ОСНОВИ CSS

О.В.Ковтунець. Основи клієнтської розробки

Передумови появи CSS

HTML 1.0 не мав структур для відображення текстів якимось окремим способом. Не було жодних засобів керування зовнішнім виглядом інформації, була задача забезпечення доступності інформації для будь-яких пристроїв.

Microsoft і Netscape впровадили власні теги і «покращили» наявні для керування оформленням. Частина змін прижилась і пішла в маси.

HTML став набором несумісних тегів і розширень. Результатом виправлення ситуації став стандарт HTML 3.2, який «узаконив» зміни і усунув проблеми сумісності.

Результат версіонування HTML

Всі засоби і теги, які спрямовані керувати зовнішнім виглядом наданої інформації, є **атавізмом** попередніх версій HTML.

Стандарт HTML 3.2 лише виправив найсерйозніші недоліки попередніх версій. Справжнім вирішенням проблеми став стандарт HTML 4.0, в якому запропонували відділити описання структури HTML-документа від його оформлення.

Насправді – ще в SGML, на якому базувалась перша версія HTML, пропонувався окремий "файл стилів" документа.

☺

Отже, еволюційний розвиток стандартів HTML закономірно призвів до виникнення каскадних таблиць стилів.

Поява CSS

Концепцію каскадних таблиць стилів

https://www.w3.org/People/howcome/p/cascade.html

у 1994 р. описав Хокон Віум Лі – норвезький вчений та IT-спеціаліст консорціуму W3C.

Розробляв броузер Opera, захистив дисертацію на тему «Каскадні таблиці стилів», створив тест Acid2

http://acid2.acidtests.org/

на відповідність броузерів специфікаціям CSS v2.1.

Суть CSS

CSS (Cascading Style Sheets – каскадні таблиці стилів) – мова описання зовнішнього вигляду документа, створеного з допомогою мови розмітки.

Зазвичай CSS застосовують разом із HTML для задання кольорів, параметрів шрифтів, розміщення блоків та інших елементів візуалізації HTML-документів.

Призначення CSS

Розширення можливостей візуалізації документів в рамках декларативного характеру розмітки із збереженням контролю над формою представлення елементів HTML-розмітки.

Вирішення протиріч між точністю визначення розмірів елементів і точністю визначення розмірів блоків тексту.

Переваги використання CSS

- централізоване керування відображенням множини HTML-документів з допомогою однієї таблиці стилів;
- спрощений контроль зовнішнього вигляду HTML-документів;
- наявність розроблених дизайнерських рішень і технік;
- можливість використання різних стилів для одного документа, залежно від пристрою, з допомогою якого здійснюється доступ до HTML-документа.

BOM і DOM

Стандарт HTML 4.0 зафіксував крім CSS ще і об'єктну модель броузера (Browser Object Model – BOM), яка описує вміст HTML-документа (модель є набором об'єктів, що описують заданий вміст).

Оскільки BOM унікальна для кожного броузера, виникали проблеми з міжплатформенними застосуваннями.

Врешті-решт замість BOM придумали об'єктну модель документа (Document Object Model – DOM), яка описує стандарт представлення HTML-документа у вигляді набору об'єктів.

Контроль актуальної версії CSS

З 2007 р. існує періодично оновлюваний документ «CSS Snapshot» («знімок» стану CSS): https://w3.org/TR/css.

Якщо орієнтуватися на цей документ, то ми вже маємо приблизно 8 версію CSS. Насправді зараз досі актуальна версія CSS3, в яку постійно вносять правки і вставки.

«Знімки» CSS призначаються розробникам броузерів, а не веб-розробникам, і демонструють «готовність» самих «features», а не їхню підтримку на практиці.

Тому ми користуємося https://caniuse.com/

CSS1 (1996, 1999)

Спосіб відображення елемента на сторінці.

Обтікання елемента текстом, розміри елемента.

Зовнішні і внутрішні відступи елемента (margin, padding).

Вертикальне вирівнювання в таблицях.

Вирівнювання тексту, зображень, таблиць та інших елементів.

Стилі границь елементів.

Форматування списків.

Кольори (текст, фон, рамки тощо).

Параметри шрифтів (гарнітура, розмір, курсив, жирність).

Властивості тексту (міжсимвольний інтервал, відстань між словами, міжрядкові відступи).

CSS2 (1998)

CSS1 + :

Блочна верстка (абс., фікс. і відн. позиціювання), розміщення елементів на осі Z, напрямок тексту, вигляд курсора і лапок.

Видимі області елементів, відображення елементів, що виходять за межі заданих розмірів, мін. і макс. можливі розміри елемента.

Відстань між комірками таблиці, розміри елементів таблиць.

Стилі кожної межі елемента окремо.

Стилі для різних типів носіїв (монітор, принтер, КПК тощо).

Стилі для сторінкових носіїв – для парних/непарних сторінок.

Звукове оформлення контента (гучність, паузи тощо).

Генерований вміст – додає дані, яких немає у оригінальному HTML- документі, до чи після необхідного елемента.

CSS 2.1 (2009, 2011)

Перша серйозна ревізія специфікації:

виправлено помилки CSS2,

відкинуто не реалізовані особливості CSS2,

видалено фрагменти CSS2, які стануть застарілими в CSS3,

додано нові значення властивостей,

змінено окремі властивості під напором розробників броузерів.

W3C рекомендує CSS2.1 замість CSS2.

CSS3 (2007)

Зкруглені кути, зображення і градієнт в рамках елементів, тіні елементів, нестандартні шрифти, зміна розмірів блоків користувачем, розбиття тексту на колонки, фонові зображення, можливість створювати анімовані елементи без використання Javascript, поява змінних.

Загалом тепер це формальна мова, реалізована з допомогою мови розмітки. Наймасштабніша редакція порівняно з CSS1, CSS2 и CSS2.1. Специфікацію розбито на модулі, розробка і розвиток незалежні, все базується на CSS2.1 – тому ніяких CSS4 в подальшому не планується, всі оновлення ідуть в рамках CSS3. Остання правка 2019.

Порівняння CSS і HTML

<I>Текст курсивом!</I>

Текст курсивом!

<I STYLE="text-decoration:underline;
font-style:normal;">Текст курсивом?</I>

<u>Текст курсивом?</u>

Значення CSS для розробки

Процес розробки і супроводження веб-систем можна формалізувати і подати у вигляді списку дій:

- Визначення номенклатури сторінок
- Для різних типів сторінок розробка логічної структури
- Створення навігаційної карти веб-системи
- Розробка стилів відображення для стандартних компонентів сторінки
- Створення зображень, анімацій, скриптів, внесення текстів і графіки, генерація сторінок при звертанні до них

І знову про ієрархію

Елементи можуть вкладатися в інші елементи. Відносини між вкладеними елементами можуть бути батьківськими, дочірніми і сусідніми (або ж братними, сестринськими тощо).

Дерево документа – уявна деревовидна структура елементів у HTML-документі (синонім об’єктної моделі документа DOM).

Батьківський елемент – елемент, що містить в собі поточний елемент. В рядку ** <a>... ** контейнер ** ** є батьківським відносно **<a>**.

Предок – елемент на кілька рівнів вище, який містить в собі поточний елемент. В рядку ** <a> ** контейнер ** ** є предком **<a>**.

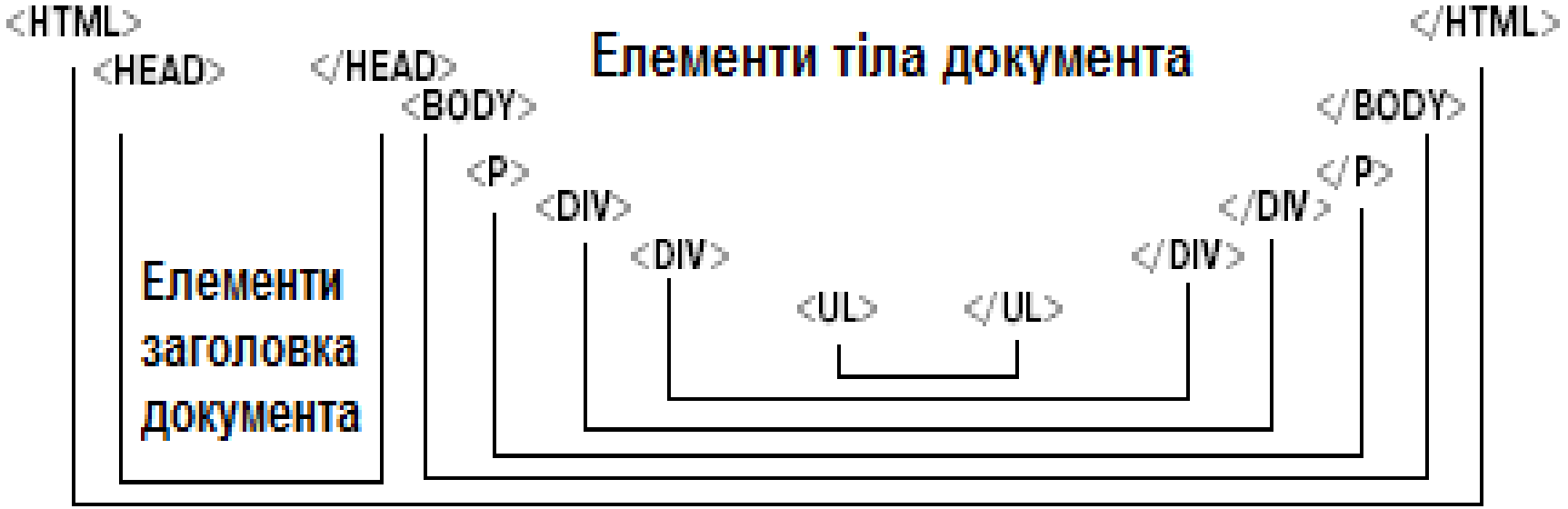
Дочірній елемент – елемент, що міститься в поточному елементі. В рядку **
 ** елемент **
** є дочірнім відносно ** **.

Потомок – елемент, що міститься в поточному елементі на кілька рівнів вкладеності нижче. В рядку **<body> <a> </body> <a>** є потомком **<body></body>**.

Сусідній елемент – елемент, який має спільного батьківського елемента з поточним. В рядку ** <a> <img> ** елементи **<img>** і **<a>** є сусідніми.

Ієрархія, наслідування та зміна стандартного стиля

Ієрархічність «таблиць стилів» спричиняє наслідування властивостей батьківського елемента дочірніми.



Задання CSS

<селектор> {

<атрибут1>: <значення атрибута1>;

<атрибут2>: <значення атрибута2>;

…

<атрибутN>: <значення атрибутаN>;

}

Селектор – формальний опис елемента (набору елементів), до якого треба задіяти інструкції стилю.

Атрибут і **значення атрибута** – інструкція стилю для елемента (не те саме, що атрибут тега).

Селектори відокремлюються пробілами або символами переносу рядка (не ';'!). В назвах селекторів та атрибутів не допускаються пробіли і переноси рядка.

Синтаксис

селектор1 [, селектор2 [, ...]] { атрибут1 : значення1; [атрибут2 : значення2; ...] }

АБО

селектор1 селектор2 [селектор3 ...] { атрибут1 : значення1; [атрибут2: значення2; ...] }

Селектором може бути а) ім’я елемента розмітки, б) ім’я класу, в) ідентифікатор об’єкта на сторінці

Атрибут визначає властивість елемента та значення цієї властивості

Селектор елемента

Селектор:	**селектор елемента**
Оформлення в CSS:	**P {color: red }**
Елемент в HTML:	** ... **

Стиль буде застосовано до всіх абзаців HTML-документа – тобто весь текст в контейнерах ** ** буде розмальований в червоне.

Селектор класу

Селектор: **селектор класу**

Оформлення в CSS: **.classname {color: red}**

Елемент в HTML:

** ... **

** ... **

** ... **

Стиль можна застосувати до будь-якого тега, який містить атрибут **class**, значення якого співпадає із іменем селектора класу.

Селектор ідентифікатора

Селектор:	**селектор ідентифікатора**
Оформлення в CSS:	**#itmRed { color: red }**
Елемент в HTML:	** ... **

Стиль буде задіяний до будь-якого тега, що містить атрибут **id**, значення якого співпадає з іменем селектора стилю. В рамках одного HTML-документа значення атрибута **id** має бути унікальним, тому фактично такий стиль можна задіяти тільки один раз для одного елемента HTML-документа.

Селектор дочірнього елемента

Селектор: **селектор дочірнього елемента**

Оформлення в CSS: **#itmRed>a { color: red}**

Елемент в HTML: **<p id=“itmRed"><a></p>**

Стиль буде застосовано до всіх контейнерів **<a>**, які є дочірніми для **<p id=“itmRed"></p>** і тільки для них.

Контекстний селектор

Селектор:	**контекстний селектор**
Оформлення в CSS:	**p a {color : red}**
Елемент в HTML:	** <a>... **

Стиль буде застосовано до тексту в рамках тега **<a>**, який іде за тегом ** **. Фактично стиль прив'язується до тега **<a>**.

Селектор тег+клас

Селектор: **селектор тег+клас**

Оформлення в CSS: **p.class123 {color: red}**

Елемент в HTML: ** ... **

Стиль буде задіяний до всіх тегів ** **, що містять атрибут **class**, значення якого = **class123**.

Поєднання селекторів

p, .classname, div a, td>#itmRed { color: red }

Допускається довільне поєднання селекторів, перелічених через кому, для певного набору властивостей.

Способи застосування CSS

Має́ться на увазі форма декларування стиля на сторінці і форму зв’язування описання стиля відображення елемента розмітки із самим елементом:

- Зміна стандартного стиля в елементі розмітки
- Розміщення описання стиля в заголовку документа в елементі <STYLE>
- Розміщення посилання на зовнішнє описання через елемент <LINK>
- Імпорт описання стиля в документ

Зміна стандартного стиля

Незважаючи на наявність явної рекомендації розміщувати стилі в окремих файлах, збережено можливість використовувати CSS в рамках самого HTML-документа:

<H1 **STYLE="font-weight:normal; font-style:italic; font-size:10pt;"**>

Заголовок першого рівня

</H1>

Елемент <STYLE>

<HEAD>

<STYLE>

p { color:darkred;text-align:justify; font-size:8pt; }

a { color:blue; font-size:1.2em; }

</STYLE>

</HEAD>

<BODY>

<P> Приклад описання стиля для стандартного елемента HTML-розмітки. </P>

</BODY>

Посилання через <LINK>

Атрибут **rel** вказує на тип підключеного файла, **type** – вказує MIME-тип файла, **href** – вказує шлях до файла стилів.

```
<LINK  TYPE="text/css"  REL="stylesheet"
        HREF="http://domen.com.ua/style.css">
```

Імпорт описання стиля

Неочевидна особливість – імпорт проводимо до появи CSS-інструкцій в даноу файлі.

<STYLE> @import url(http://domen.com.ua/style.css) A { color:cyan; text-decoration:underline; }

</STYLE>

Нюанси варіантів підключення

Внутрішні стилі корисні, оскільки є невід’ємною частиною HTML-документа – веб-сторінка буде виглядати правильно незалежно від доступності зовнішнього файла стилів.

Але є і недоліки: внутрішній стиль працює тільки на одному HTML-документі, і в загальному - застосування внутрішніх стилів протирічить положенню про необхідність відокремити структуру документа від його візуалізації.

Каскадність CSS

Зовнішній список стилів, лінк на який зустрічається в HTML-документі пізніше, має пріоритет відносно іншого списку, лінк на який зустрічається раніше.

Внутрішні списки стилів пріоритетніші за зовнішні, а стилі в елементі (<p style="...">) мають пріоритет відносно інших стилів.

Більш конкретні стилі мають пріоритет перед менш конкретними (p.classname {...} пріоритетніший за p {..}), тобто селектор класу пріоритетніший за перевизначення селектор елемента, і контекстний селектор пріоритетніший за селектор класів.

Якщо для тега вказано кілька селекторів класів, пріоритетнішими будуть ті, що вказані правіше.

Атрибуты стилю, оголошені як !important, мають пріоритет перед всіма іншими значеннями. Так, стиль p {color: red !important} зробить весь текст в тегах <p> червоним незалежно від будь-яких інших визначень стилю для <p>.

CSS-фільтри (хаки)

Відмінності в реалізації CSS різними броузерами заставляють веб-розробників шукати рішення, як заставити всі броузери показувати веб-сторінку однаково. CSS-фільтри (CSS-хаки) дозволяють вибірково застосовувати стилі до різних тегів.

Умовні коментарі виду **<!--[if condition]> HTML <![endif]-->** додають хаки для різних версій IE.

Окремі серектори працюють тільки в конкретних версіях броузерів: *** html {}** або **html:first-child {}**.

Є хаки під конкретні броузерні движки, зокрема для нових експериментальних властивостей, як наприклад анімації/трансформації елементів на веб-сторінці.

CSS-фреймворки

Корисні для спрощення роботи верстальщика, швидкості розробки і виключення максимально можливої кількості помилок верстки (проблеми сумісності броузерів тощо). Як і бібліотеки скриптових МП, CSS-бібліотеки зазвичай мають вигляд зовнішнього css-файла, і підключаються до проєкта через відповідну інструкцію в заголовку веб-сторінки.

Використання CSS-бібліотек викликає суперечки в середовищі професіоналів, і призводить до спеціалізації типів бібліотек. Іноді виходом є створення власної бібліотеки – персональне рішення знижує ризики залежності від сторонніх розробок.

Розширення CSS

В стандартному CSS немає можливостей наслідування стилів, обчислюваних значень та інших залежностей.

Для вирішення цих питань і прискорення розробки існує кілька розширень (препроцесорів) CSS (код CSS є валідним для розширення, але не навпаки). Щоб із «розширеного CSS» отримати звичайний CSS-файл, який розуміє броузер, треба виконати компіляцію:

- під час запуску сторінки на клієнті (Javascript)
- або під час запуску сторінки на сервері
- або під час верстки сторінки засобами спеціального компілятора