + li` теж застосує стилі до елемента з класом `miss`, а селектор `.miss + .hit` не спрацює.

Дочірні селектори

Нащадком називаються будь-які елементи, розташовані всередині батьківського елемента. А **доірніми елементами** називаються найближчі нащадки. Наприклад:

```
<ul>  
  <li> <span> ... </ span> </li>  
  <li> <span> ... </ span> </li>
```


Елементи та є дочірніми елементами і нащадками, а – нащадки, але не дочірні елементи.

Контекстні селектори впливають на всіх нащадків, що не завжди зручно. Іноді необхідно задати стилі тільки для дочірніх елементів. Особливо це корисно при роботі з багаторівневими списками.

Для цього існує дочірній селектор, в якому використовується символ >. Наприклад:

```
ul > li { ... }  
ul > li> span { ... }
```

Псевдокласи

Псевдокласи – це доповнення до звичайних селекторам, які роблять їх ще точнішими та потужнішими. Псевдоклас додається до селектора с допомогою символу «:» без пропусків. Наприклад:

```
a:visited {...}  
li:last-child {...}  
.alert:hover {...}
```

Переліку основних псевдокласів¹:

- **:first-child** – дозволяє вибрати перший дочірній елемент;
- **:last-child** – дозволяє вибрати останній дочірній елемент;
- **:link** – дозволяє вибрати ще не відвідані посилання;
- **:visited** – дозволяє вибрати відвідані посилання;
- **:active** – дозволяє вибрати активні посилання (кнопка миші затиснута);
- **:hover** – дозволяє вибрати елемент, коли на нього наведений курсор миші;
- **:nth-child** – використовується для додавання стилю до елементів на основі нумерації в дереві елементів;

¹ Повний перелік та приклади використання псевдокласів
<http://htmlbook.ru/css>

- **:focus** – дозволяє вибрати елемент, який в даний момент у фокусі;

Псевдоелементи

Псевдоелементи – це особливий вид селекторів, які дозволяють працювати не над самим елементом, а над його окремою частиною.

Перелік основних псевдоелементів:

- **:first-letter** – стиль першої літери текстового блоку;
- **:first-line** – стиль першого рядка текстового блоку;
- **:after** – додає вміст після елемента;
- **:before** – додає вміст до елемента;
- **::selection** – стиль виділеного користувачем тексту.

Так само як і у випадку з псевдокласів, псевдоелементи використовуються згідно наступного синтаксису:

```
p:first-letter {  
    color: #ff0000  
}
```

Псевдоелементи **after** та **before** призначені для "врізки" в сторінку сайту контенту, який спочатку відсутній в HTML документі. Вставляється зміст перед (:before) або після (:after) будь-якого елемента за допомогою властивості content, яке власне і визначає вміст для вставки. Наприклад:

```
p:after {  
    content: "Кінець абзацу!";  
}
```

Після кожного параграфа буде додаватися напис: "Кінець абзацу!".

Селектори атрибутів

Селектори атрибутів – це селектори, які дозволяють вибирати елементи з будь-яких атрибутів.

Найчастіше такі селектори використовуються при роботі з формами, так як поля форм мають атрибут type з різними значеннями.

Селектори атрибутів записуються з використанням квадратних дужок:

елемент [атрибут].

Наприклад:

```
input [checked] {...}  
input [type = "text"] {...}
```

Перший селектор вибере поля форми, у яких є атрибут *checked*, другий селектор вибере поля форми, у яких атрибут *type* має значення *text*.

Селектор по id

Існує ще один HTML-атрибут, для якого існує спеціальний селектор. Цей атрибут *id* (ідентифікатор), а селектор записується за допомогою символу *#*, наприклад

```
#some-id {...}
```

На значення *id* поширюються ті ж обмеження, що і на ім'я класу. Крім того *id* повинен бути унікальним на сторінці.

Використання селекторів такого типу при оформленні вважається поганою практикою.

Спадкування

Спадкування – механізм, за допомогою якого значення властивостей елемента-батька передаються його елементам-нащадкам.

Стили, присвоєні деякому елементу, успадковуються всіма нащадками (вкладеними елементами), якщо вони не перевизначені явно. Наприклад, розмір шрифту і його колір досить застосувати до тегу *body*, щоб всі елементи всередині мали ті ж властивості.

Спадкування дозволяє скоротити розмір таблиці стилів, але якщо стилів багато, то відстежити який батьківський елемент встановив деякий властивість, стає складніше.

До спадкоємних властивостей відносяться в першу чергу властивості, що визначають параметри відображення тексту:

`font-size, font-family, font-style, font-weight, color, text-align, text-transform, text-indent, line-height, letter-spacing, word-spacing, white-space, direction.`

Також до спадкоємною властивостей відносяться `list-style, cursor, visibility, border-collapse` і деякі інші. Але вони використовуються значно рідше.

Всі інші властивості не успадковуються. Як правило, це параметри позиціонування, розмірів, відступів, фону, рамок:

`background, border, padding, margin, width, height, position`
Повний перелік властивостей обох типів наведено у стандарті CSS.

Каскадність

Каскадність – набір правил, які визначають, які саме стильові властивості елементів веб-сторінок будуть застосовані до конкретних елементів. Або іншими словами до одного і того ж елементу може застосовуватися кілька CSS-правил (наборів CSS-властивостей). Серед цих властивостей можуть бути і конфліктуючі між собою. Тому існують інструкції, які визначають, яким буде фінальний набір властивостей елемента.

Стили для елемента можуть бути визначені в декількох місцях:

- внутрішні стилі – вказані у атрибуті `style`;
- глобальні стилі – стилі визначені тегамі `<style>` у документі;
- пов'язані стилі – стилі, що підключаються як зовнішні файли;
- стилі визначені браузером.

Браузер знаходить всі CSS-правила, що зачіпають даний елемент, а потім комбінує їх і отримує підсумковий список властивостей для цього елемента. Комбінування властивостей проводиться за чіткими правилами, які спираються на пріоритетність та специфічність, порядок вихідного коду.

Порядок пріоритетності правил невступний:

1. правила визначені атрибутом `style`;
2. правила визначені селектором по `id`;
3. правила визначені селектором класу, псевдокласу та атрибута;
4. правила визначені селектором тега та псевдоелементи.

Але у разі контекстних та дочерніх селекторів, псевдокласів або псевдоелементів пріоритетність визначається з урахуванням специфічності селекторів.

Всі 4 правила пріоритетності зводяться в одну систему a-b-c-d (де a - найвищий пріоритет) і утворюють специфічність.

Принцип побудови такої системи специфічності приведено у наступній таблиці:

Селектор	Специфічність a-b-c-d	Правило №
*	0-0-0-0	-
li	0-0-0-1	4
li:first-line	0-0-0-2	4
ul li	0-0-0-2	4
ul ol+li	0-0-0-3	4
form + *[type=text]	0-0-1-1	3, 4
table tr td.second	0-0-1-3	3, 4
h2.block.title.	0-0-2-1	3, 4
#xyz	0-1-0-0	2
style=" "	1-0-0-0	1

Завдання

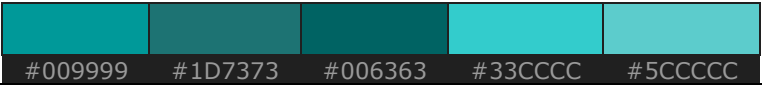
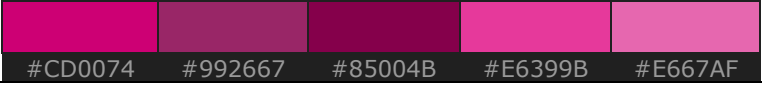
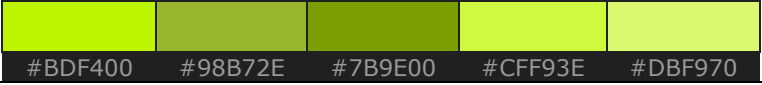
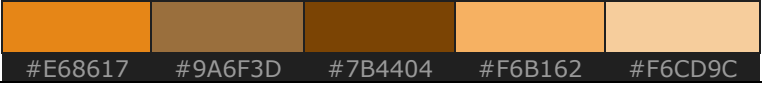
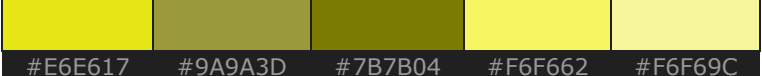
1. У проєкті з лабораторної роботи №3 налаштувати роботу ладерів для завантаження *.css та *.scss файлів та додати плагін MiniCssExtractPlugin.
2. У проєкті створити директорію scss та файл index.scss.
3. Створити (завантажити у мережі) файл reset.css² або normalize.css³ для скидання стандартних стилів браузера.
4. Імпортувати стилі з попереднього пункту до файлу index.scss.

² <http://habrahabr.ru/post/45296/>

³ <https://necolas.github.io/normalize.css/>

5. Ознайомитися з методологією іменування класів елементів ВЕМ⁴.
6. Зверстати сторінки (створити необхідні файли css/scss та правила css для форматування тексту, заголовків, посилань і т.д) з лабораторної роботи №3 за шаблоном та кольорами визначених варіантом палітри⁵ (бажано з використання ВЕМ).
7. Використовуючи параметри CSS3 для роботи з градієнтами, встановити фоновий колір сторінок сайту у вигляді градієнта використовуючи кольори палітри.
8. Зафіксувати ширину контенту сайту та вирівняти його по центру. Встановити відступи та поля (додати для цього необхідні теги у всіх сторінках).
9. Результати виконання лабораторної роботи розташувати у власному репозиторії будь якої відкритої системи контролю у директорії lab5 та розмістити посилання на роботу на публічній github або bitbucket сторінці.
10. Предметами захисту роботи є вихідний текст у репозиторії системи контролю версій та загальнодоступний ресурс, що відображає результат виконання лабораторної роботи.

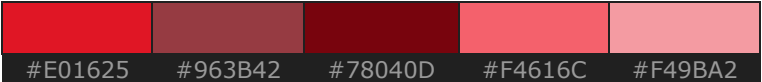
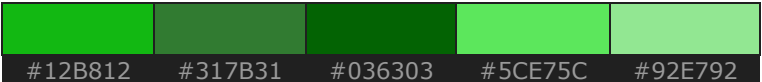
Варіанти палітри кольорів

№	Палітра	Посилання на тему
1	 #009999 #1D7373 #006363 #33CCCC #5CCCCC	https://colorscheme.ru/#3i11Tw0w0w0w0
2	 #CD0074 #992667 #85004B #E6399B #E667AF	https://colorscheme.ru/#5a11Tw0w0w0w0
3	 #BDF400 #98B72E #7B9E00 #CFF93E #DBF970	https://colorscheme.ru/#2c11Tw0w0w0w0
4	 #E68617 #9A6F3D #7B4404 #F6B162 #F6CD9C	https://colorscheme.ru/#0D11WsOsOFFfFf
5	 #E6E617 #9A9A3D #7B7B04 #F6F662 #F6F69C	https://colorscheme.ru/#1T11TsOsOFFfFf

⁴ <https://ru.bem.info/methodology/>

⁵ За бажанням виконавця можливо змінювати параметри контрастності та насичення кольорів палітри

6	 #49C514 #4C8434 #226A04 #88EB5D #AFEB95	https://colorscheme.ru/#1T11TsOsOFfFf
7	 #E63B17 #9A4D3D #7B1904 #F67C62 #F6AC9C	https://colorscheme.ru/#0811TsOsOFfFf
8	 #185C94 #2B4A63 #042E50 #60A3DB #90B9DB	https://colorscheme.ru/#3y11TsOsOFfFf
9	 #BE008A #8F2471 #7C005A #DF38B1 #DF64BD	https://colorscheme.ru/#5311Tw0w0w0w0
10	 #EA0037 #B02C4B #980023 #F53D68 #F56E8D	https://colorscheme.ru/#5t11Tw0w0w0w0
11	 #FF2800 #BF4630 #A61A00 #FF5D40 #FF8973	https://colorscheme.ru/#0711Tw0w0w0w0
12	 #FF5900 #BF6230 #A63A00 #FF8240 #FFA473	https://colorscheme.ru/#0k11Tw0w0w0w0
13	 #FF9F00 #BF8930 #A66800 #FFB740 #FFCA73	https://colorscheme.ru/#0Q11Tw0w0w0w0
14	 #FFC900 #BFA130 #A68200 #FFD640 #FFE173	https://colorscheme.ru/#1i11Tw0w0w0w0
15	 #BDF400 #98B72E #7B9E00 #CFF93E #DBF970	https://colorscheme.ru/#2c11Tw0w0w0w0
16	 #009999 #1D7373 #006363 #33CCCC #5CCCCC	https://colorscheme.ru/#3i11Tw0w0w0w0
17	 #133CAC #2B4281 #062270 #476DD5 #6D89D5	https://colorscheme.ru/#3N11Tw0w0w0w0
18	 #7E07A9 #67237F #52026E #AC3BD4 #B764D4	https://colorscheme.ru/#4M11Tw0w0w0w0

19	 #E01625 #963B42 #78040D #F4616C #F49BA2	https://colorscheme.ru/#5A11TsOs OFFFf
20	 #12B812 #317B31 #036303 #5CE75C #92E792	https://colorscheme.ru/#2P11TsOs OFFFf

Контрольні питання

1. Яке основне призначення каскадної таблиці стилів?
2. Для чого необхідні контекстні селектори?
3. Для чого призначені псевдоелементи?
4. Як працює механізм спадкування CSS?
5. Як визначити специфічність селектора?

Для нотаток: