Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

О. Г. Сузікова

ВЕБ ТЕХНОЛОГІЇ В ІНФОРМАЦІЙНОМУ ОБМІНІ

Методичні рекомендації
до практичних занять для здобувачів вищої освіти першого (бакалаврського) рівня з дисципліни «Обробка, зберігання та передача даних в сучасних інформаційних технологіях» за спеціальністю «Прикладна математика»

У 3 частинах

Частина 2: CSS

Електронний ресурс

Харків – 2025

УДК:004.439 (072)
С 89

Рецензенти:

Ю. В. Ромашов – доктор технічних наук, доцент, професор кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки Харківського національного університету радіоелектроніки;

С. М. Севидов – кандидат фіз.-мат. наук, доцент кафедри математичного моделювання та аналізу даних ННІ комп'ютерних наук та штучного інтелекту Харківського національного університету імені В. Н. Каразіна.

Затверджено до розміщення в мережі Інтернет рішенням Науково-методичної ради Харківського національного університету імені В. Н. Каразіна (протокол № 11 від 25 червня 2025 року)

Сузікова О. Г.

С 89 Веб технології в інформаційному обміні : методичні рекомендації до практичних занять для здобувачів вищої освіти першого (бакалаврського) рівня з дисципліни «Обробка, зберігання та передача даних в сучасних інформаційних технологіях» за спеціальністю «Прикладна математика». У 3 ч. Частина 2: CSS [Електронний ресурс] / О. Г. Сузікова. – Харків : ХНУ імені В. Н. Каразіна, 2025. – (PDF 32 с.)

Методичні рекомендації містять початкові відомості про CSS. До кожного розділу наведені питання для самоконтролю та практичні завдання.

Методичні рекомендації призначені для здобувачів вищої освіти першого (бакалаврського) рівня, які вивчають дисципліну «Обробка, зберігання та передача даних в сучасних інформаційних технологіях».

УДК:004.439 (072)

ЗМІСТ

Вступ

Каскадні таблиці стилів (CSS) — це один із основних інструментів веб-розробки, що відповідає за вигляд та оформлення веб-сторінок, написаних мовою HTML. Завдяки CSS можна змінювати кольори, шрифти, розміри елементів, створювати адаптивний дизайн і навіть анімації. Сучасні сайти не обходяться без цього потужного інструменту, який допомагає зробити їх естетично привабливими та зручними для користувачів.

Ці методичні рекомендації допоможуть вам освоїти CSS на практиці від основ, що складуть базу для засвоєння до більш просунутих технік. Ми розглянемо базові принципи стилізації, селектори, роботу з макетами, додавання стилю у документ та валідацію стилів. Матеріал подано у зрозумілій формі, з прикладами та практичними завданнями, що допоможуть вам закріпити отримані знання.

Методичні рекомендації містять початкові відомості про каскадні таблиці стилів. До кожного розділу наведені питання для самоконтролю та практичні завдання. Рекомендуємо обов’язково виконувати ці завдання, щоб швидше освоїтися у практичному застосуванні CSS.

1. Контекст

CSS Cascading Style Sheets (або каскадні таблиці стилів) – це набір стилів, де кожен стиль – це певне правило оформлення гіпертекстового документу. Тобто стилі – це набір правил оформлення і форматування документу або документів, який може бути застосований до елементів сторінки або сторінок послідовно. Застосовуючи CSS, ви можете один раз описати властивості елементів і визначити цей опис як стиль, а надалі просто вказувати, що сторінка, яку ви бажаєте оформити відповідним чином, повинна мати властивості стилю, описаного у документі CSS.

CSS підтримує таблиці стилів для конкретних носіїв інформації. Використання стилів не обмежується тільки зовнішнім виглядом сторінок, програмісти можуть адаптувати подання своїх документів до візуальних браузерів, слухових пристроїв, принтерів, брайлівських пристроїв, кишенькових пристроїв і т.д.

Каскадні таблиці стилів описують правила форматування елементів за допомогою властивостей і допустимих значень цих властивостей. Для кожного елемента можна використовувати обмежений набір властивостей, інші властивості не будуть чинити на нього ніякого впливу.

Іноді CSS (Cascading Style Sheets) описують як своєрідну мову стилів, яка використовується для опису вигляду і форматування різних веб-сторінок, написаних мовою розмітки HTML або XHTML. CSS дозволяє відокремити структуру веб-сторінки від її візуального подання, що полегшує процес розробки та підтримки веб-сайтів.

CSS працює за принципом каскадування, де стилі застосовуються зверху вниз, і останні правила мають пріоритет. Це дозволяє легко створювати складні стилі за допомогою простих декларацій, які можуть перезаписувати або доповнювати одна одну.

Основна структура CSS включає селектори, властивості та значення. Селектори визначають, до яких елементів HTML буде застосовано стиль, а властивості та значення визначають, які стилі будуть застосовані.

Каскадування – це механізм, який керує кінцевим результатом в ситуації, коли до одного елементу застосовуються різні CSS-правила.

```
body {
   background-color: #f0f0f0;
   font-family: Arial, sans-serif;
}

h1 {
   color: #333;
}
```

Цей код надає браузеру інформацію, що тіло документа треба подати шрифтом з сімейства Arial, з іменем sans-serif та на тлі дуже світлого сірого кольору, що має спеціальну назву «антибліковий білий». Заголовки першого рівня треба подати темно-сірим кольором.

Коли почали вживати мову гіпертексту, в стандартному HTML для присвоєння якому-небудь елементу певних властивостей (колір, розмір, положення на сторінці тощо) доводилося щоразу описувати ці властивості. Якщо сторінок було багато, то надання сторінкам спільного стильового оздоблення могло займати багато часу, якщо працювати окремо над кожною сторінкою. Тому наступним кроком було рознесення структури та змісту документу та його оздоблення. Так з’явився CSS.

Тож CSS був створений для вирішення проблеми змішування структури та стилю в HTML-документах. До появи CSS, стилі застосовувалися безпосередньо в HTML-коді, що робило його незручним, громіздким і важким для підтримки.

Ідея відокремлення стилю від змісту була вперше запропонована Хоконом Віум Лі (Håkon Wium Lie) у 1994 році, коли він працював в CERN разом з Тімом Бернерсом-Лі (Tim Berners-Lee), творцем WWW.

Стандарти CSS:

- Перший офіційний стандарт CSS був прийнятий World Wide Web Consortium (W3C) у 1996 році. Це був CSS1, який включав базові можливості для стилізації веб-сторінок.
- Згодом стандарт розвивався, і в 1998 році був представлений CSS2, що включав більше можливостей для макетування, таких як позиціонування і медіа-типи.
- Найсучаснішою версією є CSS3, яка була випущена у 1999 році та досі розвивається. CSS3 включає нові модулі, які забезпечують ще більшу гнучкість і потужність у стилізації веб-документів.

Що таке стиль?

Властивість стилю – це параметр стильового оформлення документа, що визначається специфікацією CSS.

HTML-документ – це деревоподібна ієрархічна структура, яка має вкладені елементі – багато вкладених один в один елементів (наприклад, елементи head, body, p, h1, h2 та ін.). Для придання документу певного вигляду для кожного тегу HTML-документа передбачено певний набір властивостей, що записуються у вигляді атрибутів та їх значень, які вказують у відкриваючому тегу. Серед усіх припустимих властивостей тегу є такі, які безпосередньо пов’язані з оздобленням документу, наприклад, гарнітура або колір шрифту конкретного абзаца, колір фону документа, розміри малюнка, це саме ті (стильові) властивості, які

можливо задати за допомогою CSS. Саме такі властивості і називаються стильовими властивостями (або властивостями стилю).

Хто і як розвиває CSS?

CSS розробляється і підтримується консорціумом World Wide Web Consortium (W3C), міжнародною організацією, яка займається розробкою стандартів для Всесвітньої павутини. Група W3C CSS Working Group складається з представників різних компаній, організацій та незалежних експертів, які спільно працюють над розвитком стандарту. Процес розвитку CSS включає обговорення пропозицій, створення чернеток, тестування і фінальне затвердження специфікацій. Це забезпечує, що стандарт відповідає потребам індустрії і залишається актуальним в умовах постійного розвитку технологій.

Роль CSS у веб-розробці.

CSS відіграє ключову роль у веб-розробці, надаючи засоби для створення естетично привабливих та функціонально ефективних веб-сайтів. Завдяки CSS розробники можуть контролювати такі аспекти дизайну, як кольори, шрифти, інтервали між елементами, розташування на сторінці, анімації та багато іншого. CSS є невід'ємною частиною сучасного веб-розроблення, забезпечуючи можливість створення багатих і інтерактивних веб-додатків, які виглядають і працюють чудово на будь-якому пристрої.

Основні переваги використання CSS включають:

- ✓ Відокремлення стилю від змісту: CSS дозволяє зберігати стилі окремо від HTML-коду, що полегшує обслуговування та оновлення веб-сайтів.

- ✓ Зменшення розміру HTML-документів: замість того, щоб дублювати стилі в кожному HTML-елементі, стилі можна визначити один раз у CSS-файлі і застосовувати їх до кількох елементів.
- ✓ Підтримка адаптивного дизайну: CSS дозволяє створювати веб-сайти, які автоматично адаптуються до різних пристроїв і розмірів екранів.
- ✓ Покращення продуктивності: зовнішні CSS-файли можуть кешуватися браузерами, що зменшує кількість завантажень і покращує швидкість завантаження сторінок.

2. Робота з каскадними таблицями стилів

2.1. Як влаштовані стилі CSS

Як ми відмітили раніше, каскадні таблиці стилів (CSS) використовуються для визначення зовнішнього вигляду веб-сторінок. Вони дозволяють розділити структуру HTML-коду і його візуальне оформлення, що робить розробку більш гнучкою та ефективною.

Основні принципи роботи CSS ґрунтуються на правилах, селекторах, властивостях та значеннях.

CSS-код складається з окремих правил, кожне з яких визначає стиль для певного елемента або групи елементів. Основна структура правила CSS виглядає так:

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

Наприклад, щоб змінити колір тексту всіх заголовків <h2> на синій, використовується таке правило:

```
h2 {
   color: blue;
}
```

2.2. Селектори в CSS

Селектори визначають, до яких HTML-елементів застосовуватиметься певний стиль.

Основні типи селекторів:

- ✓ **Тегові селектори** — застосовуються до всіх елементів певного типу (h1, p, div).

```
p {
   font-size: 16px;
}
```

Цей код наказує браузеру надрукувати всі параграфи шрифтом розміру 16 пікселів.

- ✓ **Класові селектори** — визначають стилі для елементів із певним класом (.button, .container)

```
.button {
   background-color: blue;
   color: white;
padding: 10px 20px;
}
```

Цей код задає фоновий колір, колір тексту та відступи.

- ✓ **ID-селектори** – задають стилі для елементів з унікальним ідентифікатором (#header, #footer).

```
#header {
    background-color: red;
    padding: 20px;
text-align: center;
}
```

Цей код задає фоновий колір, відступи та тип вирівнювання.

- ✓ **Комбіновані селектори** – дозволяють комбінувати стилі для конкретних елементів (наприклад, div p застосує стиль до всіх<p>, що знаходяться всередині).

```
div p {
   color: darkgray;
   font-style: italic;
}
```

Цей код задає колір та стиль шрифту.

2.3. Властивості та значення

CSS містить сотні властивостей, які змінюють вигляд елементів. Основні групи властивостей:

- ✓ **Текстові** (color, font-size, text-align)

```
.container {
   color: white;
   }
```

- ✓ **Фонові** (background-color, background-image)

```
div  p{
   background-color: red;
   }
```

- ✓ **Розміщення елементів** (margin, padding, border)

```
.button {
   padding: 20px;
}
```

- ✓ **Розміри та позиціонування** (width, height, position)

```
div p {
   heigh: 24px;
}
```

Наприклад, такі правила задають червоний фон, білий текст і відступи для блоку та висоту елемента.

```
.container {
```

```
   background-color: red;
   color: white;
   padding: 20px;
}
```

Корисні властивості. Вирівнювання тексту.

Як Ви пам'ятаєте з теми HTML, для того щоб вирівняти текст, наприклад, по центру екрана, до тегу, що містить в собі текст, було застосовано атрибут align (вирівнювання) і одне з його можливих значень:

- left – Вирівняти текст по лівому краю елемента (за замовчуванням).
- right – Вирівняти текст по правому краю.
- center – Вирівняти текст по центру.
- justify – Вирівняти текст по обох краях.

Запис мав такий вигляд:

```
<p align = "center"> текст по центру </ p>
```

В CSS це завдання бере на себе властивість text-align, яка вирівнює текст щодо елемента батька (наприклад, блоку div) або ж вікна браузера.

Властивість text-align (так само як і html атрибут align) має такі значення:

- left – Вирівняти текст по лівому краю елемента (за замовчуванням).
- right – Вирівняти текст по правому краю.
- center – Вирівняти текст по центру.
- justify – Вирівняти текст по обох краях.

Оформлення тексту.

Властивість text-decoration дозволяє декорувати текст, присвоївши йому одне або кілька значень з нижче представлених варіантів оформлення тексту.
Можливі значення:

- blink – Текст буде блимати.
- line-through – Робить текст перекресленим.
- overline – Надкреслення тексту.
- underline – Підкреслення тексту.
- none – Текст без оформлення.

Відступ першого рядка.

Властивість text-indent – задає відступ першого рядка в текстовому блоці з лівого боку, простіше кажучи, робить "новий рядок".
Відстань від лівого краю вікна браузера або ж елемента батька (блоку в який поміщений блок з текстом) може бути задано у відсотках від ширини вікна браузера або ж одиницях вимірювання прийнятих в CSS.

Трансформація тексту

Властивість text-transform трансформує символи в зазначеному текстовому блоці, роблячи їх великими або прописними за одним з правил в залежності від присвоєного значення даній властивості.

Може мати наступні значення:

- none – текст відображається без будь-яких змін (цей параметр використовується за замовчуванням;
- capitalize – кожне слово в тексті починається з великого символу;
- lowercase – усі символи перетворюються в символи нижнього регістру;
- uppercase – усі символи перетворюються в символи верхнього регістру.

Вертикальне вирівнювання

Вертикальне вирівнювання тексту в рядку встановлює властивість vertical-align.

Можливі значення властивості vertical-align:

- baseline – вирівнює базову лінію елемента за базовою лінією елемента-батька.
- bottom – вирівнює елемент по нижній частині рядка.
- middle – вирівнює середину елемента за базовою лінією батька і додає половину висоти батьківського елементу.
- sub – нижній індекс (розмір шрифту не змінюється).
- super – верхній індекс (розмір шрифту не змінюється).
- text-bottom – нижня межа елемента вирівнюється по нижньому краю рядка.
- text-top – для тексту: верхня межа текстового елемента вирівнюється по верхньому краю рядка.
- top – вирівнює будь-який елемент по верхній частині рядка.

Пробіли і перенесення рядка.

Текст, набраний у будь-якому текстовому редакторі, за замовчуванням виводиться браузерами на екран у вигляді суцільного тексту: переноси рядків розставляються автоматично і прибираються зайві (більше одного) пробіли між символами.

Властивість white-space імітує роботу тега <pre>, визначаючи, показувати чи ні пробіли між символами, якщо їх більше ніж один, а також дозволяє або забороняє перенесення рядка.

Може мати наступні значення:

- normal – текст виводиться як завжди (зайві прогалини прибираються), переноси рядків визначаються автоматично (за замовчуванням).
- nowrap – забороняє автоматичне перенесення рядка.
- pre – показує текст в тому вигляді, в якому він був набраний, пробіли і переноси рядків не видаляються.

Відстань між словами.

Властивість word-spacing задає відстань між словами (тобто групами літер, що не розділені пробілами) в рядку.

Може мати наступні значення:

- normal – нормальна відстань (за замовчуванням)
- px – відстань задається в пікселях або будь-яких інших одиницях виміру, прийнятих в CSS.

Аналогічно задається властивість letter-spacing, яке визначає відстань між символами в тексті. І так само, як і word-spacing, може бути задана такими самими значеннями.

Інтерліньяж – це відстань між рядками тексту.

Відстань між рядками тексту можна задати за допомогою властивості line-height. Значення:

- normal – норма (за замовчуванням).
- % – відсотки; за сто відсотків береться висота шрифту
- 0.5 – множник; може бути використано будь-яке число більше нуля. Так наприклад, множник 0.5 буде дорівнювати половинному міжрядковому інтервалу, а множник 2 – подвійному.
- px – пікселі та будь-які інші одиниці виміру, прийняті в CSS.

Всі ці властивості будуть вам корисні при створенні однотипних сторінок.

2.4. Принцип каскадності та успадкування

CSS працює за принципом каскаду – правила можуть перекривати одне одне залежно від специфічності. Також деякі властивості успадковуються (наприклад, color та font-family), а інші – ні (наприклад, margin, padding). Використовуючи ці принципи, можна створювати стильні та гнучкі веб-інтерфейси, що легко змінюються та адаптуються під різні пристрої.

```
p {
color: black; /* Базовий стиль для всіх абзаців */
}

.special {
color: blue; /* Перевизначає колір тексту для елементів з класом .special */
}

#unique {
color: red; /* Перевизначає всі попередні правила для елемента з ID unique
*/
}
```

В результаті впровадження стилю будемо бачити наступне.

```
<p>Цей текст буде чорним.</p>
<p class="special">Цей текст буде синім.</p>
<p id="unique" class="special">Цей текст буде червоним.</p>
```

У цьому прикладі текст останнього <p> буде червоним, оскільки ID має вищий пріоритет, ніж клас або тегові селектори.

2.5. **Питання для самоконтролю**:

Що таке css?

Як побудовано css правило?

Що таке селектор?

Які види селекторів ви знаєте?

Атрибути та їх значення?

Як проявляється каскадність css?

2.6. Практичне завдання

Описати правила стилю css зі вживанням селекторів (4 типи), властивостей та значень. Створіть HTML-сторінку, де є заголовок та декілька абзаців, оформлених різними стилями. Використовуйте каскадність CSS, щоб змінити колір тексту залежно від селектора.

```
<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Каскадність CSS</title>
<style>
	p {
color: black; /* Загальний стиль для всіх абзаців */
	}
	.highlight {
color: green; /* Перевизначення кольору для елементів з класом .highlight
*/
	}
	#special {
color: orange; /* Перевизначення кольору для елемента з ID special */
	}
</style>
</head>
<body>
<h1>Демонстрація каскадності CSS</h1>
<p>Цей текст буде чорним.</p>
<p class="highlight">Цей текст буде зеленим.</p>
<p id="special" class="highlight">Цей текст буде оранжевим.</p>
```

```
</body>
</html>
```

В цьому прикладі очікуваний результат: перший абзац буде чорним, другий – зеленим, а третій – оранжевим, оскільки стиль ID має вищий пріоритет, ніж клас.

3. Експорт CSS стилів

3.1. Три способи додавання CSS стилю до HTML документу

Отже, перелічимо три способи, за допомоги яких можна впровадити CSS в HTML-документ, тобто зв'язати стильовий опис елемента з самим елементом або з HTML тегом.

1. Написати стильовий опис безпосередньо в самому елементі. Такий спосіб хороший лише в тому випадку, якщо такий елемент – один в HTML документі, який потребує окремого стильового опису.
2. Написати стильовий опис для всіх ідентичних елементів HTML документа. Такий спосіб виправдовує себе, якщо стиль окремих сторінок принципово відрізняється від загального дизайну сайту.
3. Винести стильовий опис елементів HTML в окремий файл CSS. Це дозволить керувати дизайном всього сайту загалом, кожною сторінкою сайту, в якій вказано звернення до CSS файлу. Цей спосіб є найбільш ефективним використанням таблиці каскадних стилів.

3.2. Додавання до самого елемента через атрибут style.

Практично кожен HTML тег має атрибут style, який говорить про те, що до цього тегу застосовується певний стильовий опис.

```
<p style = ""> це параграф з індивідуальним стилем </p>
```

Все, що буде написано між лапками атрибута style, і буде стильовим описом для даного елемента, в даному випадку елемента <p>.

Наприклад:

```
<p style = "color: # ff0000; font-size: 12px"> це параграф з індивідуальним стилем </p>
```

В даному випадку ми вказали, що цей параграф повинен відображатися червоним кольором і мати розмір шрифту в 12 пікселів.

Тобто ми бачимо, як застосувати CSS до якогось HTML тегу. За таким же принципом можна вказати індивідуальний стиль практично для кожного HTML елемента.

Приклад застосування для різних елементів документа:

```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 Transitional // EN"
"Http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title> Атрибут style</title>
</ head>
<bodystyle = "background-color: # c5ffa0">
<h1 style = "color: # 0000ff; font-size: 18px"> Все про мене </ h1>
<p style = "color: # ff0000; font-size: 14px"> На цьому сайті Ви знайдете будь-яку інформацію про мене</ p>
```

```
<h2 style = "color: # 0000ff; font-size: 16px">Інтереси</ h2>
<p style = "color: # ff0000; font-size: 14px"> Вподобання, знання </ p>
<h2 style = "color: # 0000ff; font-size: 16px">Як мене знайти</ h2>
<p style = "color: # ff0000; font-size: 14px"> Контакти</ p>
</ body>
</ html>
```

Такий спосіб впровадження CSS хороший лише в тому випадку, якщо потрібно задати певний стиль невеликій кількості HTML-елементів.

3.3. Додавання стилю через тег <style> розділу head

Для того, щоб описати необхідні елементи одночасно на всій сторінці, до заголовку HTML-документа потрібно додати тег <style> </style> (зверніть увагу: він має таку саму назву, як атрибут), в якому описати елементи.

- ✓ Тег <style> додається до заголовка HTML-документа між тегами <head> і </head>.
- ✓ Атрибут тега <style> type – повідомляє браузеру, який синтаксис використовувати для правильної інтерпретації стилів.
- ✓ Для правильної інтерпретації браузерами CSS значення type (MIME тип даних) повинно дорівнювати text / css.
- ✓ Усередині тега <style> </style> мітиться оголошення стилів HTML-елементів за таким синтаксисом:

```
<head>
<style type="text/css">
елемент {атрибут: значення}
</style>
</head>
```

Приклад.

```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 Transitional // EN"
"Http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title> Ter style</title>
<style type = "text / css">
body {background-color: # c5ffa0}
h1 {color: # 0000ff; font-size: 18px}
h2 {color: # 0000ff; font-size: 16px}
p {color: # ff0000; font-size: 14px}
</style>
</head>
<body>
<h1>Все про мене</h1>
<p> На цьому сайті Ви знайдете будь-яку інформацію про мене </p>
<h2>Інтереси</h2>
<p> Вподобання, знання </p>
<h2> Як мене знайти </h2>
<p>Контакти</p>
</body>
</html>
```

Ми зробили нашу сторінку та отримали такий самий результат, що й за допомогою атрибуту style, але тепер ми не приписуємо кожному елементу стиль окремо, а винесли його в "головний" розділ документа, вказавши що всі заголовки <h1>, <h2> – будуть синіми а параграфи <p> – червоними.

При використанні такого підходу ми досягли економії нашої роботи та зменшили розмір документа за рахунок видалення всіх повторюваних стильових описів для кожного окремого елемента. Якщо наш документ складається з великої кількості сторінок та елементів – економія суттєва.

3.4. Експорт стилю з зовнішнього документу

Нарешті, ми розглянемо основний спосіб – як винести все, що стосується дизайну сайту, в окремий зовнішній файл.

Як створити CSS в окремому зовнішньому файлі? Наведемо приклад.

У блокноті напишемо, наприклад, щось на кшталт такого тексту:

```
body {background-color: # c5ffa0}
a {color: # 000060; font-weight: bold;}
a: hover {color: # ff0000; font-weight: bold; text-decoration: none}
h1 {color: # 0000ff; font-size: 18px}
h2 {color: # ff00ff; font-size: 16px}
p {color: # 600000; font-size: 14px}
```

Далі потрібно зберегти цей файл з розширенням .css (зазвичай файл зі стилями називають style.css). Це і є файл зі стильовим описом.

Тепер зробимо так, щоб сторінки нашого сайту використовували цей файл. Для цього призначений тег <link> (зв'язок). Тег <link> «зв'язує» HTML-документ з додатковими зовнішніми файлами, що забезпечують його належну роботу. Тег <link> є посиланням, призначеним для браузерів.

Тег <link> додається до заголовка HTML-документа між тегами <head> і </head>, оскільки він включає лише службову інформацію і не має виводитись на екран.

Тег <link> має такі атрибути:

- href – шлях до файлу.
- rel – визначає відносини між поточним документом і файлом, на який робиться посилання.
- shortcuticon – визначає, що підключається файл є іконкою.
- stylesheet – визначає, що підключається файл містить таблицю стилів.

- application / rss + xml – файл в форматі XML для опису стрічки новин.
- type – MIME тип даних підключається файлу.

Оскільки ми як зовнішній файл підключаємо каскадну таблицю стилів, то посилання має такий вигляд:

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

Детальніше, ми зробили наступне:

- Атрибуту rel присвоюємо значення stylesheet (оскільки підключаємо каскадну таблицю стилів).
- Вказуємо шлях до файлу css (це файл mystyle.css, він розташований там, де і документ HTML, в якому міститься дане посилання).
- Вказуємо, що даний файл текстовий і містить стильовий опис (вказуємо type = "text / css")

Нарешті, додамо цей рядок до заголовків сторінок нашого сайту і побачимо, що описані у файлі mystyle.css стилі впроваджені у всі сторінки.

3.5. Питання для самоконтролю

Які способи додавання стилю в гіпертекстовий документ ви можете навести?

Які переваги та обмеження кожного зі способів?

В яких випадках краще застосовувати кожен зі способів?

3.6. Практичне завдання

Оформити підготовлений структурований гіпертекст, представлений в файлах index.html, aboutme.html, відповідно до обраних вами стилів. Для виконання роботи можна використовувати документи html, підготовані в ході виконання завдань з теми «html».

1. Сформулювати ознаки потрібного стилю в термінах css.
2. Оформити підготовлений структурований гіпертекст, представлений в файлах index.html, aboutme.html відповідно до обраних вами стилів наступними шляхами:
    - За допомогою інлайнових стилів;
    - За допомогою глобальних стилів;
    - За допомогою зовнішнього файлу стилів.
3. Відповісти на питання, який зі способів буде оптимальним.

*Обов'язкове використання:

- Інтерліньяжу
- Міжсимвольної відстані.
- Відстані між словами.
- Прогалин і переносів рядка.
- Вертикального вирівнювання
- Трансформації тексту
- Відступу першого рядка.

- Оформлення тексту.
- Вирівнювання тексту.

Приклад виконання

файл mystyle.css

```
body {background-color: # c5ffa0}
a {color: # 000060; font-weight: bold;}
a: hover {color: # ff0000; font-weight: bold; text-decoration: none}
h1 {color: # 0000ff; font-size: 18px}
h2 {color: # ff00ff; font-size: 16px}
p {color: # 600000; font-size: 14px}
```

файл index.html

```
<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 Transitional // EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>каскадна таблиця стилів</ title>
<link rel = "stylesheet" href = "mystyle.css" type = "text / css">
</ head>
<body>
<h2>Меню: </ h2>
<a href="index.html">Все про мене</a>
<a href="aboutme.html"> Інтереси</a>
<a href="aboutme.html">Як мене знайти</a>
<br>
<h1> Все про мене</ h1>
<p> На цьому сайті Ви знайдете будь-яку інформацію про мене</ p>
</ body>
</ html>
```

файл aboutme.html

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

```
<head>
<title>каскадна таблиця стилів</ title>
<link rel = "stylesheet" href = "mystyle.css" type = "text / css">
</ head>
<body>
<h2>Меню: </ h2>
<a href="index.html">Все про мене. </a>
<a href="aboutme.html"> Інтереси</a>
<a href="aboutme1.html">Як мене знайти</a>
<br>
<h1>Інтереси</ h1>
<p>Вподобання,  знання</ p>
</body>
</html>
```

```
файл aboutme1.html

<! DOCTYPE HTML PUBLIC "- // W3C // DTD HTML 4.01 Transitional // EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>каскадна таблиця стилів</ title>
<link rel = "stylesheet" href = "mystyle.css" type = "text / css">
</head>
<body>
<h2>Меню: </ h2>
<a href="index.html">Все про мене. </a>
<a href="aboutme.html"> Інтереси</a>
<a href="aboutme1.html">Як мене знайти</a>
<br>
<h1> Як мене знайти</h1>
<p>Контакти</p>
</body>
</html>
```

Загальні рекомендації

- Використовуйте атрибут style для будь-якого елемента, якщо цей елемент – єдиний на всьому сайті з відмінним від інших елементів стилем.
- Використовуйте тег <style> зі стильовим описом у тому випадку, якщо сторінка повинна мати індивідуальний дизайн, принципово відмінний від інших сторінок сайту.
- У більшості випадків має сенс виносити каскадну таблицю стилів в окремий css файл.

4. Валідація стилів

4.1.Як валідувати стиль

Валідація (перевірка) коду CSS на відповідність стандартам має певні переваги і нічим не відрізняється від валідації HTML.

- ✓ Документ з CSS кодом буде вважатися дійсним в тому випадку, якщо він пройшов відповідну перевірку і не містить помилок.
- ✓ Якщо ви використовуєте таблиці стилів переважно в зовнішньому файлі, це суттєво полегшує перевірку коду, оскільки потрібно перевіряти лише один файл, а не кожну сторінку окремо.
- ✓ Якщо ви допустили помилки при написанні коду таблиці стилів, можна так само скористатися валідатором, для того, щоб не шукати помилки вручну.

Для перевірки CSS коду на помилки і відповідність стандартам можна скористатися валідатором CSS, який надає три способи перевірки:

- ✓ Перевірка по URL

Введіть URL веб-сторінки, яку хочете перевірити, і натисніть кнопку "перевірити". Після перевірки коду валідатором ви отримаєте одне з двох повідомлень – або про те, що все пройшло успішно і вас привітають з дійсним кодом, або про те, що в вашому css коді були виявлені помилки, які потрібно виправити.

- ✓ Перевірка завантаженого файлу

Цей спосіб дозволяє завантажити документ для перевірки на сервер. Щоб це зробити, натисніть кнопку «Огляд» і виберіть файл, який потрібно перевірити. Сервер автоматично розпізнає тип завантаження.

- ✓ Перевірка набраного коду

Цей спосіб чудово підходить для перевірки частини CSS-файлу. Потрібно скопіювати код в текстове поле валідатора й отримати результат перевірки.

4.2.Питання для самоконтролю

Як і для чого користуються валідатором?
Які засоби валідації css ви знаєте?
В чому специфіка кожного засобу?

4.3.Практичні завдання

Використовуючи матеріал попередньої роботи і валідатор, виконати наступне завдання:

1. Виконати усіма способами валідацію файлів HTML.
2. Виконати валідацію файлів CSS.

Список літератури

1. Інтернет технології. Методичні вказівки до лабораторних робіт для студентів напрямку підготовки 6.050802 «Електронні пристрої та системи». – К.: НТУУ “КПІ”, 2014. – 90 с.
2. Методичні вказівки і завдання до лабораторних робіт з курсу інформатики. Частина 2. Основи web-розробки: Навч.-метод. посіб. / Автори-укладачі: О.В. Присяжнюк, О.В. Рєзіна – Кропивницький: ЦДПУ імені В. Винниченка, 2021. – 42 с.
3. Пасічник О.Г., Пасічник О.В., Стеценко І.В. Основи веб-дизайну : навч. посіб. – К.: Вид. група BHV, 2009. – 336 с.

Інтернет-ресурси

https://www.w3schools.com/css/default.asp – офіційна документація

Електронне навчальне видання комбінованого використання
Можна використовувати в локальному та мережному режимі

Сузікова Олена Геннадіївна

ВЕБ ТЕХНОЛОГІЇ В ІНФОРМАЦІЙНОМУ ОБМІНІ

Методичні рекомендації
до практичних занять для здобувачів вищої освіти першого (бакалаврського) рівня з дисципліни «Обробка, зберігання та передача даних в сучасних інформаційних технологіях» за спеціальністю «Прикладна математика»

У 3 частинах

Частина 2: CSS

В авторській редакції

Підписано до розміщення 25.06.2025. Гарнітура Times New Roman.
Ум. друк. арк. 1,35. Обсяг 0,981. Зам. № 247/25.

Харківський національний університет імені В. Н. Каразіна,
61022, м. Харків, майдан Свободи, 4.
Свідоцтво суб'єкта видавничої справи ДК № 3367 від 13.01.2009.
Видавництво ХНУ імені В. Н. Каразіна