ЛАБОРАТОРНА РОБОТА № 1

**Тема:** Засоби побудови веб-сторінок.

**Мета:** Набуття теоретичних та практичних навичок по роботі з основними технологіями побудови веб-сторінок.

Теоретичні відомості

Документи HTML є звичайними текстовими файлами, що містять спеціальні теги (або керуючі елементи ) розмітки. Теги розмітки вказують браузеру Web, як треба вивести сторінку.

Файли HTML зазвичай мають розширення htm або html. Їх можна створювати за допомогою будь-якого текстового редактора.

Мова HTML є підмножиною потужного мови SGML ( Standard Generalized Markup Language ), яка широко використовується у видавничій діяльності. Основний виграш від використання цих мов полягає в однозначній інтерпритації тексту різних видавничих системах. Ця ж особливість зберігається і в HTML. Так, читаючи документ, користувачі можуть встановлювати способи виділення тексту, гарнітуру і розмір шрифтів за своїм смаком ; вони можуть скасувати перегляд малюнків.

У документі HTML можна виділити два основних блоки : головний частина і тіло документа. Вміст головної частини не виводиться на екран користувача, за винятком заголовка, в ній, як правило, вказують ключові слова, авторів та іншу службову інформацію, а також підключають зовнішні таблиці стилів і скрипти. У тілі документа розміщують ту інформацію, яка буде виведена користувачеві.

У різних операційних системах є різні редактори, які можна використовувати для створення документів HTML.

Є також велика кількість спеціалізованих редакторів для створення файлів HTML, таких як FrontPage, Macromedia Dreamweaver або Adobe Web Bundle, які володіють можливістю WYSIWYG ( What You See Is What You Get – що бачиш, те й отримаєш ). З їх допомогою можна легко створювати документи HTML, за допомогою кнопок та елементів меню, а не писати самому теги розмітки. Однак, тим, хто хоче стати технічно грамотним розробником Web, настійно рекомендується використовувати простий текстовий редактор для початкового вивчення HTML (наприклад: Notepad++).

Окрім того корисним буде використання спеціалізованих IDE – IDEA, Atom, Sublime, Visual Code.

Структура сторінки HTML

Надрукуйте наступний текст:

<html>

<head>

<title>Це заголовок сторінки</title>

</head>

<body>

<h1>Привіт!</h1>

<p>Це моя перша сторінка HTML. <b>Цей текст виводиться жирним шрифтом.</b></p>

</body>

</html>

Збережіть файл як page1.html.

При збереженні файлу HTML можна використовувати розширення htm, або html. Розширення htm було прийнято для старих версій операційних систем, які допускали трибуквенне розширення для файлів. В даний час практично всі операційні системи не мають подібного обмеження і можна використовувати розширення html.

Ваш перший HTML-документ починається з тега <html>, який повідомляє браузеру про початок документа HTML і закінчується тегом </html>, який інформує браузер про досягнення кінця документа.

Текст між тегами <head> і </head> є інформацією заголовка документа. Ця інформація не виводиться у вікні браузера.

Текст «Це заголовок сторінки » між тегами <title> і </ title > є заголовком документа. Цей заголовок виводиться в рядку заголовка вікна браузера.

Текст між тегами <body> і </body> є текстом, який буде виведений у вікні браузера. Текст «Привіт!» між тегами <h1> і </h1> буде відображений стилем заголовка, зазвичай жирним шрифтом більшого розміру.

Тег <p> означає, що починається новий параграф, тег </p> означає кінець параграфа.

Текст «Цей текст виводиться жирним шрифтом.» між тегами <b> і </b> буде виведений жирним шрифтом.

Елементи HTML

Загалом елементи (теги) являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст (контент). Існують певні настанови щодо кожного атрибута та контента елемента, які треба виконувати задля того, щоб HTML-документ був визнаний валідним.

У елемента є початковий тег, який має вигляд <element-name>, та кінцевий тег, який має вигляд </element-name>. Атрибути елемента записуються в початковому тегу одразу після назви елемента, контент елемента записується між його двома тегами. Наприклад: <element-name element-attribute="attribute-value">контент елемента</element-name>.

Деякі елементи, наприклад br, не містять контенту, тож і не мають кінцевого тега. Елемент може не мати початкового та кінцевого тега (наприклад, елемент head), проте він завжди буде представлений в документі.

Більшість з атрибутів елемента являє собою пару «назва-значення», розділених між собою знаком рівняння, та записаних у початковому тегу одразу після назви елемента. Значення атрибуту може бути взято в лапки (подвійні або одиничні), також, якщо значення атрибуту складається з певних символів, його можна не виділяти лапками зліва. Проте невзяття значення атрибутів у лапки вважається небезпечним кодом. На відміну від атрибутів виду «назва-значення», є певні атрибути, що впливають на елемент, назва яких лише з'явилась в початковому тегу (наприклад, атрибут ismap елемента img).

Більшість елементів можуть мати будь-який з загальних атрибутів:

1. Атрибут **id** впроваджує унікальний ідентифікатор елемента по всьому документу. Доданий до URL документа, він впроваджує глобальний унікальний ідентифікатор елемента.

Це може використовуватися:

* таблицями стилів для впровадження презентаційних властивостей;
* браузерами для фокусування уваги на певному елементі;
* скриптами для виконання дій над елементом.

1. Атрибут **title** використовується для додавання пояснювального тексту для елемента. В більшості браузерів значення цього атрибуту можна побачити як підказку, що виникає при наведенні курсора на елемент.
2. Атрибут **class** впроваджує засіб об'єднання схожих елементів у класи.

Це може використовуватися для:

* відтворення візуальних ефектів. Для презентаційної розмітки, наприклад, документ може містити class="notation", який визначає всі елементи, у яких клас визначений як "notation", підпорядкованими головному тексту документа. Такі елементи можна зібрати докупи і показати як виноски внизу сторінки, замість того, щоб показувати їх на тому місці, де вони з'являються в самому HTML-коді документа;
* структурного поділу тексту. Для семантичної розмітки, наприклад, класи використовуються у створенні мікроформатів.

Тегів розмітки та атрибутів є велика кількість, тому привести їх опис в межах лабораторної роботи не має можливості. Тому пропонується ознайомитися з ними самостійно. Для цього можна скористатися матеріалами, наведеними в «Корисних матеріалах»

CSS

CSS (Cascading Style Sheets) – це мова стилів, яка визначає відображення HTML-документів. CSS працює з шрифтами, кольором, полями, рядками, висотою, шириною, фоновими зображеннями, позиціонуванням елементів і багатьма іншими речами.

HTML може (неправильно) використовуватися для оформлення web-сайтів, в то час як CSS надає набагато більші можливості для цього. Крім того, CSS підтримується всіма браузерами.

**У чому різниця між CSS і HTML?**

HTML використовується для структурування вмісту сторінки. CSS використовується для форматування цього структурованого вмісту.

Давним-давно мова HTML використовувалася тільки для виведення структурованого тексту. Автор міг тільки розмічати текст: "це - заголовок" або "це - параграф", використовуючи HTML-теги, такі як <h1> і <p>.

У міру розвитку Web дизайнери почали шукати можливості форматування онлайнових документів. Щоб задовольнити вимогам споживачів, виробники браузерів винайшли нові HTML-теги, такі, наприклад, як <font>, які відрізнялися від оригінальних HTML-тегів тим, що вони визначали зовнішній вигляд, а не структуру.

Це також призвело до того, що оригінальні теги структурування, такі як <table>, стали більше застосовуватися для дизайну сторінок замість структурування тексту. Багато нових тегів дизайну, таких як <blink>, підтримувалися тільки одним браузером. «Вам потрібен браузер X для перегляду цього ресурсу» – така відмова стала звичайним явищем на web-сайтах.

CSS був створений для виправлення цієї ситуації шляхом надання web-дизайнерам можливостей точного дизайну, підтримуваних всіма браузерами. Одночасно відбувся поділ представлення і вмісту документа, що значно спростило роботу програмістів та дизайнерів.

**Які переваги CSS?**

Поява CSS стало революцією в світі web-дизайну. Конкретні переваги CSS:

* управління представленням безлічі документів за допомогою однієї таблиці стилів;
* більш точний контроль над зовнішнім виглядом сторінок;
* різноманітні представлення для різних носіїв інформації (екран, друк, і т. д.);
* складна і пророблена техніка дизайну.

**Базовий синтаксис CSS**

Багато властивостей, що використовуються в Cascading Style Sheets (CSS), аналогічні властивостям HTML.

Розглянемо на конкретному прикладі.

Скажімо, нам потрібен червоний колір фону web-сторінки:

В HTML це можна зробити так:

**<body bgcolor="#FF0000">**

За допомогою CSS того ж самого результату можна досягти так:

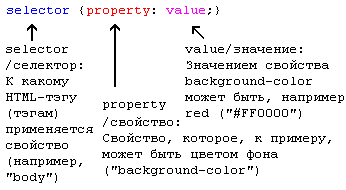
body {background-color: #FF0000;}

Як бачите, ці коди більш-менш ідентичні в HTML і CSS.

CSS має порівняно простий синтаксис і використовує небагато англійських слів для найменування різних складових стилю.

Стилі складаються зі списку правил. Кожне правило має один або більше селектор (англ. *selector*) та блок визначення (англ. *declaration block*). Блок визначення складається із оточеного фігурними дужками списку властивостей.

Властивості в списку оформлюються у вигляді **назва властивості**, двокрапка (:), **значення**, крапка з комою (;).



**Застосування CSS до HTML-документу**

Є три способи застосувати правила CSS до HTML-документу. Рекомендую зосередитися на третьому – тобто зовнішньої / external таблиці стилів, хоча все залежить від ситуації.

**Метод 1: Інлайн / In-line (атрибут style)**

Можна застосовувати CSS до HTML за допомогою HTML-атрибуту style. Червоний колір фону можна встановити так:

<html>

<head>

<title>Example</title>

</head>

<body **style="background-color: #FF0000;"**>

<p>This is a red page</p>

</body>

</html>

**Метод 2: Внутрішній (тег style)**

Другий спосіб вставки CSS-кодів – HTML-тег <style>. наприклад:

<html>

<head>

<title>Example</title>

**<style type="text/css">**

**body {background-color: #FF0000;}**

**</style>**

</head>

<body>

<p>This is a red page</p>

</body>

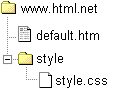
</html>

**Метод 3: Зовнішній (посилання на таблицю стилів)**

Рекомендований метод – створення посилання на так звану зовнішню таблицю стилів.

Зовнішня таблиця стилів це просто текстовий файл з розширенням «**.css»**. Ви можете помістити таблицю стилів на ваш web-сервер або на жорсткий диск, як і інші файли.

Наприклад, ваша таблиця стилів має назву style.css і знаходиться в папці style. Це можна проілюструвати так:



Весь фокус полягає в тому, щоб створити посилання з HTML-документа (default.htm) на таблицю стилів (style.css). Це можна зробити одним рядком HTML-коду:

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

Зверніть увагу, як вказано шлях до вашої таблиці стилів атрибутом href.

Цей рядок коду потрібно вставляти в розділі header HTML, тобто між тегами <head> і </head>. Наприклад, так:

<html>

<head>

<title>My document</title>

****<link rel="stylesheet" type="text/css" href="style/style.css" />****

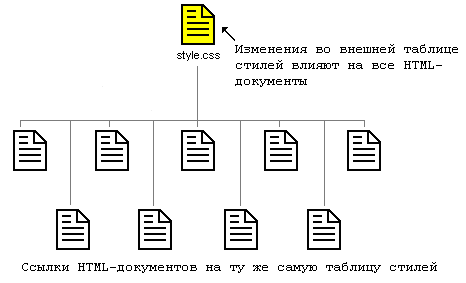
</head>

<body>

...

Це посилання вказує браузеру, що він повинен використовувати правила відображення HTML-файлу з CSS-файлу.

Найважливіше тут те, що кілька HTML-документів можуть посилатися на одну таблицю стилів. Інакше кажучи, один CSS-файл можна використовувати для управління відображенням безлічі HTML-документів.



Це допоможе Вам заощадити багато часу і сил. Якщо Ви, наприклад, хочете змінити колір фону web-сайту з 100 сторінок, таблиця стилів позбавить вас від необхідності вручну змінювати всі сто HTML-документів. Використовуючи CSS, ці зміни можна зробити за кілька секунд, просто змінивши один код в центральній таблиці стилів.

CSS може містити коментарі. Синтаксис коментарів подібний до синтаксису, що використовується в багатьох мовах програмування (наприклад C, PHP).

*/\* comment \*/*

CSS має спеціальні позначення (селектори, від англ. *selector*) для визначення діапазону застосування правила.

CSS селектори – це своєрідна логіка, завдяки якій присвоюють ті, чи інші значення потрібним елементам.

Річ напрочуд корисна, якщо врахувати, що логіка використовується не лише в каскадних (CSS) стилях, але й javascript та його бібліотеках (Jquery, Mootools), а також в багатьох інших фреймворках.

Тобто, зрозумівши сам принцип, Ви зможете легко управляти елементами в інших середовищах.

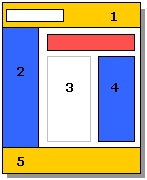
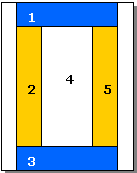
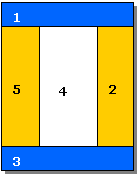
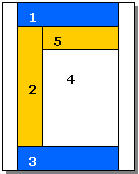
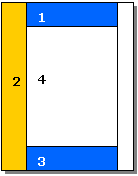
|  |  |  |  |
| --- | --- | --- | --- |
| **Селектор** | **Приклад** | **Опис** | **CSS** |
| .*class* | .intro | Вибирає усі елементи з class=”intro” | 1 |
| #*id* | #firstname | Вибирає усі елементи з id=”firstname” | 1 |
| \* | \* | Просто вибирає усі елементи | 2 |
| *element* | p | Вибирає елементи з тегом <p> | 1 |
| *element,element* | div, p | Вибирає елементи з тегом <div> та усі <p> елементи | 1 |
| *elementelement* | div p | Вибирає усі елементи <p>, **які є всередині** <div> елементів | 1 |
| *element*>*element* | div > p | Вибирає усі елементи <p> безпосереднім материнським елементом яких є <div> елемент | 2 |
| *element*+*element* | div + p | Вибирає першого сусіда <p> який розташовані відразу після <div> на тому ж рівні вкладеності | 2 |
| *element1*~*element2* | p ~ ul | Вибирає усі сусідні списки <ul>, котрі йдуть після <p> елементів на тому ж рівні вкладеності | 3 |
| [*attribute*] | [target] | Вибирає усі елементи, які мають target атрибут | 2 |
| [*attribute*=*value*] | [target=\_blank] | Вибирає усі елементи, в яких атрибут дорівнює конкретному значенню, наприклад: target=”\_blank” | 2 |
| [*attribute*~=*value*] | [title~=порося] | Вибирає усі елементи, в яких атрибут **title** містить слово **порося** | 2 |
| [*attribute*|=*value*] | [lang|=en] | **|=** — означає, що параметр має вже не просто містити, а починатись з **en** | 2 |
| [*attribute*^=*value*] | a[href^=”https”] | Вибирає кожен елемент <a> в якого атрибут **href** починається з “https” | 3 |
| [*attribute*$=*value*] | a[href$=”.pdf”] | Протилежна властивість — тут вибирається кожен елемент, атрибут **href** якого закінчується на “.pdf” | 3 |
| [*attribute*\*=*value*] | a[href\*=”intech”] | Вибирає усі елементи <a>, в яких атрибут містить “intech” | 3 |

Для точнішого визначення стилю можуть використовуватися *псевдо-класи* ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *pseudo-classes*).

|  |  |  |  |
| --- | --- | --- | --- |
| **Селектор** | **Приклад** | **Опис** | **CSS** |
| :active | a:active | Вибирає всі активні лінки | 1 |
| ::after | p::after | Вставляє певний контент після кожного елементу з тегом <p> | 2 |
| ::before | p::before | Вставляє певний контент перед кожним елементом з тегом  <p> | 2 |
| :checked | input:checked | Вибирає усі **checked** <input> (вирані галочками) | 3 |
| :disabled | input:disabled | Вибирає всі “зняті” <input> | 3 |
| :empty | p:empty | Вибирає кожен пустий <p>, тобто без дочірніх блоків та тексту | 3 |
| :enabled | input:enabled | Вибирає кожен “включений”<input> | 3 |
| :first-child | p:first-child | Вибирає перший <p>в заданому рівні вкладення | 2 |
| ::first-letter | p::first-letter | Вибирає першу букву кожного <p> | 1 |
| ::first-line | p::first-line | Кожна перша лінія елементів з тегом <p> | 1 |
| :first-of-type | p:first-of-type | Selects every <p> element that is the first <p> element of its parent | 3 |
| :focus | input:focus | Selects the input element which has focus | 2 |
| :hover | a:hover | Selects links on mouse over | 1 |
| :in-range | input:in-range | Selects input elements with a value within a specified range | 3 |
| :invalid | input:invalid | Selects all input elements with an invalid value | 3 |
| :lang(*language*) | p:lang(it) | Selects every <p> element with a lang attribute equal to “it” (Italian) | 2 |
| :last-child | p:last-child | Selects every <p> element that is the last child of its parent | 3 |
| :last-of-type | p:last-of-type | Selects every <p> element that is the last <p> element of its parent | 3 |
| :link | a:link | Selects all unvisited links | 1 |
| :not(*selector*) | :not(p) | Selects every element that is not a <p> element | 3 |
| :nth-child(*n*) | p:nth-child(2) | Selects every <p> element that is the second child of its parent | 3 |
| :nth-last-child(*n*) | p:nth-last-child(2) | Selects every <p> element that is the second child of its parent, counting from the last child | 3 |
| :nth-last-of-type(*n*) | p:nth-last-of-type(2) | Selects every <p> element that is the second <p> element of its parent, counting from the last child | 3 |
| :nth-of-type(*n*) | p:nth-of-type(2) | Selects every <p> element that is the second <p> element of its parent | 3 |
| :only-of-type | p:only-of-type | Selects every <p> element that is the only <p> element of its parent | 3 |
| :only-child | p:only-child | Selects every <p> element that is the only child of its parent | 3 |
| :optional | input:optional | Selects input elements with no “required” attribute | 3 |
| :out-of-range | input:out-of-range | Selects input elements with a value outside a specified range | 3 |
| :read-only | input:read-only | Selects input elements with the “readonly” attribute specified | 3 |
| :read-write | input:read-write | Selects input elements with the “readonly” attribute NOT specified | 3 |
| :required | input:required | Selects input elements with the “required” attribute specified | 3 |
| :root | :root | Selects the document’s root element | 3 |
| ::selection | ::selection | Selects the portion of an element that is selected by a user |  |
| :target | #news:target | Selects the current active #news element (clicked on a URL containing that anchor name) | 3 |
| :valid | input:valid | Вибирає усі елементи з валідним значенням **value** | 3 |
| :visited | a:visited | Вибирає усі відвідані лінки | 1 |

Порядок виконання роботи

1. Засобами HTML та CSS створіть адаптивну веб-сторінку заданого вигляду, використовуючи:
   * розмітку за допомогою таблиці;
   * розмітку за допомогою блоків.



а) б) в) г) д)

1. В основному блоці (виділений білим) вивести ненумерований список та текст, який складатиметься з заголовка та кількох абзаців. При виводі взяти довільні дані. При форматуванні тексту використати теги h1, span, p, div, b, i.
2. Для кожного з блоків описати власний стиль та зробити так, щоб він реагував на переміщення курсору миші (виділення, підсвічування, зміна фону тощо).
3. Вихідні коди вивантажити в git-репозиторій.
4. Результати роботи оформити звітом.

Зміст звіту

1. Назва й мета лабораторної роботи.
2. Скріншоти результатів роботи.
3. Знімок екрану з файловою структурою проекту.
4. Висновки по лабораторній роботі.

Корисні посилання

1. Навчальні матеріали по HTML:

* <https://www.w3schools.com/html/default.asp>
* <http://htmlbook.ru/samhtml>

1. Навчальні матеріали по CSS:

* <http://htmlbook.ru/samcss>
* <https://www.w3schools.com/css/default.asp>

1. CSS Frameworks:

* <http://getbootstrap.com/>
* <http://materializecss.com/>
* <http://turretcss.com/>
* <http://concisecss.com/>
* <http://www.blueprintcss.org/>
* <https://getuikit.com/>
* <http://modestgrid.com/>
* <http://danmalarkey.github.io/schema/>
* <https://metroui.org.ua/>
* <http://www.responsivegridsystem.com/>
* <http://getleaf.com/>
* <http://semantic-ui.com/>
* <http://foundation.zurb.com/>
* <http://www.cascade-framework.com/>
* <http://basegui.de/>
* <https://siimple.juanes.xyz/>

1. Шрифти – <https://fonts.google.com/>
2. Історія розвитку WEB – <http://www.evolutionoftheweb.com/?hl=en>