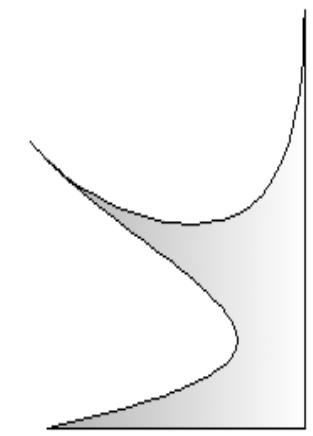
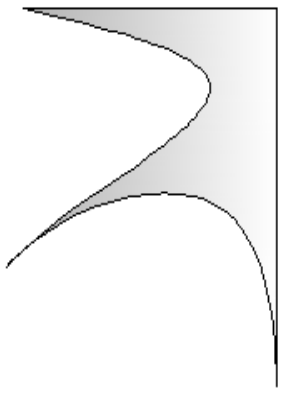


О.М. Шикула

Дизайн Web-сторінок: HTML+CSS



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ

О.М. Шикула

Дизайн Web-сторінок: HTML+CSS

Навчальний посібник

Київ 2018

БКК 32.97
УДК 631.1301

Шикула О.М. Дизайн Web-сторінок:HTML+CSS: Навчальний посібник. / О.М. Шикула – К. ІПДО, 2018. – 60 с.

Анотація

В навчальному посібнику викладено основи мови розмітки гіпертекстових документів (HTML), основні прийоми і методи розроблення та оформлення дизайну Web-сторінок засобами HTML; висвітлено основні способи використання CSS-стилів оформлення зовнішнього вигляду HTML-документів.

Навчальний посібник розроблено на кафедрі інформатики та обчислювальної техніки Інституту післядипломної освіти Національного університету харчових технологій.

Призначено для широкого кола науковців, аспірантів, викладачів, науково-технічних працівників, професійна діяльність яких пов’язана з Web-дизайном.

Автор: О.М. Шикула, доктор фізико-математичних наук, професор

Редактор: Н.Я.КОСТІНА

МОВА РОЗМІТКИ ГІПЕРТЕКСТІВ (HTML)

Лекція 1. Вступ. Структура Web-документа. Робота з текстом в HTML. Типи та описання списків засобами HTML.

Основна мета: ознайомитися з мовою розмітки гіпертекстів HTML та структурою Web-документа; отримати навички зі створення простих Web-сторінок мовою HTML, навчитися працювати з текстовими блоками засобами HTML. Актуалізація знань про призначення, види та структурну організацію списків і таблиць. Формування у фахівця теоретичних знань і практичних навичок щодо створення різних видів списків та таблиць засобами HTML.

МОВА РОМІТКИ ГІПЕРТЕКСТІВ (HTML) 1
Лекція 1. Вступ. Структура Web-документа. Робота з текстом в HTML. Типи та опис списків засобами HTML. 1
Вступ до мови розмітки гіпертекстів (HTML) 1
World Wide Web та її призначення 1
Формат документів HTML. Web-сторінки та Web-сайти 2
Гіпертекстові посилання 2
Адреса Web-документу 2
Браузери та їх можливості 3
Теги та їх атрибути. Синтаксис опису тега 3
Структура Web-сторінки 5
Обов'язкові теги 5
Теги роботи з текстовими блоками. 6
Фізичне форматування тексту 7
Логічне форматування тексту 8
&- послідовності 9
Типи та описання списків засобами HTML 9
Списки та їх типи 9
Організація маркованих списків 10
Організація нумерованих списків 10
Організація вкладених списків 11
Організація списків визначень. 11
Контрольні питання 12

Вступ до мови розмітки гіпертекстів (HTML)

World Wide Web та її призначення

World Wide Web (*WWW, "Всесвітня павутина"*) – найвідоміша і найпопулярніша служба Інтернету. Часто її ототожнюють з усією мережею Internet. Проте, Internet має кілька служб, зокрема, електронну пошту, телеконференції (дискусійні групи), протокол передачі файлів, тощо. Word Wide Web – глобальна розподі-

лена інформаційна гіпертекстова система, яка дозволяє з'єднувати в одну систему розрізнену інформацію, що зберігається на різних комп'ютерах. WWW призначена для інтерактивного пошуку різнотипної інформації за певним способом обміну інформацією, чи протоколом. "Всесвітня павутина" побудована на основі протоколу передавання гіпертексту **http** (hypertext transfer protocol), який дозволяє передавати документи формату HTML із Web-серверів до Web-клієнтів.

В основі World Wide Web лежать два поняття: *формат документів HTML* та *гіпертекстові посилання*.

Формат документів HTML. Web-сторінки та Web-сайти

Формат документів HTML – це певні вказівки форматування тексту документа, задані та описані спеціальною мовою розмітки гіпертексту – HyperText Markup Language (HTML), запропонованої співробітником Європейської лабораторії фізики елементарних часток Тімом Бернерсом-Лі у 1989 році.

Документи формату HTML – Web-документи чи Web-сторінки – мінімальні одиниці подання інформації у просторі WWW. Власне вони і створюють Web-простір. Web-сторінки зберігаються на жорстких дисках Web-серверів. Web-сервери – спеціалізовані комп'ютери з відповідним програмним забезпеченням, яке дає можливість доступу користувачів до Web-сторінок.

Група сторінок, присвячених одній певній темі чи об'єднаних за змістом тем, називається Web-вузлом або Web-сайтом. Один фізичний Web-сервер може містити декілька Web-сайтів. З погляду користувача, наявні межі сайтів мають чисто змістовний характер. Фізично сторінки одного сайту можуть розміщуватися як на одному, так і на різних комп'ютерах мережі Інтернет.

Гіпертекстові посилання

Основним принципом використання Web-сторінок є активізація гіпертекстових посилань. ***Гіперпосилання*** – це або просто текст, або малюнок, який містить посилання на інші місця цієї Web-сторінки або на деяку іншу Web-сторінку. Активізуючи посилання (час від часу, натискаючи кнопкою миші на них), можна переміщатись з одного фрагмента цієї сторінки до іншого, чи до іншої сторінки, тим самим уточнюючи чи роз'яснюючи поняття, які були виділені як гіперпосилання, або виконати певні дії, асоційовані з ними. При цьому фізичне місцезнаходження сторінки не має значення.

Адреса Web-документу

Щоб за допомогою гіпертекстових посилань зв'язуватись з будь-яким Web-документом, останній повинен у Internet мати свою унікальну адресу, яка складається з двох частин: назви протоколу чи виду сервісу (адреси сервера), до якої додається повний шлях пошуку файлу з документом на самому сервері. Наприклад, *http://www.agrari.com/index.htm*

http: –протокол передавання тексту;

// – роздільник, який відокремлює тип протоколу від імені сервера;

www – сервіс прямого доступу, зазначення служби Word Wide Web;

www. agrari.com – доменне ім’я сервера (agrari – власне ім’я сервера);
index.htm – потрібний файл.

Заключною частиною адреси (але не обов’язковою) є повний шлях пошуку потрібного файлу. Такий шлях починається із символу /, потім йде ім’я папки (або декількох вкладених папок) і, нарешті, ім’я файлу. Адресу, записану в такому вигляді називають URL – адресом файлу. URL (uniform resource locator) – універсальний вказівник на ресурс. Стандарт для визначення розташування довільних ресурсів Internet.

Браузери та їх можливості

Програми для перегляду Web-сторінок називають ***браузерами*** (наприклад, Microsoft Internet Exsplorer, Opera, Firefox, Chrome та інші). Принцип роботи браузера полягає в інтерпретаціїрозміток форматування тексту.

Серед **основних можливостей програм-браузерів** є такі:

- завантаження або відображення Web-сторінок за заданою адресом або після активізації гіперпосилань;
- набір засобів навігації (рядок – адреса, кнопки: назад, вперед, папка – журнал, тощо);
- робота з Web-сторінками (перегляд у повно екранному режимі, налаштування кольорів, збереження сторінки чи її компонент, тощо);
- збереження й відкриття файлів різних типів, наприклад: графічних файлів, звукових або відео файлів, тощо.

Базову функцію перегляду Web-сторінок браузера виконує *Браузерний рушій*, часто також *рушій виведення* чи *двигун розміщення*. **Браузерний рушій** — це програмний рушій, який працює з розміченим вмістом (таким як HTML, XML файли, файли зображень тощо), а також форматувальною інформацією (CSS, XSL файли тощо), і виводить вміст на екран згідно з форматуванням у зручному для спостереження вигляді.

Найпоширенішими браузерними рушіями є: *Gecko* — відкритий двигун проекту Mozilla, *Presto* — двигун, розроблений компанією Opera Software, *Trident* — двигун браузера Internet Explorer тощо. При розробленні Web-сторінок слід пам’ятати, що кожен браузерний рушій по різному опрацьовує гіпертекст документу HTML.

Теги та їх атрибути. Синтаксис опису тега

Формат HTML дуже схожий на звичайний текст, тому Web-сторінки можна створювати як у звичайному текстовому редакторі, наприклад **Блокнот**, так і в спеціалізованих редакторах, наприклад **Notepad++.** Текст у форматі HTML розмічається за допомогою вказівок форматування тексту. Такі вказівки називають ***тегами*** (tag). Синтаксис опису тегу визначається в описі спеціальної мови розмітки гіпертексту.

Документи, розмічені за допомогою цієї мови, називають HTML-документами або документами у форматі HTML. Файли з HTML-документами мають розширення .**htm** або .**html.**

Браузер інтерпретує наявні дані у прийнятому документі вказівки HTML і згідно з ними відображає Web-сторінку у своєму вікні. При цьому вказівки форматування тексту не відображаються, проте впливають на спосіб відображення як HTML – документа в цілому, так і його частин. За їх допомогою текст сторінки можна зробити різнобарвним, використовувати різні шрифти, вбудовувати мультиплікацію, відео фрагменти, звук, тощо.

Теги завжди записуються між дужками вигляду <>. У мові HTML використовуються теги двох типів*: парні (контейнери)* та *непарні (одиночні)*. Непарний тег використовується самостійно, а парний може включати в себе як інші теги так і звичайний текст. Непарний тег складається з одного тега і має вигляд <TEG>. Парний тег чи контейнер складається з двох частин – відкриваючого і закриваючого тегів. Відкриваючий тег позначається так як і непарний – <TEG>, а закриваючий додатково використовує слеш </TEG>. Цей тег впливає на виділену частину, що розташована між його початком та кінцем. Непарний тег не містить закриваючої частини. Він може впливати на частину Web-документа, що розташований за ним. Його дія, як правило, закінчується таким подібним тегом, або тегом із більш широкою сферою дії.

Опис тегів складається – завжди з *імені тегу* і додатково з одного або набору *атрибутів*. Імена тегів можна вводити з клавіатури як великими, так і маленькими літерами, які браузером сприймаються однаково.

Багато тегів окрім імені можуть містити ***атрибути*** – елементи, які розширяють можливості тегів і дозволяють гнучко керувати різними настройками відображення елементів веб-сторінки. Атрибути розділяються між собою пробілом.

Загалом **тег має такий синтаксис:**

<Назва_тегу атрибут1=значення .....атрибутN=значення >

Окрім безпосередньо текстової інформації у Web-документі можна задавати кольори елементів гіпертексту. Кольори кодуються за допомогою послідовності з трьох пар символів. Кожна пара кодує шістнадцяткове значення насиченості заданого кольору одним з трьох основних кольорів – червоним (R), зеленим (G) та синім (B) в діапазоні від нуля (00) до 255 (FF). Деякі основні кольори наведено нижче.

Таблиця 1.1 Кольори елементів гіпертексту

Колір			**RRGGBB**
білий	white		FFFFFF
чорний	black		000000
червоний	red		FF0000
темно-бордовий	maroon		800000
оливковий	olive		808000
зелений	green		00FF00
бірюзовий	azure		00FFFF
синьо-зелений	teal		008080
синій	blue		0000FF
темно-синій	navy		000080

голубий	aqva		00FFFF
сірий	gray		A0A0A0 808080
сріблястий	silver		C0C0C0
фуксія	fuchsia		FF00FF
фіолетовий	purple		800080
ліловий	violet		8000FF
жовтий	yellow		FFFF00
коричневий	brown		996633
помаранчевий	orange		FF8000

Крім того, кольори можна задавати назвою кольору англійською мовою.

В гіпертекстових документах можливе використання комбінованих тегів. Але при їх використанні потрібно пам'ятати таке *правило запису комбінованих тег*:

<TEG1><TEG2> … <TEGn></TEGn> … </TEG2></TEG1>

На рисунку 1.1 зображено, як можна та неможна додавати один тег всередину іншого.

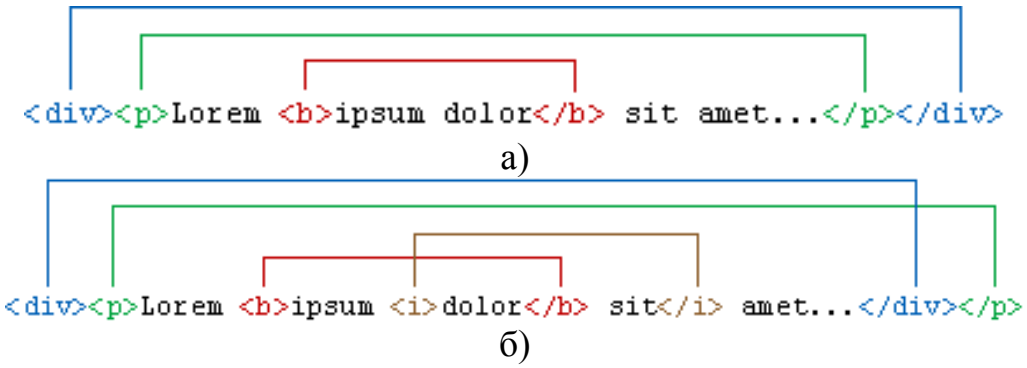


Рисунок 1.1. Правила вкладеннятегів, а) — вірне, б) — невірне

Структура Web-сторінки

Обов'язкові теги

Кожний Web-документ повинен містити такі **обов'язкові теги**:

<HTML> ... </HTML> – парний тег Web-документа, який вказує про початок і закінчення гіпертекстового документа.

<HEAD>...</HEAD> – парний тег, який вказує на початок і кінець заголовка документа.

<TITLE> ... </TITLE> – парний тег назви документа, наприклад показує назву поточного документа в заголовку вікна браузера і друкує його у верхньому лівому куті кожної сторінки при виводі на принтер. Для гіпертекстових документів рекомендується назва не більше 64 символів.

<BODY> ... </BODY> – парний тег, який вказує на тіло гіпертекстового документа, тобто, власне на вміст самого документа.

<!-- текст коментарів -->- тег коментарів. При використанні такого тегу текст не виводиться на екран браузером, а служить для інформації розробнику документа безпосередньо в html -коді.

Структура WEB – документа має такий вигляд:

<HTML>- початок документа
<HEAD>- початок заголовка документа
<TITLE>- заголовок сторінки (відображається у заголовку вікна браузера)
</TITLE>
●●●
Команди (теги) опису зв'язків із ресурсами
●●●
</HEAD>- кінець заголовка документа
<BODY параметри >- початок тіла документа
●●●
Команди (теги) опису WEB документа
●●●
</BODY>- кінець тіла документа
</HTML> − кінець документа

Для тегу <BODY>визначено такі атрибути (параметри):
bgcolor – визначає колір фону документу;
text – визначає колір звичайного тексту в документі;
link – визначає колір елементів гіперпосилань – елементів тексту, які не були переглянуті; при натисканні на них відбувається перехід до іншого елемента документа за гіпертекстовим посиланням;
vlink – колір гіперпосилань, які були переглянуті;
alink – колір гіперпосилань в момент активізації, безпосередньо перед переходом за посиланням (коли на посилання наведений курсор мишки).

Теги роботи з текстовими блоками.

Для визначення заголовків використовуються такі теги:
<H1> заголовок першого рівня </H1>
<H2> заголовок другого рівня </H2>
<H3> заголовок третього рівня </H3>
<H4> заголовок четвертого рівня </H4>
<H5> заголовок п'ятого рівня </H5>
<H6> заголовок шостого рівня </H6>

<P> ... </P>- парний тег абзацу. Все, що знаходиться між відкриваючою і закриваючою частиною, відображається як єдиний абзац.

<DIV> …</DIV>- парний тег блоку. Все, що знаходиться між відкриваючою і закриваючою частиною, сприймається як єдиний блок.

Вказані теги можуть містити атрибут
align – вирівнювання тексту
по лівому краю (***left***)
по центру (***center***),
по правому краю (***right***) або
по ширині (***justify***).

– непарний тег переходу тексту на наступний рядок (розрив рядка) не переривaючи абзацу (використовується, наприклад, при відображенні віршів або пустих рядочків).

<HR>- непарний тег, який вставляє в документ горизонтальну лінію та може мати такі атрибути:
align – вирівнювання лінії по лівому краю (***left***), по центру (***center***) або по правому краю (***right***);
color – колір лінії;
size – визначає товщину лінії в пікселях;
width – ширина лінії, яка встановлюється заданим цілим числом пікселів або у відсотках до ширини сторінки.

<NOBR> … </NOBR>– парний тег, який забороняє використання переносів слів в тексті.
<WBR>– непарний тег, який вставляється в <NOBR> і дозволяє м'які переноси.

Фізичне форматування тексту

<CENTER> … </CENTER>– парний тег вирівнювання тексту по центру

<FONT> ... </FONT>– парний тег форматування шрифту. Може мати такі атрибути:
color – колір шрифту;
dir – напрямок виводу тексту – зліва направо (за замовчуванням або ***ltr***) або справа наліво (***rtl***);
face – гарнітура шрифту (Arial, Courier тощо);
size – визначає розмір шрифту. В HTML розмір шрифту виміряється в умовних одиницях від 1 до 7, середній розмір тексту, використовуваний за замовчуванням, прийнято 3. Шрифти можуть бути задані відносно базового +1, -1, +2, -2, … , +7, -7.

<B> ... </B> – парний тег для відображення тексту **напівжирним**.
<I> ... </I> – парний тег для відображення тексту *курсивом*.
<U> ... </U> – парний тег для відображення тексту підкреслюванням.
<TT> ... </TT>– парний тег для моноширинного (однакова ширина всіх букв) тексту – ефект друкарської машинки.

<BIG> ... </ BIG>– парний тег для збільшення розміру шрифту на одиницю в порівнянні зі звичайним текстом.

<SMALL> ... </SMALL>– парний тег для зменшення розміру шрифту на одиницю в порівнянні зі звичайним текстом.

_{...}– парний тег для нижнього (підрядкового) індексу.

^{...}– парний тег для верхнього (надрядкового) індексу.

<STRIKE> ... </STRIKE>– парний тег для ~~закресленого тексту~~.

<PRE> ... </PRE>– парний тег, який виводить вкладений текст у раніше відформатованому стилі з урахуванням переносів строк та пробілів.

<MARQUEE> ... </ MARQUEE>– парний тег для рядка, що рухається. Може мати такі атрибути:

width – визначає ширину рядка, що рухається, в пікселях або в відсотках до ширини екрану;

height – встановлює висоту рядка, що рухається (в пікселях або в відсотках);

bgcolor –визначає колір фону рядка, що рухається.

behavior – визначає тип руху рядка, має такі значення:

scroll – циклічна прокрутка рядка з одного кінця в іншій;

slide – текст з'являється з одного краю і зупиняється у іншого;

alternate – текст рухається від одного краю до іншого на зразок маятника;

direction – визначає напрямок руху рядка, має такі значення:

left –зображення рухається вліво по рядку;

right – зображення рухається вправо по рядку;

up – весь рядок переміщається знизу нагору;

down – рядок рухається зверху вниз;

scrollamount – визначає кількість пікселів, що відокремлюють один текст від іншого;

scrolldelay – вказує на затримку в мілісекундах перед відображенням наступного текстового блока;

loop – визначає кількість повторів анімації – будь-яке додатне число. За замовчуванням або при заданні значення ***-1*** (***infinite***) браузер буде прокручувати текст нескінченно.

Логічне форматування тексту

<EM> ... </ EM>– парний тег виділеного тексту. Відображається курсивом.

<STRONG> ... </STRONG>– парний тег дуже виділеного тексту. Відображається напівжирним шрифтом.
<CITE> ... </CITE>– парний тег цитати.
<CODE> ... </CODE>– парний тег тексту-фрагменту HTML коду.
<SAMP> ... </SAMP>– парний тег тексту-фрагменту коду програми.
<DFN> ... </DFN>– парний тег тексту-означення.
<KBD> ... </KBD>– парний тег тексту,-що вводиться з клавіатури.
<VAR> ... </VAR>– парний тег тексту-назви змінної або функції. Відображається курсивом.
<BLOCKQUOTE> ... </BLOCKQUOTE> – парний тег, який виводить текст на екран зі збільшеним лівим полем.
<ACRONYM текст – повна назва > текст-скорочення </ACRONYM>– парний тег акроніму.
<!-- -->– тег додає коментарі в код документу.

&- послідовності

Оскільки символи < та > сприймаються браузерами як початок і кінець опису тегу, то в HTML такі символи, в разі потреби, відображаються за допомогою &-послідовностей (символьних об'єктів або ескейп-послідовностей). За допомогою &-послідовностей кодуються букви грецького алфавіту.

Деякі символи кодуються за допомогою таких послідовностей:
& lt; – символ <;
> – символ >;
  – нерозривний пробіл ;
§ – параграф §;
° – градус °;
± – плюс-мінус ±;
& – символ & (амперсанд);
" – лапки – символ ";
&#8212– тіре –.

Примітка: крапка з комою є **обов'язковим** елементом &-послідовностей. Окрім того, всі літери послідовності повинні бути **малими.**

ТИПИ ТА ОПИСАННЯ СПИСКІВ ЗАСОБАМИ HTML

Списки та їх типи

Важливим елементом тексту, який допомагає організувати текст і привернути увагу до найбільш важливих моментів є список.

Список – набір абзаців, які є переліком дій, об'єктів, предметів, тощо, відмічений або номером, або спеціальною позначкою – маркером.

Списки поділяються на:

- ***марковані*** – дозволяють виділяти абзаци за допомогою спеціальних графічних знаків (маркерів), які ставляться на початку абзацу;
- ***нумеровані*** – як маркери використовується нумерація (арабськими або латинськими цифрами), найчастіше застосовуються для визначення послідовності дій або процесів;
- ***вкладені (багаторівневі)*** – складаються з кількох маркованих (нумерованих списків), вкладених один в один (різного рівня).

HTML дозволяє організовувати текст в списки і виводити їх на екран у відформатованому вигляді. Також в HTML існує можливість форматування типів маркерів, які використовуються в маркованих та багаторівневих списках.

Окрім того, в HTML існує можливість створювати спеціальні списки визначень.

Організація маркованих списків

Маркірований список в HTML визначається за допомогою парного тегу <UL>. Вказаний тег має такі атрибути:

compact – відформатовати список більш компактно;

type – призначає тип маркера абзацу як

disc (круг, заповнене коло, встановлюється за замовчуванням),
circle (кільце, незаповнене коло) або
square (заповнений квадрат).

Кожний новий елемент списку необхідно починати з непарного тегу <LI>, який вказує на початок нового абзацу в списку. Для такого тегу також можливе використання атрибуту *type* з типами маркерів. Тоді маємо список, в якому кожен елемент позначений власним маркером.

Формат UL
<UL TYPE=тип маркера**>***текст заголовка списку* **(**виводиться без маркера)
<LI> *елемент списку*
<LI>*елемент списку*
...
<LI>*елемент списку*
</UL>

Організація нумерованих списків

Нумерований список в HTML визначається за допомогою парного тегу <OL>. Вказаний тег має такі атрибути:

compact – відформатовати список більш компактно;

start – почати нумерацію списку із заданого номера (а не з 1);

type – призначає формат нумерації списку як

A – великі літери,
a – маленькі літери,
I – великі римські цифри,
i – маленькі римські цифри

1 – арабські цифри задаються за замовчуванням, або ***1***

Кожний новий елемент списку також необхідно починати з непарного тегу <LI>.

Формат нумерованого списку аналогічний формату списку маркірованого.

***Формат* OL**

<OL TYPE= формат нумерації списку*>текст заголовка списку* **(**виводиться без номера)

<LI> *елемент списку*

<LI>*елемент списку*

...

<LI>*елемент списку*

</OL>

Організація вкладених списків

За допомогою тег <OL> і <UL> можливо створювати багаторівневі (вкладені списки). Але у випадку створення таких списків необхідно уважно слідкувати за правилом запису комбінованих тег, наприклад:

Багаторівневий список (нумерований в маркованому)

<UL TYPE=тип маркера*>текст заголовка списку*

<LI> *елемент списку*

<OL TYPE=тип маркеру*>текст заголовка списку*

<LI> *елемент списку*

<LI>*елемент списку ...*

<LI>*елемент списку*

</OL>

<LI>*елемент списку*

...

<LI>*елемент списку*

</UL>

Організація списків визначень.

Для організації списку визначень використовують парний тег <DL>. При цьому як мітки списку використовуються теги

<DT> (definition term) – тег, який вказує на термін, що визначається, і

<DD> (definition definition) – тег, який вказує на визначення терміну, який входить до списку визначень.

Формат DL

<DL>

<DT>HTML

<DD> Термін HTML (HyperText Markup Language) означає мова опису гіпертекстів.
<DT>HTML-документ
<DD> Текстовий файл з розширенням *.htm або *.html.
</DL>

На екран цей фрагмент буде виведено так:

HTML
Термін HTML (HyperText Markup Language) означає мова опису гіпертекстів.
HTML-документ
Текстовий файл з розширенням *.htm або *.html.

Якщо терміни достатньо короткі, то для <DL> можна використовувати атрибут *compact*. Наприклад:

<DL COMPACT>
<DT>А
<DD> Перша буква абетки
<DT>Б
<DD> Друга буква абетки
<DT>В
<DD>Третя буква абетки
</DL>

На екран цей фрагмент буде виведено так:

А Перша буква абетки
Б Друга буква абетки
В Третя буква абетки

КОНТРОЛЬНІ ПИТАННЯ

1. Призначення, структура та основні можливості Word Wide Web.
2. Принципи розміщення інформаційних ресурсів у WWW (Web-сторінки, Web-сайти, (Web-вузли), Web-сервери).
3. Адреса WEB-документа.
4. Поняття браузера та основні його можливості, принцип роботи.
5. Загальна характеристика мови HTML, поняття гіпертексту.
6. Основні принципи WWW (формат HTML, гіпертекстові посилання).
7. Поняття тегу; парний тег (контейнер), не парний (одиночний) тег, параметри тегу.
8. Загальна структура WEB-документа та призначення кожного розділу.
9. Обов’язкові теги WEB-документа.
10. Теги керування абзацами та блоками; вирівнювання абзаців та блоків.
11. Теги керування розривом (переносом) рядка.
12. Теги фізичного форматування тексту.

13. Теги логічного форматування тексту.
14. Тег встановлення шрифту документа (Font) та його параметри.
15. Списки, їх типи. Теги, які задають типи списків.
16. Нумерований список, його параметри та опис засобами HTML.
17. Маркірований список, його параметри та опис засобами HTML.
18. Опис елемента списку, його параметри.
19. Списки визначень, їх параметри та опис засобами HTML.
20. Багаторівневі списки (списки в списках).

МОВА ОПИСУ ГІПЕРТЕКСТІВ (HTML)

Лекція 2. Таблиці, їх організація, опис та форматування засобами HTML. Вставка та форматування малюнків на Web-сторінці. Поняття гіперпосилань (адреси), їх опис та створення

Основна мета: Набуття знань про призначення, види та структурну організацію таблиць. Формування у фахівця теоретичних знань і практичних навичок щодо створення таблиць засобами HTML. Актуалізація знань про типи форматів зберігання графічної інформації та способи розміщення графічних зображень на Web-сторінці. Формування у фахівця теоретичних знань і практичних навичок щодо підключення графічних файлів засобами мови HTML. Актуалізація знань про призначення та основні можливості гіпертекстових посилань. Формування у фахівця теоретичних знань і практичних навичок використання методів опису елементів, способів прив'язки та застосування гіпертекстових посилань для створення меню Web-документа

МОВА ОПИСУ ГІПЕРТЕКСТІВ (HTML)..... 14
Лекція 2. Таблиці, їх організація, опис та форматування засобами HTML. Вставка та форматування малюнків наWeb-сторінці. Поняття гіперпосилань (адреси), їх опис та створення..... 14
ТАБЛИЦІ, ЇХ ОРГАНІЗАЦІЯ, ОПИС ТА ФОРМАТУВАННЯ ЗАСОБАМИ HTML..... 15
Таблиці у Web-документах 15
Створення структури таблиці 15
Створення рядків та колонок таблиці 17
ВСТАВКА ТА ФОРМАТУВАННЯ МАЛЮНКІВ НА WEB-СТОРІНЦІ.... 18
Графічні файли у Web-документах 18
Вставка фонового зображення на Web-сторінці..... 20
Включення графічних зображень в HTML – документ. 20
Використання графіки для створення маркерних списків..... 21
ПОНЯТТЯ ГІПЕРПОСИЛАНЬ (АДРЕСИ), ЇХ ОПИС ТА СТВОРЕННЯ.. 22
Гіпертекстові посилання, їх призначення та структура. 22
Створення гіперпосилань на Web-сторінці 23
Створення внутрішніх посилань на Web-сторінці 24
КОНТРОЛЬНІ ПИТАННЯ 25

ОРГАНІЗАЦІЯ, ОПИС ТА ФОРМАТУВАННЯ ТАБЛИЦЬ ЗАСОБАМИ HTML

Таблиці у Web-документах

Таблиця – цифрово-текстова інформація, яка організована у вигляді рядків та стовпчиків. Таблиця складається з комірок – прямокутних елементів, утворених перетином одного рядка та одного стовпчика. Кожна комірка може містити текст, графічну інформацію, іншу вкладену таблицю, тощо.

Традиційно таблиці використовуються як метод подання різного типу числових даних. Наприклад, подання статистичних даних у формі таблиць полегшує порівняльний їх аналіз. Крім того, таблиці часто використовують для розбиття друкованої сторінки на розділи з метою її компоновки (в цьому випадку границі таблиці не видно, оскільки рамки комірок таблиці не промальовуються).

Для Web-сторінок таблиці також служать одним із основних елементів дизайну. Використання таблиць у Web-документах не обмежується тільки організацією даних, які містяться в рядках та стовпчиках. HTML-таблиці застосовуються для керування розташуванням елементів та їх групуванням. Одним із методів застосування таблиць є організація розташування різного типу даних на сторінці, які можуть складатись із простого тексту, зображень, списків, інших таблиць, тощо.

Опис таблиці повинен міститись всередині тіла, тобто, всередині розділу <BODY>. Документ може містити довільну кількість таблиць, крім того допускаються вкладені одна в одну таблиці.

Таблиця Web-документа, як і таблиця звичайного друкованого документа, має свою певну структурну організацію – логічно організовані розділи. Вона має заголовкову частину, основну частину (тіло), тобто, подання деяких даних, та заключну, підсумкову частину, в якій містяться висновки, узагальнення, зноски, тощо. У складі таблиці допустимий один розділ заголовків, один розділ підсумків і будь-яка кількість розділів тіла таблиці. У складі кожного розділу може бути будь-яка кількість рядків таблиці. Усі рядки, що входять у групу, розглядаються одночасно, це набагато зручніше, ніж задавати формат для кожного рядка.

Створення структури таблиці

Визначення таблиці у Web-документі визначає парний тег <TABLE атрибут> … </TABLE>. Тег таблиці має такі атрибути:

- *align* – вирівнювання таблиці по лівому краю (***left***), по центру (***center***) або по правому краю (***right***);
- *bgcolor* – визначає колір фону для всієї таблиці;
- *background*– визначає адресу фонового малюнка;
- *border* – встановлює ширину рамки таблиці і комірок в пікселях. Якщо встановити 0, рамку не буде видно;
- *bordercolor* – визначає колір рамки для всієї таблиці;
- *bordercolordark* / *bordercolorlight* – встановлює темний / світлий відтінок відображення рамки для всієї таблиці;

cellspacing – відстань між комірками (в пікселях). За замовчуванням воно дорівнює 2;

cellpadding – відстань між рамкою та даними (в пікселях). За замовчуванням воно дорівнює 1;

width – ширина таблиці, яка встановлюється заданим цілим числом пікселів або у відсотках до ширини сторінки (в останньому випадку потрібно додати до числа знак %).

height – висота таблиці, яка встановлюється заданим цілим числом пікселів або у відсотках до висоти сторінки (в останньому випадку потрібно додати до числа знак %).

<CAPTION атрибут>*текст*</CAPTION> – визначає заголовок (підпис) таблиці. За замовчанням заголовок розташовується над таблицею по центру. Але можливо використання двох атрибутів, які визначають місцезнаходження підпису таблиці:

align – вирівнювання заголовка по лівому краю (***left***), по центру (***center***) або по правому краю (***right***);

valign – розташування підпису над таблицею (***top***) або під таблицею (***bottom***).

Примітка: Для деяких браузерів використовується тільки атрибут *align* із значеннями ***top*** і ***bottom***, вирівнювання заголовка по горизонталі відбувається автоматично по центру.

Тег CAPTION не є обов'язковим, але якщо необхідно підписати таблицю, то він вказується всередині тегу TABLE, який визначає таблицю в цілому.

<THEAD атрибут>*текст рядків заголовку*</THEAD> – визначає заголовкову частину таблиці.

<TBODY атрибут>*текст рядків основної частини*</TBODY> – визначає основну частину таблиці (тіло таблиці, групу рядків).

<TFOOT параметри >*текст рядків підсумкової частини*</TFOOT> – визначає нижню частину таблиці (містить підсумки).

Для трьох вищенаведених тегів можливо використання атрибутів *align* та *valign*:

align – горизонтальне вирівнювання тексту рядків відповідної частини таблиці: по лівому краю (***left***), по центру (***center***) або по правому краю (***right***) (за замовчуванням – по центру);

valign – вертикальне вирівнювання тексту рядків відповідної частини таблиці: по верхньому краю(***top***),по середині (***middle***) або по нижньому краю (***bottom***) (за замовчуванням – по центру).

Формат таблиці (приклад)

<TABLE BORDER=10 BORDERCOLOR = red><!--початок таблиці-->
<CAPTION=TOP>назва таблиці</CAPTION>

<THEAD ALIGN = CENTER BGCOLOR=0000FF><!--заголовок таблиці--></THEAD>
<TBODY ALIGN = RIGHT BGCOLOR=00FF00>
Опис рядків та колонок таблиці
</TBODY><!--кінець основної частини таблиці (тіла)-->
<TFOOT BGCOLOR = FF0000>
<!--підсумкова частина таблиці-->
</TFOOT><!--кінець підсумкової частини таблиці-->
</TABLE><!--кінець таблиці-->

Створення рядків та колонок таблиці

В HTML-документах для визначення рядків застосовується парний тег <TR>. Тегу, який визначає стовпчики в HTML не передбачено. Рядок складається з комірок, які і задають стовпчики Web-таблиці. Для опису комірок у HTML застосовують парні теги (контейнери) <TH> (за звичаєм для комірок рядків заголовку) та <TD> (для рядків основної частини).

***Складові таблиці*:**

<TR атрибут> ... </TR> – визначає рядок таблиці.

Для тегу <TR> можливо використання атрибутів:

align – вирівнювання тексту в комірках рядка по лівому краю (***left***), по центру (***center***) або по правому краю (***right***);

valign – вирівнювання тексту в комірках рядка по верхньому краю (***top***) по середині (***middle***), по нижньому краю (***bottom***) або за базовою лінією (***baseline***) – перші рядки кожної комірки вирівнюються за однією (базовою) лінією;

height – висота рядка в пікселях або відсотках до висоти таблиці;

bgcolor – визначає колір фону для комірок рядка;

border – встановлює ширину рамки комірок рядка в пікселях;

bordercolor – визначає колір рамки для комірок рядка;

bordercolordark / *bordercolorlight* – встановлює темний / світлий відтінок відображення рамки для комірок рядка.

Для визначення комірок рядка таблиці використовуються теги <TD> … </TD> та <TH> … </TH>:

<TD атрибут> (*текст комірки)*</TD> – за замовчуванням вирівнювання по лівій межі комірки.

<TH атрибут> (*текст комірки)*</TH> – за замовчуванням вирівнювання по центру комірки, напівжирний шрифт.

Для тегів <TD> і <TH> використовуються такі атрибути:

align – вирівнювання тексту в комірці по лівому краю (***left***), по центру (***center***) або по правому краю (***right***);

valign – вирівнювання тексту в комірці по верхньому краю (***top***) по середині (***middle***), по нижньому краю (***bottom***) або за базовою лінією (***baseline***).

nowrap – заборона переносу тексту по словах в комірці (текст розташовується в один рядок);

width – ширина комірки в пікселях або відсотках до ширини таблиці;

bgcolor – колір фону для комірки;

bordercolor –колір рамки для комірки;

bordercolordark / *bordercolorlight* – встановлює темний / світлий відтінок відображення рамки для комірки;

cellspacing – відстань між комірками (в пікселях);

cellpadding – відстань між рамкою та даними (в пікселях);

colspan – об'єднання коміркою стовпчиків (розтягування на кілька стовпчиків вправо: colspan=3 – комірка розтягується на три колонки);

rowspan– об'єднання коміркою рядків (розтягування на кілька рядків вниз: rowspan=3 – комірка розтягується на три рядки)

Примітка. Деякі браузери, якщо комірка таблиці порожня, рамку навколо неї не малюють. У випадку, коли потрібно створити порожню комірку з рамкою, в цю комірку можна ввести символьний об’єкт   (non-breaking space – пробіл, що не розриває).

ВСТАВКА ТА ФОРМАТУВАННЯ МАЛЮНКІВ НА WEB-СТОРІНЦІ

Графічні файли у Web-документах

Зображення на екрані монітора подається певною кількістю різнокольорових точок – пікселів, які утворюють картинку. Графічний файл містить інформацію про те, як подати цей набір точок на екрані. Існує багато способів описання графічної інформації, відповідно є значна кількість форматів зберігання графічних файлів.

Усі **формати зберігання графічної інформації** можна звести до двох типів: *векторного та растрового*.

Файли векторної графіки містять математичні дані про те, як перемалювати зображення за допомогою відрізків прямих (векторів) при виведенні його на екран. Процес виведення вимагає додаткового оброблення. Але таке подання графічної інформації має важливі переваги: масштаб зображення може бути змінений без втрати якості, оскільки не існує фіксованого зв’язку між тим, як він визначений у файлі, та виведенням точок на екран.

Векторна графіка, як правило, застосовується для зображень із чіткими геометричними формами. Прикладом її застосування є системи автоматизованого проектування (САПР). У векторному вигляді зберігається інформація для деяких типів шрифтів.

Ра́строва гра́фіка (англ. Raster graphics) є частиною комп'ютерної графіки, яка має справу зі створенням, обробкою та зберіганням растрових зображень. Растро-

ве зображення є масивом кольорових точок (пікселів). Для відображення растрової графіки не потрібно складних математичних розрахунків, досить лише одержати дані про кожну точку та відобразити її на екрані. Але при масштабуванні растрової графіки відбувається втрата роздільної здатності, що погіршує якість зображення.

Більшість форматів зберігання растрових зображень припускає стиснення даних. Тому перед виводом зображень браузерами допустимо змінювати їх масштаб. Залежно від алгоритму, який застосовується для масштабування зображень, можна одержати різний результат. Усі ці питання успішно вирішуються сучасними браузерами. Причому це виконується миттєво, при завантаженні зображення, однак, конкретні алгоритми можуть відрізнятись, що призводить до різного подання одного й того ж документа в різних браузерах.

На Web-сторінках, в основному використовується растрова графіка у форматах **GIF** (Graphic Interchange Format), **JPG** та формату **JPEG** (Joint Photographic Expert Group) і формат **PNG** (Portable Network Graphic).Крім того, деякі браузери безпосередньо або за допомогою додаткових модулів підтримують графічні файли у форматах **xbm, xmp, bmp**.

Формат **BMP** є стандартом MS Windows й підтримується браузером Internet Explorer. Однак його застосування не рекомендується, тому що цей формат не підтримує стиснення даних. Це означає, що Web-сторінки, які містять зображення в такому форматі, завантажуються набагато довше.

Використання форматів GIF та JPG зумовлене тим, що тільки для них здійснюється вбудована підтримка в більшості браузерів, а також структура файлів GIF та JPG найкраще застосовувати для передачі даних по мережі незалежно від платформи. Для мережевих додатків визначальним фактором є розмір файлу, від якого безпосередньо залежить час передавання даних

Для таких зображень, як креслення, побудовані за допомогою графічних пакетів, подання у форматі GIF компактніше в порівнянні з поданням у форматі JPEG. Формат GIF підтримує 256 кольорів, тому графічні зображення, які містять невелику кількість кольорів, найкраще зберігати у цьому форматі. Стиснення зображень при цьому буде максимально допустимим.

Формат JPEG підтримує 16,7 мільйона кольорів, тому його застосовують для збереження фотознімків у цифровому форматі та інших зображень, для яких характерними є часті зміни кольорів. Розмір файлів формату JPEG, як правило, буде меншим у порівнянні з GIF-файлами.

Нижче наведено схему стиснення інформації різних форматів графічних файлів.

Формат	Схема	Опис
GIF	Без втрат	Інформація стискається без втрати деталей. Вибирається, у випадку, якщо більше цікавить збереження деталей, ніж швидкість завантаження.
PNG	Без втрат	Інформація стискається без втрати деталей. Альтернативний GIF, але не підтримується всіма браузерами WWW.

JPEG	З втратами	Призводить до втрати деталей при збереженні зображення. Застосовуємо у випадку, якщо розмір файлу нам важливіший, ніж деталі.

Оскільки час завантаження зображення визначає час завантаження всієї сторінки, бажано подавати зображення в такому форматі, який забезпечує мінімальний розмір файлу, не ігноруючи наших вимог до якості зображення.

Графічні елементи в Web-сторінках можуть використовуватися як фонові зображення, так і як графічні зображення в тексті Web-документа. Як фонові зображення використовуються графічні файли форматів GIF та JPG. Фонове зображення HTML-документів завжди заповнює весь простір вікна перегляду. Якщо розмір зображення менше вікна перегляду, воно буде скопійоване і за принципом мозаїки заповнить весь простір вікна. Це накладає певні вимоги при підборі фонових зображень. Зазвичай, як фонове береться невелике зображення, для завантаження якого по мережі не потрібно багато часу.

Вставка фонового зображення у Web-сторінку

У випадку встановлення **фонового зображення** Web-сторінки, застосовують параметр *background***,** в якому як значення вказується назва файлу, що містить малюнок, який буде утворювати фон нашої сторінки. Цей параметр застосовують у відкриваючому тегу **<BODY>.**

Формат тегу: <BODY BACKGROUND=“fon.gif”>.

Фонові зображення можна застосовувати не тільки до всього документа. Багато браузерів підтримують задавання фонового зображення окремій комірці таблиці.

Включення графічних зображень в HTML – документ.

Для включення у текст Web-сторінки графічного зображення використовують непарний тег <**IMG**>. Він є елементом текстового рівня, тобто, його застосування не створює новий абзац. Це слід враховувати, і, при необхідності, стежити за створенням абзаців за допомогою тегу
 або інших тегів блокового рівня.

Непарний тег <**IMG**> має обов’язковий атрибут *src*. Він визначає розташування файлу, який містить те зображення, яке ми будемо включати в документ. Значенням *src* може бути як абсолютна, так і відносна URL-адреса файлу.

Формат тегу:

```
< IMG   SRC = “ http://www.sam-izd.com / ~ tar / portrait.jpg ”
                ALT=”my foto”>
< IMG   SRC = “images / Vasay. gif” ALT=”my cat”>
```

Атрибут *alt* визначає рядок, який виводиться замість зображення в браузерах, що не підтримують графічні файли (так званий альтернативний текст). У специфікації HTML 4.0 цей параметр є обов’язковим.

Для тегу <IMG> можливо використання таких **атрибутів**:
height – встановлює висоту зображення в пікселях;
width – встановлює ширину зображення в пікселях;

Примітка. Якщо у складі тегу <IMG>атрибути *height* та *width* вказані, браузер може спочатку відобразити текст, а потім уже завантажити та вивести зображення, не змінюючи при цьому структуру сторінки. Якщо значення атрибутів *height* та *width* не співпадають із реальними розмірами зображення, останнє або розтягнеться, або стиснеться до розмірів, заданих такими параметрами у складі тегу.

align – вирівнює зображення відносно тексту або інших елементів:
left – зображення міститься біля лівого поля вікна, текст обтікає зображення з правого боку;
right – зображення міститься біля правого поля вікна, текст обтікає зображення з лівого боку;

Примітка. Ці значення атрибута *align* застосовуються тоді, коли зображення використовується як ілюстрація до змісту фрагмента документа – „плаваючі" зображення.

top – верхня межа зображення вирівнюється за найвищим елементом поточного рядка;
textop – верхня межа зображення вирівнюється за найвищим текстовим елементом поточного рядка;
middle – вирівнювання середини зображення за базовою лінією поточного рядка;
absmiddle – вирівнювання середини зображення посередині поточного рядка;
baseline, bottom – вирівнювання нижньої межі зображення за базовою лінією поточного рядка;
absbottom – вирівнювання нижньої межі зображення за нижньою межею поточного рядка.

Примітка. При використанні параметрів цієї групи зображення „вбудовується" в рядок тексту, а параметри вирівнювання задають розташування зображення відносно рядка тексту. Отже, вбудовані зображення є звичайним елементом рядка (нібито одна велика літера тексту).

hspace – визначає величину відступів від зображення по горизонталі (вільний простір зліва та справа) у пікселях;
vspace – визначає величину відступів від зображення по вертикалі (вільний простір зверху та знизу) у пікселях;
border – встановлює ширину рамки зображення в пікселях.

Використання графіки для створення маркерних списків

Графічні зображення можна використовувати як маркери маркерних списків. Такі зображення надають останнім привабливості та зручності при перегляді їх

користувачем. Для організації такого списку перед кожним елементом списку треба вставити потрібне графічне зображення, використовуючи тег <IMG>. При цьому необхідно пам’ятати, що це елемент текстового рівня, тобто, його застосування не буде відокремлювати елементи списку один від одного. Для відокремлення можна використати теги абзацу <P> або примусового переведення рядка
.

Формат списку:

```
<UL ><B>Заголовок</B><BR>
        <IMG SRC="image.gif"> елемент списку 1 <BR>
        <IMG SRC="image.gif"> елемент списку 2 <BR>
        …
        <IMG SRC="image.gif"> елемент списку n <BR>
</UL>
```

ПОНЯТТЯ ГІПЕРПОСИЛАНЬ (АДРЕСИ), ЇХ ОПИС ТА СТВОРЕННЯ

Гіпертекстові посилання, їх призначення та структура.

Важливим поняттям в HTML є поняття ***гіпертекстових посилань***. Сама назва – HTML, тобто, мова розмітки (чи опису) гіпертексту свідчить про принцип організації таких документів. Зазвичай Web-сторінка не містить послідовного викладу якогось матеріалу. Гіпертекстовий документ містить посилання на інші документи або розділи того ж документа. Активізація посилань здійснює швидкий перехід до іншої сторінки, яка, в свою чергу, може мати зв’язок з іншими, зв’язаними між собою кількома сторінками. Така організація Web-документа робить його інтерактивним та динамічним, тобто, дає можливість користувачам самим переходити за посиланнями до теми, яка найбільш цікавить їх, а браузеру – відображати необхідну інформацію.

Гіперпосилання **складається із двох частин**. ***Перша частина*** відображається на Web-сторінках. Таке місце в документі, помічене як посилання (покажчик посилання), називається *елементом прив’язки* або *якорем* (anchor). Елемент прив’язки в посиланні може бути текстовим чи графічним – словом, групою слів, фразою або зображенням. Зовнішній вигляд посилання залежить від його типу, способів створення та конфігурації браузера.

Текстові покажчики зазвичай виділяються підкресленням. Колір текстового елемента прив’язки регулюється розробником сторінки та установками програми перегляду. Зазвичай браузер виділяє його іншим кольором, ніж звичайний текст. Текстові елементи прив’язки можуть міститись всередині тексту і бути його частиною, а можуть розташовуватись у вигляді списку посилань, із яких користувач вибирає потрібне. Посилання на документи, які уже були переглянуті користувачем в поточному сеансі, зазвичай виводяться з підкресленням та відображаються кольором, який відрізняється від початкового.

За принципом дії графічні посилання нічим не відрізняються від текстових. Але вони не підкреслюються і не виділяються кольором. Для виділення графічних посилань, браузери зазвичай обводять їх рамкою. Графічні елементи прив'язки не схожі між собою, а їх зовнішній вигляд залежить від вибраного автором зображення. Перевагами графічних посилань є багатоваріантність та багатофункціональність – це можуть бути маркери списку, піктограми, малюнки, реклама, тощо.

При створенні гіпертекстових посилань варто враховувати те, що працювати з документом набагато зручніше, якщо текст або графічний елемент, який оформлений як посилання, відображає її призначення.

Друга частина, яка дає інструкцію браузеру, називається *адресною частиною* посилання (URL-адреса). При активізації покажчика посилання браузер завантажує документ, адреса якого дається **URL-адресом**. Указана адреса може бути як ***відносною,*** так і ***абсолютною.***

Якщо в URL-адресі не вказується повний шлях до файлу, то таке посилання називають ***відносним*** – посилання на сторінки, розміщені на одному сервері. В цьому випадку визначення місцезнаходження файлів виконується з врахуванням місцезнаходження документа, в якому є таке посилання, тобто підкаталог на тій же машині. (Адреси таких посилань задаються відносно адреси Web-сторінки, на якій вони розміщені.) Відносні посилання зручні в користуванні – набагато простіше вставити тільки ім'я файлу, а не весь довгий шлях URL-адреси. Вони також дозволяють переміщати файли у межах Вашого сервера без великих змін у міжсторінковій адресації.

URL-адреси, які повністю визначають комп'ютер, каталог та файл, називають ***абсолютними.*** На відміну від відносних, абсолютні покажчики можуть посилатися на файли, які містяться на інших серверах.

Крім посилань на інші документи, часто буває корисно включати посилання на різні частини поточного документа. Для організації внутрішніх посилань спочатку обов'язково треба створити в тексті якір, який визначає на яку частину документа треба переходити при виконанні посилання. За допомогою внутрішніх посилань можна організовувати переходи між розділами, текстовими блоками, поясненнями, а також створювати змісти і меню.

Створення гіперпосилань у Web-сторінці

Для створення гіперпосилань в HTML використовують парний тег <**A**> … </**A**>. Текст, який міститься всередині контейнера буде текстовим елементом прив'язки.

Обов'язковим атрибутом тегу <A>, який визначає гіперпосилання є параметр *href,* який вказує URL-адресу ресурсу, що призначається гіперпосиланням. Тобто, він визначає адресу сторінки (ресурсу), який повинен завантажити браузер в тому випадку, якщо користувач активізує це гіперпосилання. Покажчик може бути як відносним, так і абсолютним.

Формат тегу:

<A HREF=”ім’я_файла” *інший атрибут*>текст чи малюнок (прив’язка) **</A>**

Для тегу <A> можливим є використання таких атрибутів:
color– колір;
font-size– абсолютний розмір шрифту в пунктах;

Зауваження. Щоб заголовок, який створюється за допомогою тегу <H1> перетворити в посилання, потрібно такий текст вставити в тег <A>. В HTML5 послідовність тегів не має значення, тому елемент <H1> можливо вставити в середину тегу <A>. А в HTML4 и XHTML тег <A>повинен знаходитися всередині <H1>.

Приклади використання тегу:
<A href=’’File.html”> “*Натисніть тут”*</A> – посилання-текст на сторінку, яка міститься на цьому ж сервері, в тій же папці, що і гіперпосилання; якщо файл знаходиться в іншій папці, необхідно вказати повний шлях до нього.
<A href=’’File_2.htm”><IMG SRC=”mal_1.GIF”></A> – посилання-малюнок.
<A href=’’http://Agrari.Kiev.Ua/~(повний шлях)~/File.html”> “*Натисніть тут*” </A> – посилання на HTTP-протокол.
<A href=’’FTP://ftp:Lic145.Kiev.Ua// ~(повний шлях)~ /File.html”> “*Натисніть тут*” </A> – посилання на FTP- протокол.
<A href=’’mailto: vasia@mail.kiev.ua>*Моя пошта*</A> – посилання на адресу електронної пошти.

Створення внутрішніх посилань на Web-сторінці

Внутрішні посилання організовують переходи усередині одного HTML-документа. Вони застосовуються, коли на одній сторінці багато тексту. Для простоти навігації можна створити посилання, при натисканні клавішею миші на які користувач автоматично перейде до потрібної частини документа.

Щоб створити таке посилання, спочатку потрібно визначити місце, до якого посилання приводить. Це робиться за допомогою атрибута *name* тегу <A>. Необхідний фрагмент тексту міститься в середині тегу <A>. Хоча зовсім не обов'язково поміщати туди текст, можна просто встановити теги такого елемента в місці, до якого браузер повинен переходити при натисканні клавішею миші на посиланні. У якості значення атрибута *name* можна взяти будь-яке ім'я, бажано, щоб воно характеризувало поточне місце, так буде простіше користуватися мітками.

Потім потрібно створити посилання на цю мітку. Посилання на внутрішню мітку створюється так само, як і посилання на зовнішній документ, тільки замість URL-адреси потрібної сторінки треба ввести адресу мітки у вигляді *#met1*. При цьому *met1* — ім’я елемента прив’язки (ім'я мітки).

Тепер при натисканні клавішею миші на посиланні браузер автоматично перейде до місця, вказаного міткою. Отже, для організації внутрішніх посилань на

ції Web-сторінці використовується атрибут *name* тегу <A>. При цьому параметр *href***не вказується**.

Формат тегу:

<ANAME =“ім’я елемента прив’язки” >*текст*</**A**>.

Ім’я елемента прив’язки – це довільний текст, який не виводиться на екран браузером і позначає місце переходу. Якщо на сторінці є кілька міток, то всі вони повинні мати різні імена.

Після створення такого елемента прив’язки, на нього можна посилатись. Для такого використовують дещо інший формат тегу<AHREF>. В цьому форматі уже URL-адреса не вказується. Замість нього вказують ім’я елемента прив’язки.

Формат тегу:

<A HREF= “# ім’я елемента прив’язки”>*Елемент прив’язки*</**A**>

Під час організації внутрішніх посилань також можна застосовувати вищеназвані атрибути тегу <A>.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Таблиці та їх структура. Теги, які задають структуру таблиці.
2. Заголовок (підпис) таблиці.
3. Складові таблиці: теги рядка та комірки.
4. Параметри, які використовуються для оформлення вигляду таблиці в цілому.
5. Параметри, які використовуються для оформлення виду складових таблиці (комірок, рядків, стовпчиків).
6. Об’єднання рядків та стовпчиків.
7. Векторний формат зберігання графічної інформації.
8. Растровий формат зберігання графічної інформації.
9. Вибір формату растрових графічних файлів для документів HTML.
10. Способи розміщення графічних зображень на Web-сторінці.
11. Створення фонових графічних зображень.
12. Включення графічних зображень в HTML-документ.
13. Форматування розмірів області зображення в HTML. Рамки навколо зображень.
14. Вирівнювання зображень в HTML.
15. Відділення зображення від тексту в HTML. Рамки навколо зображень.
16. Створення маркованих списків в HTML за допомогою графічних зображень.
17. Поняття гіпертекстових посилань. Їх види.
18. Організація гіпертекстових посилань.
19. Формат тегу <A> та його параметри.
20. Створення посилань на інші Web-сторінки.
21. Створення внутрішніх посилань.
22. Графічні та текстові елементи прив’язки.
23. Організація адресної частини посилання.

КАСКАДНІ ТАБЛИЦІ СТИЛІВ (CSS)

Лекція 3. Вступ. Визначення CSS-стилю. Форматування тексту засобами CSS.

Основна мета: ознайомитися з можливостями і призначення каскадних таблиць стилів CSS. Формування у фахівця теоретичних знань і практичних навичок щодо визначення каскадних таблиць стилів та форматування тексту засобами CSS.

КАСКАДНІ ТАБЛИЦІ СТИЛІВ (CSS) 27
Лекція 3. Вступ. Визначення CSS-стилю. Форматування тексту засобами CSS. 27
Каскадні таблиці стилів, їх можливості і призначення 27
Додавання CSS-стилів в документ 28
Внутрішні стилі 28
Глобальні стилі (стилі рівня документу) 29
Визначення стилів через класи та ідентифікатори 30
Імпортування зовнішніх таблиць стилів 31
Зв'язані стилі 32
Пріоритет стилів 32
Форматування тексту засобами CSS 34
Колірне оформлення 34
Робота з шрифтами 35
Робота з текстом 36
Робота з блоками тексту 38
Визначення полів документа 40
Створення відступів та інтервалів у заголовків 41
Абзацний відступ 41
Контрольні питання 41

Каскадні таблиці стилів, їх можливості і призначення

CSS (*Cascading Style Sheets)*– це каскадні таблиці стилів, які є набором параметрів форматування зовнішнім видом і положенням елементів Web-сторінки. Або, якщо пояснити це більш зрозумілою мовою, **CSS**–це технологія опису зовнішнього вигляду сторінок, написаних на **HTML**.

Головне призначення таблиць стилів – це розділити код сторінок і її зовнішній вигляд. Доцільність такого можна пояснити на прикладі. Припустимо є сайт, на якому 100 сторінок (це зовсім небагато). Припустимо, шрифт звичайного тексту на сайті - 15pt, а потрібно змінити його на 17pt. Якщо не використовувати **CSS,** то в кожній з 100 сторінок, у кожному місці, де вставляється текст виправляти з 15pt на 17pt. Це займе кілька годин.

Якщо використовувати **CSS**, то тоді потрібно лише відкрити спеціальний файл зі стилями, знайти в цьому файлі параметр визначення розміру шрифту та усього один раз змінити з 15pt на 17pt. І сайт повністю і відразу змінить зовнішній вигляд. Є також багато можливостей створення незвичайних дизайнерських рішень за допомогою **CSS**, які неможливо реалізувати тільки за допомогою **HTML**.CSS не чутливий до регістру, переносу рядків, пробілів та символів табуляції.

Правила опису стилів записуються у форматі, який відрізняється від **HTML.** Базовим поняттям виступає **селектор** - це деяке ім'я стилю, для якого додаються параметри форматування. В ролі селектора виступають теги, класи та ідентифікатори. Загальний спосіб опису селектора має вигляд.

В загальному переваги використання **CSS** такі:

- розмежування коду і оформлення;
- різне оформлення для різних пристроїв;
- розширені способи оформлення елементів в порівнянні з **HTML;**
- прискорення завантаження сайтів;
- єдиний стиль для оформлення сукупності документів;
- централізоване зберігання.

Технологія **CSS** використовується в першу чергу дизайнерами, адже саме вони створюють дизайн сайту. Крім мобільності свого сайту, таблиці стилів дозволяють створювати різні класичні дизайнерські рішення (наприклад меню, що випадає). А разом з **JavaScript** дозволяють створювати динамічні **HTML**-сторінки (**DHTML**), краса та зручність яких, часом, просто вражають.

ДОДАВАННЯ CSS-СТИЛІВ В ДОКУМЕНТ

Існує п'ять способів задання **CSS**-стилів для тегів **HTML:**

- Перевизначення стилю всередині будь-якого тегу с використанням параметра style="…"
- Визначення в тегу <STYLE>, що міститься в заголовку документу.
- Визначення стилів через класи та ідентифікатори.
- Імпортування зовнішніх таблиць стилів.
- Посилання на стильовий файл за допомогою тегу <LINK>.

Внутрішні стилі

За допомогою *внутрішніх стилів* форматування зовнішнього вигляду елементів сторінки виконується в середині тегу: потрібно включити в тег атрибут

style зі списком властивостей та їх значень. Браузер використовує властивості стилю при виведенні вмісту тільки поточного тегу.

Наприклад, наступний стиль потребує, щоб браузер вивів заголовок першого рівня «Цей текст блакитного кольору» синім кольором і курсивом:

<H1 style=”color: blue; font-style: italic”>Цей текст блакитного кольору</H1>

Такий спосіб визначення стилю називається ***внутрішнім*** (inline): він реалізується в тегу, що міститься в документі. Діапазон дії стилю охоплює тільки цей тег. Оскільки внутрішні стилі розкидані по документу, їх підтримка може виявитися скрутною. Внутрішні стилі доцільно використовувати тільки в тих випадках, коли іншим способом не можна досягти потрібного ефекту.

Глобальні стилі (стилі рівня документу)

Реальна сила таблиці стилів проявляється при розміщенні списку правил подання в заголовок документа **HTML.** Таблиці стилів рівня документа, вкладені кожна у свою пару тегів <STYLE> … </STYLE>, впливають на всі теги в документі із указаним типом, за винятком тих, які містять перевизначений внутрішній атрибут ***style.***

Тег <STYLE> повинен бути присутнім між тегами <HEAD> документу. Все, що знаходиться між тегами <STYLE> и </STYLE>, розглядається як правила стилю, що повинні бути застосовані до документа.

Визначення стилю починається з імені тегу, для якого визначається стиль. Визначення стилю міститься в фігурних дужках, що йдуть за ім’ям тегу. Для одного тегу можна одночасно визначити декілька атрибутів. Наприклад, наступна таблиця стилів рівня документу налаштовує виведення вмісту усіх тегів <H1> блакитним кольором і курсивом:

```
<HTML>
    <HEAD>
    <TITLE>Все блакитне</TITLE>
        <STYLE type=“text/css”>
            /*зробити всі заголовки блакитними*/
            H1 {color: blue;
                font-style: italic}
        </STYLE>
    </HEAD>
    <BODY>
        <H1>Це написано блакитним кольором</H1>
        <H1>І це написано блакитним кольором</H1>
    </BODY>
</HTML>
```

Важливим атрибутом тегу <STYLE> є ***type*** Атрибут ***type*** визначає типи стилів, що містяться в тегу:

каскадні таблиці стилів **CSS** мають тип ***text/css*;**

таблиці стилів **JavaScript** використовують тип ***text/javascript.***

Крім того, що для одного тегу можна одночасно визначити декілька атрибутів, є можливість одночасно визначити параметри декількох тегів:

H1 {color: blue; font-style: italic; font-size: 20;}
H2, H3 {text-align: left; color: black; font-size: 14;}

В цьому випадку блоки тексту, що знаходяться в тегах<H2> і <H3> будуть відображатися однаково, якщо не вказані ще будь-які додаткові параметри для таких тегів.

Встановлення стилів для одного тегу можна задавати в декількох місцях:

DIV, P {font-size: 20;}
P, H1 {color: red;}

Визначення стилів через класи та ідентифікатори

Класи. В тому випадку, коли необхідно для одного й того ж тегу визначити декілька видів відображення символів, потрібно використовувати класи тегів. В такому випадку в блоці визначення стилів необхідно описати ці класи. Наприклад:

```
<HTML>
    <HEAD>
    <TITLE>Визначення класів</TITLE>
        <STYLE>
            p.big {
                    text-align: center;
                    font-style: italic;
                    font-size: 40;
                  }
            p.middle {
                    text-align: left;
                    color: green;
                    font-size: 30;
                  }
            p.small {
                    text-align: right;
                    text-decoration: underline;
                    font-size: 20;
                  }
        </STYLE>
    </HEAD>
    <BODY>
        <P class=”big”>Великий абзац</P>
        <P class=”middle”>Середній абзац</P>
        <P class=”small”>Маленький абзац</P>
    </BODY>
</HTML>
```

Якщо потрібно створити клас, який не належить жодному з тегів, необхідно в описі класу опускати визначення тегу, наприклад:

```
<STYLE>
    .green {
            color: green;
          }
    red   {
            color: red;
          }
</STYLE>
```

а потім в самому документі використовувати будь-які теги з атрибутом **class="…"**

```
<DIV class="green">Блок тексту зеленого кольору</DIV>
<P class="red">Абзац тексту червоного кольору </P>
```

***Ідентифікатори*.** Крім класів, стилі можна описувати з використанням ідентифікатора **{#id}.** Наприклад, можна визначити ідентифікатор, який змінює гарнітуру шрифту на **Arial**, а потім в тегу вказати атрибут:

```
<HTML>
    <HEAD>
    <TITLE>Використання ідентифікаторів</TITLE>
        <STYLE>
                #idFontArial {
                                font-family: Arial;
                                }
        </STYLE>
    </HEAD>
    <BODY>
        <P id=idFontArial>Шрифт з гарнітурою Arial</P>
    </BODY>
</HTML>
```

Імпортування зовнішніх таблиць стилів

В таблицю глобальних стилів базового HTML-документа можна імпортувати вміст іншого CSS-файлу за допомогою команди **@import**.

```
<HTML>
    <HEAD>
    <TITLE>Імпортовані таблиці стилів</TITLE>
        <STYLE>
            @import url(http://www.oreily.com/styles/spec_styles.css);
            @import url("my.css");
```

```
        </STYLE>
    </HEAD>
    <BODY>
        …
    </BODY>
</HTML>
```

Команда **@import** приймає один параметр **URL,** в якому вказано мережний шлях до зовнішньої таблиці стилів. **URL** може вказувати як абсолютний, так і відносний до базового документу шлях.

Зв'язані стилі

При використанні зв'язаних стилів опис селекторів та їх значень розміщується в окремому файлі, з розширенням CSS, а для зв'язування з цим файлом застосовується тег <LINK> з атрибутом **href="style.css"**: **<LINK href="style.css">.** Цей тег необхідно розташувати в заголовку документа в контейнер <HEAD>. В цьому тегу також можуть міститися атрибути **type="text/css"** та **rel="stylesheet".** Атрибут **type="…"** призначено для визначення мови листа стилів, а атрибут **rel="…"** показує браузеру, що посилання **href="…"** вказує на файл, що містить визначення стилів.

```
<HTML>
    <HEAD>
    <TITLE>Посилання на стилі</TITLE>
            <LINK rel="stylesheet" type="text/css" href="style.css">
    </HEAD>
    <BODY>
        …
    </BODY>
</HTML>
```

Сам стильовий файл містить тільки опис тегів, наприклад:

```
P    {
       text-indent: 25;
       font-size: 20;
       color: black;
       }
  H1 {text-align: center;
       color: black;
       }
```

Пріоритет стилів

Одним з ключових слів CSS є **каскад**. Під каскадом в даному випадку розуміється одночасне застосування різних стильових правил до елементів документа - за допомогою підключення декількох стильових файлів, наслідування властиво-

стей та інших методів. Щоб в подібній ситуації браузер розумів, яке в підсумку правило застосовувати до елементу, і не виникало конфліктів в поведінці різних браузерів, введені деякі пріоритети.

Нижче наведені пріоритети браузерів, якими вони керуються при обробці стильових правил. Чим вище в списку знаходиться пункт, тим нижче його пріоритет, і навпаки.

- Стиль браузера.
- Стиль автора.
- Стиль користувача.
- Стиль автора з додаванням !important.
- Стиль користувача з додаванням !important.

Найнижчим пріоритетом володіє стиль браузера - оформлення, яке за замовчуванням застосовується до елементів веб-сторінки браузером. Це оформлення можна побачити у випадку «голого» HTML, коли до документу не додається ніяких стилів.

Щодо правил застосування пріоритетів згідно синтаксису CSS, то пріоритети в порядку зростання наступні:

1. Зв’язаний стиль.
2. Імпортований стиль.
3. Глобальний стиль.
4. Внутрішній стиль.

Таким чином, **внутрішній стиль** має найбільший пріоритет. І можна зробити висновок, що в прикладі колір елементу буде червоним, а не чорним.

Які висновки можна зробити?

1. **Загальний стиль** для всього сайту повинен бути винесений в окремий файл і підключатися на кожній сторінці через тег <LINK>**.** Через те, що даний стиль є стилем з мінімальним пріоритетом, його в окремих випадках можна буде змінити.
2. **Імпортований стиль** треба використовувати, коли **2 і більше сторінок (але не всі)** мають певні особливості в стилі.
3. **Впроваджений стиль** треба використовувати для визначення унікальних **CSS**-стилів для конкретної сторінки. Ці стилі унікальні для кожної сторінки сайту.
4. **Внутрішній стиль** треба використовувати, коли окремий елемент на окремій сторінці вимагає особливого вигляду.

ФОРМАТУВАННЯ ТЕКСТУ ЗАСОБАМИ CSS

Колірне оформлення

Для визначення колірного оформлення тексту використовуються чотири параметра:

Атрибут **color** дозволяє встановити колір шрифта. Він має такий синтаксис:

color: «Колір»;

Колір може бути задано одним з п'яти способів:

- іменем кольору (**blue, white** або **bisque**);
- значенням виду **#{R}{G}{B}**, де **R** — червона складова, **G** — зелена, а **B** — синя, причому всі складові повинні бути однозначними шістнадцятковими числами. Приклад: **#F00** — це червоний колір;
- значенням, аналогічним попередньому, але вже виду **#{RR}{GG}{BB}**, тобто кожна складова має вже по два знаки. Приклад: **#00FF00** — зелений колір;
- значенням **rgb({R},{G},{B})**, де кожна складова визначається десятковим числом. Приклад: **rgb(255,0,0)** — червоний колір;
- аналогічно попередньому, але кожне окреме значення виражає частку в загальному кольорі. Приклад: **rgb(100%,0%,0%)** — знову ж червоний колір.

Атрибут **color** теж корисно використовувати для визначення кольору посилання в документі. Якщо сторінка є, наприклад, статтею і колір тексту чорний, то посилання можна зробити синіми або сірими, тоді вони не будуть сильно впадати в око та у той же час будуть виділятися в основному тексті. Визначення властивостей відображення для посилань нічим не відрізняється від визначення властивостей відображення для основного тексту, в CSS виділяють чотири типи посилань згідно відображення:

- **link** — усі посилання на сторінці, які користувач ще не відвідав;
- **visited** — усі посилання на сторінці, які користувач відвідав;
- **active** — усі активні посилання;
- **hover** — посилання, над яким перебуває курсор миші.

Щоб встановити тип посилання, до якого застосовуються властивості форматування, потрібно вказати без пробілів символ **:** і назву типу посилання. Наприклад, щоб колір посилання на сторінці мінявся на сірий, коли користувач наводить на нього курсор миші, потрібно використовувати наступний код:

```
a:hover {color: grey;}
```

Атрибут **background-color** встановлює колір фону тексту. Допускається або колір, або значення **transparent** що робить фон прозорим. Має такий синтаксис:

background-color: transparent|«Колір»;

Приклад:

h1 {background-color: blue;}

Атрибут **background-image** вказує на малюнок, який необхідно використовувати як фон тексту. Допускається або адреса, або значення **none,** що відключає фонову заливку. Має такий синтаксис:

background-image: none|«Адреса файлу малюнка»;

Приклад:

p {background-image: url(http://www.my.com.image.gif);}

Атрибут **background-attachment** має такий синтаксис:

background-attachment: scroll|fixed;

Значення **scroll** включає прокрутку фонового малюнка разом з вмістом сторінки**,** а **fixed** відключає.

Робота з шрифтами

Для форматування тексту в **CSS** використовується декілька параметрів.

Атрибут**font-family** визначає накреслення шрифтів для тексту. Він має такий синтаксис:

font-family: “«**Ім'я шрифту»|«Ім'я сімейства шрифтів»,...**”**;**

Ім'я шрифту визначається у вигляді “Arial”, “Times New Roman”. При визначеннітакого атрибута необхідно, щоб на комп'ютері клієнта такі шрифти було встановлено.

Крім імені конкретного шрифту можна встановити ім'я одного із сімейств шрифтів, що включають цілі набори сумісних шрифтів. Таких сімейств п'ять: “serif” (шрифти із зарубками: Times New Roman, Garamond, Georgia), “sans-serif” (шрифти без зарубок: Trebuchet, Arial, Verdana), “cursive”, “fantasy” і “monospace” (моноширинні шрифти: Courier, Courier New, Andala Mono).

Допускається задавати кілька шрифтів через кому. У цьому випадку спочатку на клієнтському комп'ютері шукається перший шрифт зі списку, у випадку невдалого пошуку — другий, потім третій і т.д. Якщо жоден шрифт, визначений у властивості, не знайдений, використовується шрифт за замовчуванням.

Наприклад:

Div.fixedwidth {font-family: “Verdana”, “Arial”, “sans-serif”;}

Якщо атрибут **font-family** визначений у вбудованому стилі, назви шрифтів беруться в апострофи, а не лапки:

<P style=“**font-family: ‘Verdana’, ‘Arial’, ‘sans-serif**’”**>**

Атрибут **font-style** визначає накреслення шрифту.

font-style: normal|italic|oblique;

Доступні три значення, що визначають звичайний шрифт (**normal,** це значення за замовчуванням), курсив (**italic**) і похиле накреслення (**oblique**).

Атрибут **font-size** визначає розмір шрифту.

font-size: «Абсолютний розмір у пунктах» pt|xx-small|x-smail|small| medium|large|x-large|xx-large|larger|smaller;

Розмір шрифту може бути заданий в абсолютних і відносних величинах. Абсолютні величини можуть бути задані як в пунктах так і у вигляді імені розміру; доступно сім різних значень імен розмірів від **xx-small** до **xx-large**. Значенням за замовчуванням є **medium**. Слід зауважити, що значення імен розмірів залежать від настроювань браузерів та операційної системи. Як відносні величини можна встановити відсоток від величини шрифту батьківського елемента (наприклад, 120%) або значення **larger** і **smaller**, що визначають наступний розмір шрифту відповідно по зростанню та зменшенню. Так, наприклад, якщо для батьківського тегу визначений шрифт розміру **medium**, то значення **larger** установить для поточного елемента шрифт **large**.

Атрибут **font-weight** встановлює жирність шрифту.

font-weight: normal|bold|bolder|lighter|100|200|300|400|500|600|700|800|900;

Тут доступні сім абсолютних значень від 100 до 900, що відображають різну жирність шрифту, при цьому звичайний шрифт буде мати жирність **400** (або **normal),** а напівжирний — **700** (або **bold).** Значенням за замовчуванням є**normal** (або **400).** Значення **bolder** і **lighter** відносні й визначають наступні ступені жирності відповідно в більший та менший бік від базового.

Атрибут **font-variant** визначає, як будуть виглядати великі й маленькі літери шрифту.

font-variant: normal|small-caps;

Значення **small-caps** визначає таку поведінку шрифту, коли його маленькі букви мають той самий вигляд, як великі, просто меншого розміру. Значення за замовчуванням — **normal**.

Робота з текстом

Для роботи з текстом і абзацами призначені такі параметри.

Атрибут **text-indent** визначає величину відступу першого рядка блоку тексту.

text-indent: «Абсолютне або відносне значення відступу»;

Тут допускаються абсолютні й відносні (щодо ширини текстового елемента), у тому числі відсоткові, величини відступу. За замовчуванням значення відступу дорівнює нулю, тобто відсутній.

Наприклад:

P { text-indent: 50;}

визначає відступ в 50 пікселів. Можна також визначати відступ в сантиметрах (cm), дюймах (in) та відсотках (%).

Атрибут **text-align** визначає горизонтальне вирівнювання тексту.

text-align: left|right|center|justify;

Тут доступні значення **:**

left (ліве вирівнювання; поведінка за замовчуванням),
right (праве вирівнювання),
center (центрування) і
justify (вирівнювання по ширині).

Атрибут **text-height** визначає вертикальну відстань між двома рядками (вірніше, між базовими лініями двох рядків).

text-height: normal|«Абсолютне або відносне значення»;

Тут допускаються абсолютні й відносні (щодо висоти шрифту), у тому числі відсоткові, величини відстані.

Усі абсолютні значення при цьому множаться на висоту поточного шрифту а потім застосовуються. Таким чином, щоб зробити стандартний відступ у два інтервали, можна просто написати **text-height:** 2. Значення **normal** встановлює звичайну величину відстані між рядками; це значення властивості за замовчуванням.

Атрибут **vertical-align** визначає вертикальне вирівнювання тексту.

vertical-align: baseline|sub|super|top|text-top|middle|bottom|text-bottom;

Можливі вісім визначених значень:

baseline (значення за замовчуванням) — визначає вирівнювання базової лінії елемента сторінки по базовій лінії батька;
sub — перетворює текст у верхній індекс;
super — перетворює текст у НИЖНІЙ індекс;
top — вирівнює верх елемента сторінки по самому верхові батька;
text-top — вирівнює верх тексту елемента сторінки по верхові тексту батька;
middle — вирівнює центр елемента сторінки по центру батька;
bottom — вирівнює низ елемента сторінки по низу батька;
text-bottom— вирівнює низ тексту елемента сторінки по низу тексту батька.

Атрибут **text-decoration** визначає ефекти, які будуть застосовані до тексту: підкреслення, надкреслення, закреслювання або мерехтіння.

text-decoration: none|underline|overline|line-through|blink;

Тут доступні п'ять різних значень:

none — відміняє всі ефекти (це поведінка за замовчуванням);
underline — підкреслює текст;

overline — "надкреслює", якщо так можна виразитися, текст, тобто проводить лінію над рядками;
line-through — закреслює текст;
blink — змушує текст мерехтіти.

Атрибут**text-transform** змінює регістр символів тексту.
text-transform: capitalize|uppercase|lowercase|none;
Цей атрибут дозволяє
uppercase — перетворити всі літери виділеного тексту в прописні;
lowercase — перетворити всі літери виділеного тексту в рядкові
capitalize —перетворити в прописну першу букву кожного слова;
none —не перетворювати текст взагалі**(**це поведінка за замовчуванням).

Атрибут **direction** керує напрямком виведення тексту. Він може приймати значення:
ltr — напрямок зліва направо (за замовчуванням);
rtl — напрямок справа наліво.

Атрибут **white-space** визначає правила відображення пробілів між словами**.**

white-space: nomal|pre|nowrap;

Тут доступні три значення:
nomal — текст у вікні браузера відображається як у звичаному текстовому редакторі, переноси рядків встановлюються автоматично(це установка за замовчуванням);
pre — текст відображається з врахуванням всіх пробілів. Визначає таку поведінку тексту, немов він укладений в теги <**PRE**>і</**PRE**>;
nowrap — пробіли не враховуються, діє аналогічно тегам <**NOWRAP**>і</**NOWRAP**>

Робота з блоками тексту

В CSS визначено атрибути, які дозволяють встановити такі характеристики текстового абзацу, які немає у звичайному **HTML**, наприклад прикрасити його рамкою та встановити відступи.

Чотири атрибути **margin-left, margin-top, margin-right** і **margin-bottom** дозволяють установити додаткові відступи між зовнішньою границею елемента сторінки і його сусідами відповідно ліворуч, зверху, праворуч і знизу. Потрібно встановити абсолютне або відносне (в тому числі відсоткове) значення відступу; у випадку відсоткового значення за основу береться ширина батьківського елемента. Допускаються також від'ємні значення.
Розглянемо приклад — невелике правило CSS, яке встановлює значення ширини верхнього поля, рівне **2 em**. Одиниця виміру **em** визначається щодо розміру шрифту елемента: **1 em** дорівнює розміру використовуваного шрифту.

H1 { margin-top: 2em }

Докладніше розберемо скорочений запис таких властивостей — **margin**. Не дуже зручно щораз набирати наступний код:

```
body { margin-top: 2em;
       margin-right: 2em;
       margin-bottom: 2em;
       margin-left: 2em;
       }
```

Якщо властивість **margin** приймає тільки одне значення, то воно застосовується до всіх полів. Якщо властивість приймає два значення, то перше застосовується до верхнього та нижнього полів, а друге — до правого й лівого. Якщо ж воно приймає три значення, то перше значення застосовується до верхнього, друге — до лівого та правого, а третє — до нижнього поля. І, нарешті, якщо воно приймає чотири значення, то вони застосовуються до верхнього, правого, нижнього й лівого полів відповідно.

Розглянемо такий приклад:

```
body { margin: lem 2em 3em }
```

Ширина верхнього, правого, нижнього та лівого полів сторінки рівна 1 em, 2 em, 3 em і 2 em відповідно.

Крім створення зовнішніх границь є можливість встановити величини внутрішніх границь для об’єкту. Атрибути **padding-left, padding-top, padding-right і padding-bottom** задають проміжок відповідно ліворуч, зверху, праворуч і знизу між елементом і рамкою, якщо вона є. Також допускаються абсолютні й відносні (у тому числі відсоткові) значення відступу; у випадку відсоткового значення за основу береться ширина батьківського елемента. Від’ємні значення не допускаються.

Властивість **padding** працює, як і скорочений запис для полів. Якщо властивість приймає тільки одне значення, то воно застосовується до всіх відступів. Якщо властивість приймає два значення, то перше застосовується до верхнього та нижнього відступів, а друге — до правого та лівого відступів. Якщо воно приймає три значення, то перше значення застосовується до верхнього, друге — до лівого та правого, а третє — до нижнього відступу. Якщо властивість приймає чотири значення, то вони застосовуються до верхнього, правого, нижнього та лівого відступів відповідно.

Фон відступів буде використовувати колір або зображення, задані цьому елементу за допомогою властивості **background**:

```
H1 {
background: white;
padding: lem 2em:
}
```

Елемент **HTML**-документу можна помістити в рамку. Для цього використовуються параметр **border.**

Атрибути **border-left-width, border-top-width, border-right-width** і **border-bottom-width(border-width)** задають товщину лінії рамки.

border-left-width|border-top-width|border-right-width|border-bottom-width: thin|medium|thick|«Товщина»;

Тут можна встановити або числове значення товщини рамки в пікселях або інших одиницях виміру, або одне з конкретних значень: **thin** (тонка), **medium** (середня) або **thic** (товста). За замовчуванням лінія рамки середня, тобто **medium**.

Атрибути **border-left-color, border-top-color, border-rigid.-color** і **border-bottom-color(border-color)** задають колір лінії рамки.

border-left-color|border-top-color|border-right-color|border-bottom-color: «Колір»;

За замовчуванням значення таких атрибутів дорівнюють значенню атрибута **color** або кольору за замовчуванням, якщо атрибут **color** не визначений.

Атрибути **border-left-style, bcrder-top-style, border-right-style і border-bottom-style(border-style)** задають стиль лінії рамки або її відсутність.

border-left-style|border-top-style|border-right-style|border-bottom-style: none|dotted|dashed|sol id|double|groove|ridge|inset|outset;

Тут доступні такі значення:

none — взагалі не показує ніяку рамку, незалежно від установок інших атрибутів (це поведінка за замовчуванням);
dotted — показує пунктирну (крапкову) лінію;
dashed — показує штрихову лінію;
solid — показує суцільну лінію;
double — показує подвійну лінію. Сума товщин двох ліній, що складають її, і проміжку між ними дорівнює значенню відповідного атрибуту **border-***~width**;
groove — показує втиснену тривимірну лінію;
ridge — показує опуклу тривимірну лінію;
inset — показує тривимірну опуклу східчасту лінію;
outset — показує тривимірну втиснену східчасту лінію.

Визначення полів документа

WEB-стрінки мають більш привабливий вигляд, якщо в них є поля. Для визначення розмірів лівого і правого полів необхідно використовувати такі атрибути, як **margin-left** і **margin-right.** Наприклад:

```
<STYLE type=”text/css”>
    body { margin-left: 10%; margin-right 10%;}
</STYLE>
```

Створення відступів та інтервалів у заголовків

Праві і ліві відступи у заголовків (і абзаців) створюються за допомогою тих же атрибутів **margin-left** і **margin-right,** що і поля документу. Але їх потрібно вказувати в тегах заголовків. Допускають від'ємні значення. Наприклад:

```
STYLE type="text/css">
    body { margin-left: 10%; margin-right 10%;}
    h1 { margin-left: 20%; }
    h2, h3 { margin-left: -8%; }
</STYLE>
```

Для верхніх і нижніх відступів заголовків і абзаців використовуються атрибути **margin-top**і**margin-bottom**, що контролюють верхні і нижні відступи відповідно. Наприклад:

```
STYLE type="text/css">
    p { margin-top: 5; margin-bottom 3;}
</STYLE>
```

Якщо визначені відступи для абзацу і заголовка, і в тексті документу абзац йде за заголовком, то відступи між ними не сумуються, а використовується максимальне значення відступу.

Абзацний відступ

Атрибут **text-indent** визначає відступ для "червоного рядка".

text-indent: «Абсолютне або відносне значення відступу»;

Тут допускаються абсолютні й відносні (щодо ширини текстового елемента), у тому числі відсоткові, величини відступу. За замовчуванням відступ "червоного рядка" дорівнює нулю, тобто відсутній.

Наприклад:

P { text-indent: 50;}

визначає відступ "червоного рядка" в 50 пікселів. Можна також визначати відступ в сантиметрах (cm), дюймах (in) та відсотках (%).

КОНТРОЛЬНІ ПИТАННЯ

1. Призначення і можливості каскадних таблиць стилів.
2. Способи визначення **CSS**-стилю для тегів **HTML.**
3. Внутрішні стилі.
4. Глобальні стилі.
5. Визначення стилів через класи.
6. Визначення стилів через ідентифікатори.
7. Імпортування зовнішніх таблиць стилів.
8. Стиль з файлу.
9. Пріоритет стилів.
10. Як зробити колірне оформлення тексту?

11.Як відформатовати шрифти?
12.Як встановити розмір та гарнітуру шрифту?
13.Як встановити курсив та напівжирне накреслення шрифту?
14.Як створити абзацний відступ?
15.Як зробити горизонтальне і вертикальне вирівнювання тексту?
16.Як встановити вертикальну відстань між двома рядками?
17.За допомогою якого атрибута змінити регістр символів тексту?
18.Як підкреслити текст та закреслити його?
19.Робота з блоками тексту.
20.Як встановити поля документу?
21.Як встановити відступи та інтервали у заголовків?

КАСКАДНІ ТАБЛИЦІ СТИЛІВ (CSS)

Лекція 4. Оформлення HTML-документів засобами CSS

Основна мета: Формування у фахівця теоретичних знань і практичних навичок щодо оформлення HTML–документів засобами CSS.

КАСКАДНІ ТАБЛИЦІ СТИЛІВ (CSS) 43
Лекція 4. Оформлення HTML-документів засобами CSS 43
Оформлення HTML-документів засобами CSS 43
Позиціонування елемента 43
Генерований вміст 46
Автоматична нумерація і лічильникки 47
Форматування списків 48
Форматування таблиць 49
Колонки 49
Ширина таблиці 50
Висота таблиці 50
Положення та вирівнювання заголовка 50
Вертикальне вирівнювання 51
Горизонтальне вирівнювання 51
Динамічні ефекти рядків і колонок 51
Границі 51
Стилі границь 52
Порожні коміркі 53
Псевдостилі гиперпосилань 53
Контрольні питання 53

Оформлення HTML-документів засобами CSS

Позиціонування елемента

В **HTML** не можна взяти елемент сторінки — фрагмент тексту, малюнок або таблицю — тобто, блочний елемент та помістити його в довільне місце сторінки. Каскадні таблиці стилів подолали цей недолік **HTML.** Вони надали можливість помістити будь-який **HTML**-елемент в довільне місце WEB-сторінки, вказавши координати і геометричні розміри.

Каскадні таблиці стилів дозволяють автору WEB-сторінки розміщувати її елементи (абзац, малюнок, список і т.п.) в будь-якому місці документу, тобто існує можливість довільного позиціонування об’єкту. Щоб позиціонувати рядок тексту, потрібно помістити його в парний тег абзацу**<P>, </P>.** Щоб позиціювати малюнок, що є вбудованим елементом, потрібно помістити його в блочний тег. Найкраще — помістити його в парний тег **<DIV>, </DIV>.** Позиціонування здійснюється за допомогою спеціальних атрибутів.

Атрибут **position** визначає, відносно чого відраховуються координати елемента.

position: absolute|relative|static;

Тут доступні три значення:

- **absolute** — визначає *абсолютне позиціонування*, тобто, координати відлічуються відносно лівого верхнього кута батьківського елемента;
- **relative** — визначає *відносне позиціонування*, тобто, координати елемента відлічуються відносно того положення, в якому він знаходився. У цьому випадку допускаються від'ємні координати об'єкта;
- **static** — є значенням за замовчуванням і визначає звичайне положення **HTML**-елемента, коли він відображається без вільного позиціонування усередині "потоку" тексту. І в цьому випадку значення інших атрибутів вільного позиціонування не мають сенсу.

Наступна пара атрибутів, які потрібно знати, — це **left** і **top.** Вони задають відповідно горизонтальну і вертикальну координати лівого верхнього кута елемента WEB-сторінки. Можна вказати як абсолютне значення в одній з доступних одиниць вимірювання, так і відсоткове щодо ширини (для властивості **left)** і висоти (для властивості **top)** батьківського елемента. Допускаються негативні значення; у цьому випадку елемент може повністю або частково виявитися за межами вікна. Також доступне значення **auto,** що визначає таку поведінку елемента сторінки, коли браузер сам визначає, де його відобразити (тобто, вільне позиціонування не працює), **auto** є значенням за замовчуванням.

```
<HTML>
    <HEAD>
    <TITLE>Позиціювання елементів</TITLE>
        <STYLE>
            .class1 {position: absolute; top: 50; left: 100;}
            .class2 {position: absolute; top: 100; left: 80;}
            .class3 {position: relative; top: 20; left: 30;}
        </STYLE>
    </HEAD>
    <BODY>
        <H1 class=”class1”> CSS в дії</H1>
        <DIV class=”class2”>
            <P>Параграф № 1</P>
            <P class=”class3”>Параграф № 2</P>
        </DIV>
    </BODY>
</HTML>
```

Зауваження: Зсув **Параграф** № 1 відбувається через те, що він є дочірнім елементом тегу **<DIV>,** який зсовується щодо своєї первісної позиції. Зсув **Пара-**

граф № 2 відбувається через накладення зсувів тегу **<DIV>**і зсуву самого параграфа.

Друга пара атрибутів, яку потрібно знати, — це **width**і **height**. Вони задають відповідно ширину та висоту елемента WEB-сторінки. Тут доступні ті ж самі значення, що і для попередньої пари атрибутів.

Наступний важливий атрибут називається **z-index** і встановлює порядок, в якому вільно позиціоновані елементи перекриватимуть один одного. Тут доступні як числове значення, так і значення **auto,** що дає можливість самому браузеру задавати порядок перекриття елементів (звичайно визначені пізніше елементи перекривають визначені раніше). Значення **z-index**у батька дорівнює нулю, елементи з додатним значенням **z-index** перекривають батька, причому зверху опиняється елемент із максимальним значенням цієї властивості. Відповідно, елементи з від'ємним значенням **z-index** перекриваються батьком, причому самим нижнім опиняється елемент із мінімальним значенням цієї властивості. Значення за замовчуванням — **auto**.

І ще один атрибут, який може знадобитися, — це **overflow**. Він визначає поведінку блочного елемента, чий зміст вилазить за його межі.

overflow: visible|hidden|scroll|auto;

Тут доступні чотири значення:

- **visible** — дозволяє розширити блочний елемент так, щоб весь його вміст відобразився в ньому повністю. Це поведінка за замовчуванням;
- **hidden** — Відображає лише область всередині блочного елемента. При цьому користувач ніяк не зможе переглянути те, що приховано від його очей;
- **scroll** — аналогічно попередньому значенню, але передбачає смуги прокручування;
- **auto** — Відображає область всередині блочного елемента згідно настроювань браузера. Як правило, аналогічно **scroll.**

І знову ж, цей атрибут діє тільки тоді, коли **position** установлений в **absolute**. А якщо ні, то діє значення за замовчуванням, тобто **visible**.

Останній атрибут — це атрибут **visibility**, що встановлює видимість або невидимість елемента.

visibility: inherit|visible|hidden;

Тут доступні три значення:

- **inherit** — визначає спадкування видимості від батька. Тобто, якщо батько є видимим, дочірній елемент також є видимим. Це значення за замовчуванням;
- **visible** — визначає видимість елемента незалежно від видимості батька;
- **hidden** — приховує елемент також незалежно від видимості батька.

Генерований вміст

У деяких випадках може виникнути необхідність у тому, щоб браузер користувача відображав на екрані вміст, який не належить так званому «дереву» **HTML**-документа. Гарним прикладом може служити нумерований список — не вводити числа нумерації вручну. Також зручніше працювати, якщо, приміром, браузер сам додає слово «Малюнок» і номер малюнка перед його назвою або, наприклад, вставляє напис «Розділ 9.» перед заголовком дев'ятого розділу.

На відміну від **HTML CSS** надає можливості для вирішення даних проблем. За допомогою **CSS** можна генерувати вміст декількома способами:

- з використанням властивості **content** у комбінації із псевдоелементами**: before** і **:after;**
- з використанням елементів, властивість **display** яких приймає значення **list-item.**

Властивість **content.** Ця властивість використовується разом із псевдоелементами**: before** і **:after**для генерації вмісту в документі. За допомогою псевдоелементів**: before** і **:after** можна встановити стиль і місце розташування генерованого вмісту. Ці псевдоелементи визначають місце розташування «перед» і «після» вмісту елемента, що належить дереву HTML-документу. Використовуючи ці псевдоелементи в комбінації із властивістю **content**, можна визначити вміст і місцерозташування генерованого вмісту.

Розглянемо простий приклад. Нехай браузер сам розставить крапки наприкінці абзаців на сторінці:

```
p:after { content: ”.”}
```

В документ може бути вставлений звичайний текстовий рядок, який обов'язково повинен бути взято в лапки. Для прикладу додамо перед усіма абзацами напис **Абзац:**:

```
p:before { content: "Абзац:"}
```

Можна вставляти в генерований вміст переходи на новий рядок за допомогою послідовності **\A.** Вставка даної послідовності приводить до примусового розриву рядка, аналогічному тому, який утворюється при використанні елемента **BR.** Скорегуємо наш приклад:

```
p:before { content: "Абзац:\A"}
```

Тепер перед кожним абзацом буде доданий рядок **Абзац:**, а сам абзац буде починатися з наступного рядка.

Треба пам'ятати, що псевдоелементи**: before** і **:after** можуть бути пов'язані з елементом HTML-документу. Тоді вони будуть успадковувати його властивості. Приклад:

p:before {content: open-quote; color: red}

Тепер перед кожним елементом **P** буде додана лапка червоного кольору, а шрифт лапки буде такий же, який заданий усьому абзацу.

Автоматична нумерація і лічильники

В CSS існує дві властивості для керування нумерацією: **counter-increment** і **counter-reset**. Лічильники, які визначені даними властивостями, використовуються функціями **counter ()** і **counters ()** властивості **content**. Розглянемо докладно властивості для керування нумерацією.

counter-increment — за допомогою цієї властивості можна встановити одне або кілька імен лічильників, після кожного з яких може бути указане ціле число. Воно визначає величину, на яку збільшується вміст лічильника при кожному його використанні. За замовчуванням значення лічильника збільшується на одиницю. Також можна використовувати негативні цілі числа.

counter-reset — також містить список з одного або декількох імен лічильників, після кожного з яких може бути указане ціле число. Воно визначає значення, яке спочатку привласнюється лічильнику. За замовчуванням значення рівне 0.

Для прикладу розглянемо таблицю стилів, яка нумерує розділи та підрозділи в такий спосіб:

- **Розділ 1.**
  - **1.1.**
  - **1.2.**
- **Розділ 2.**
  - **2.1.**
  - **2.2. и т. д.**

Код таблиці такий:

```
h1:before {
    content: "Розділ " counter(chapter) ".";
    counter-increment: chapter; /* Додає 1 до номеру розділу */
    */ counter-reset: section;/* Визначення значення 0 для разділу */
}
h2:before {
    content: counter(chapter) "." counter(section) ".";
    counter-increment: section;
}
```

У прикладі не вказана цифра, з якої починати нумерацію, тому що за замовчуванням вона починається з одиниці.

Якщо встановити збільшення або скидання лічильника, а також використовувати його за допомогою властивості **content**, то спочатку лічильник буде збільшений або скинутий, а потім застосується в документі.

Важливо пам'ятати, що властивість **counter-reset** підтримує правила каскаду. За такими правилами в наступній таблиці стилів скидається тільки лічильник **imagenum:**

h1{ counter-reset: section -1 } h1 { counter-reset: imagenum 99 }

Щоб виконати скидання обох лічильників, необхідно встановити їх разом:

h1 { counter-reset: section -1 imagenun 99 }

Форматування списків

Розглянемо можливості визначення спискам різного візуального форматування.

Атрибут **list-style-type** визначає вид маркера списку.

list-style-type: disc|circle|square|decimal|lower-roman|upper-roman|lower-alpha |upper-alpha|none;

Доступні такі значення:

- **disc** — позначає кожну позицію списку чорним колом (це поведінка за замовчуванням);
- **circle** — позначає кожну позицію світлим кодом;
- **square** — позначає кожну позицію світлим квадратиком;
- **decimal** — нумерує кожну позицію арабськими цифрами;
- **decimal-leading-zero** — нумерує кожну позицію арабськими цифрами, доповненими нулями (наприклад, 01,02,03...98,99);
- **lower-roman** — нумерує кожну позицію маленькими римськими цифрами;
- **upper-roman** — нумерує кожну позицію великими римськими цифрами;
- **lower-alpha** — позначає кожну позицію рядковими (маленькими) латинськими буквами;
- **upper-alpha** — позначає кожну позицію прописними **(**великими) латинськими буквами;
- **lower-greek** — позначає кожну позицію рядковими (маленькими) грецькими буквами;
- **none** — взагалі ніяк не позначає позиції списку.

Атрибут **list-style-image** дозволяє перекрити визначення, задані попередньою властивістю, і встановити як маркер списку будь-яке графічне зображення.

list-style-image: «Адреса файлу графічного зображення»|none;

Наприклад:

UL { list-style-image: url(”http://nry_site.com/my_marker.jpg”) }

Зауваження: якщо встановити як значення такого атрибута будь-яку інтернет-адресу, визначення **list-style-type** будуть перекриті. За замовчуванням значення такого атрибута дорівнює **none.**

Атрибут **list-style-position** дозволяє встановити більш компактне відображення списку.

list-style-position: inside|outside;

Значення за замовчуванням — **outside**. Якщо встановити **inside**, то список буде відображатися більш компактно, як це буває при використанні атрибута **COMPACT.**

Форматування таблиць

Таблиці — одне з головних засобів розташування елементів на сторінці. Вони дозволяють розташувати меню поруч із текстом і можуть підбудовуватися під монітор комп'ютера клієнта.

Колонки

В **CSS** комірки таблиці можуть належати одному із двох типів груп: рядкам або колонкам. В **HTML** комірки є елементами, що успадковують рядкам, а не колонкам. Проте на деякі властивості комірок впливають властивості колонок.

Наступні **властивості можуть бути задані колонкам** таблиці та, відповідно, впливати на вид комірок.

- **border** — визначає різні властивості границі колонкам, але працює, тільки якщо для властивості **border-collapse** в елементі таблиці встановлене значення **collapse**.
- **background** — визначає фон для комірок у колонці.
- **width** — визначає ширину колонки.
- **visibility** — якщо властивості колонки **visibility** присвоєно значення **collapse**, те жодна комірка колонки не подається, а комірки, що містяться в інших колонках, урізуються. Крім того, ширина таблиці зменшується на ширину цієї колонки. Інші значення властивості **visibility** не мають впливу на відображення.

Розглянемо кілька рядків **CSS** -коду, що визначають властивості колонок:

col.totals { background: blue } table { table-layout: fixed } col.totals { width: 5em }

Перше правило виділяє колонку класу **totals** синім кольором, а останні два правила вказують спосіб фіксування розміру колонки, використовуючи фіксоване розташування.

Ширина таблиці

Для керування відображенням ширини таблиці при розміщенні її комірок, рядків і колонок використовується властивість **table-layout**. Вона може приймати наступні значення.

- **fixed** — горизонтальне положення таблиці не залежить від вмісту комірок; воно залежить тільки від ширини таблиці, ширини колонок і відстані між границями або комірками. Ширина таблиці може вказуватися явно за допомогою властивості або може дорівнювати **auto**.
- **auto** — браузер автоматично розміщає таблицю. Ширина таблиці визначається шириною колонок і відстанню між границями.

Висота таблиці

Висоту таблиці можна встановити, використовуючи властивість **height** для елемента **TABLE**. Як і для ширини, можна залишити вибір висоти браузеру, використовуючи значення **auto**, або встановити значення числом і вказати одиницю виміру.

Використовуючи властивість **height**, можна також задавати висоту поля елемента **table-row**, тобто висоту рядків.

Положення та вирівнювання заголовка

Властивість **caption-side** визначає положення поля заголовка щодо поля таблиці. Список його значень, які вказують, що поле заголовка розташовується:

- **top** — над полем таблиці;
- **bottom** — нижче поля таблиці;
- **left** — ліворуч від нуля таблиці;
- **right** — праворуч від поля таблиці.

Для заголовка, що розташовується в правій або лівій частині поля таблиці, значення властивості **width**, відмінне від **auto**, явно встановлює ширину, у той час як значення **auto** пропонує браузеру вибрати ширину самому. Те ж стосується властивості **hight** для заголовка, розташованого вище або нижче поля таблиці.

Для вирівнювання вмісту заголовка по горизонталі усередині поля заголовка використовується властивість **text-align**. Для вертикального вирівнювання заголовка використовується властивість **vertical-align**(значення **top, middle** і **bottom**).

Вертикальне вирівнювання

Властивість **vertical-align** кожної комірки таблиці визначає вертикальне вирівнювання її вмісту. Вміст кожної комірки має базову лінію, верх, середину й низ. У контексті таблиць властивість **vertical-align** може приймати такі значення.

- **baseline** — базова лінія комірки розташовується на тій же висоті, що і базова лінія першого рядка, зайнятого коміркою.
- **top** — верх блоку комірки збігається з верхи першого рядка, зайнятого коміркою.
- **bottom** — низ блоку комірки збігається з низом останнього рядка, зайнятого коміркою.
- **middle** — середина комірки збігається із серединою рядків, зайнятих коміркою.

Горизонтальне вирівнювання

Горизонтальне вирівнювання вмісту комірки встановлюється властивістю **text-align**. Якщо в якості значення властивості text-align для кількох комірок у колонці заданий рядок тексту, то вміст таких комірок вирівнюється щодо вертикальної осі. Початок рядка стикається із цією віссю. Розташування рядка ліворуч або праворуч щодо осі визначається напрямком тексту.

Це стане зрозуміло із прикладу. Створимо наступний код:

td { text-align: ”.”}

Тоді цифри, розташовані в таблиці, будуть вирівнюватися щодо десяткової крапки.

Динамічні ефекти рядків і колонок

Властивість **visibility** для елементів рядків, груп рядків, колонок і груп колонок може приймати значення **collapse**. В результаті весь рядок або вся колонка не буде відображатися і місце, яке б вони займали, буде доступно для розміщення вмісту. Це дозволяє динамічно видаляти рядки або колонки, не змінюючи розташування таблиці.

Дану властивість зручно використовувати, якщо доводиться додавати на сторінку якийсь блок, який повинен, по-перше, обтікатися текстом, а по-друге, бути виділеним фоном і границею.

Границі

Для визначення границь можна використовувати наступні властивості: **border,border-collapse** і **border-spacing.**

Властивість **border** визначає стиль границі й розмір. Розмір границі визначається цілим числом та зі зазначенням одиниці виміру.

Властивість **border-collapse** дозволяє вибрати модель границь таблиці. Значення **separate** визначає модель із окремими границями. Значення **collapse** визначає модель з границями, що перетинаються.

Властивість **border-spacing** може приймати відразу два значення довжини. Задана довжина показує відстань між границями сусідніх комірок. Якщо вказана одна довжина, то вона визначає відстань по горизонталі та по вертикалі. Якщо вказано обидва значення, то перше визначає відстань по горизонталі, а друге — по вертикалі. Звісно, довжини не можуть бути від'ємними.

Розглянемо простий приклад визначення границі для таблиць на сторінці:

```
table { border: outset 10pt;
      border-collapse: separate;
      border-spacing: 15pt }
td { border: inset 5pt }
```

Цей **CSS**-код створить навколо таблиці опуклу границю розміром 10 пунктів. Оскільки задане значення **separate** властивості **border-collapse**, то для комірок будуть створені окремі границі розміром 5 пунктів. Відстань між комірками буде становити 15 пунктів.

Стилі границь

Крім розміру, **CSS** надає можливість задавати стиль границь таблиці. Для такого використовується властивість **border-style**. Розглянемо список значень даного властивості.

- **none** — границя відсутня.
- **hidden** — аналогічно значенню **none**, але в моделі з границями, що перетинаються, має пріоритет над будь-якими іншими границями.
- **dotted** — границя подається рядом крапок.
- **dashed** — границя подається рядом коротких лінійних сегментів.
- **solid** — границя подається єдиним сегментом лінії.
- **double** — границя подається двома суцільними лініями. Сума товщин двох ліній і відстані між ними дорівнює значенню **border-width**.
- **groove** — границя має вигляд втисненої.
- **ridge** — на відміну від **groove**, границя має вигляд опуклої.
- **inset** — у моделі з окремими границями весь блок має вигляд втисненого. У моделі з границями, що перетинаються, це значення дає той же ефект, що і значення **groove**.
- **outset** — у моделі з окремими границями весь блок виглядає опуклим. У моделі з пересічними границями це значення дає той же ефект, що й значення **ridge**.

Порожні комірки

CSS дозволяє визначити, чи відображати границі порожніх комірок. Можна встановити відображення порожніх комірок, використовуючи властивість **CSSempty-cells**. Ця властивість керує відображенням границь навколо комірок, що не мають видимого вмісту. Уважається, що видимого вмісту немає в порожніх комірках і комірках, для яких властивості **visibility** присвоєно значення **hidden**.

Щоб усі комірки таблиці (в тому числі й порожні) мали границі, можна використовувати такий **CSS**-код:

table { empty-cells: show }

Псевдостилі гиперпосилань

До гиперпосилань можна застосовувати стилі. Це свого роду визначені стилі, застосовувані до гіперпосилань в особливих випадках і називані *псевдостилями*:

- **link** — Застосовується до всіх гіперпосилань документа, які користувач ще не відвідав. Аналогічний атрибуту **LINK** тегу **<BODY>**;
- **active** — Застосовується до всіх активних гіперпосилань документа. Аналогічний атрибуту **ALINK** тегу **<BODY>**;
- **visited** — Застосовується до всіх гіперпосилань документа., які користувач відвідав. Аналогічний атрибуту **VLINK** тегу**<BODY>**;
- **hover** — Застосовується до гіперпосилання, на яке вказує курсор миші. Підтримується тільки **Internet Explorer**.

Можна перевизначити їх, як звичайні стилі, і в такий спосіб змінити зовнішній вигляд гіперпосилань сайту.

A: active {color: yellow}
A: visited {text-decoration: line-through}
A: hover {color: yellow; text-decoration: none}

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке позиціонування елементу і за допомогою яких атрибутів його можна здійснити?
2. Що таке генерований вміст і як його створити?
3. Що таке лічильники?
4. Як створити автоматичну нумерацію?
5. Який атрибут визначає вид маркера списку?
6. Як встановити маркер списку у вигляді графічного зображення?
7. Як встановити більш компактне відображення списку?
8. Які властивості можуть бути задані колонкам?
9. Як встановити ширину таблиці?

10.Як встановити висоту таблиці?
11.Як встановити положення та вирівнювання заголовка?
12.Як встановити горизонтальне вирівнювання вмісту комірки?
13.Як встановити вертикальне вирівнювання вмісту комірки?
14.Які атрибути використовуються для відображення границь?
15.Як встановити стилі границь?
16.Як відобразити границі порожніх комірок?
17.Що таке псевдостилі гіперпосилань?

ЗМІСТ

МОВА РОЗМІТКИ ГІПЕРТЕКСТІВ (HTML) 1
Лекція 1. Вступ. Структура Web-документа. Робота з текстом в HTML.Типи та описання списків засобами HTML. 1
Вступ до мови розмітки гіпертекстів (html) 1
World Wide Web та її призначення 1
Формат документів HTML. Web-сторінки та Web-сайти 2
Гіпертекстові посилання 2
Адреса Web-документу 2
Браузери та їх можливості 3
Теги та їх атрибути. Синтаксис опису тега 3
Структура Web-сторінки 5
Обов’язкові теги 5
Теги роботи з текстовими блоками. 6
Фізичне форматування тексту 7
Логічне форматування тексту 8
&- послідовності 9
ТИПИ ТА ОПИСАННЯ СПИСКІВ ЗАСОБАМИ HTML 9
Списки та їх типи 9
Організація маркованих списків 10
Організація нумерованих списків 10
Організація вкладених списків 11
Організація списків визначень. 11
Контрольні питання 12
МОВА ОПИСУ ГІПЕРТЕКСТІВ (HTML) 14
Лекція 2. Таблиці, їх організація, опис та форматування засобами HTML. Вставка та форматування малюнків наWeb-сторінці. Поняття гіперпосилань (адреси), їх опис та створення 14
Організація, опис та форматування таблиць засобами HTML 15
Таблиці у Web-документах 15
Створення структури таблиці 15
Створення рядків та колонок таблиці 17
Вставка та форматування малюнків на web-сторінці 18
Графічні файли у Web-документах 18
Вставка фонового зображення у Web-сторінку 20
Включення графічних зображень в HTML – документ. 20
Використання графіки для створення маркерних списків 21
Поняття гіперпосилань (адреси), їх опис та створення 22
Гіпертекстові посилання, їх призначення та структура. 22
Створення гіперпосилань у Web-сторінці 23
Створення внутрішніх посилань на Web-сторінці 24
Контрольні запитання 25

КАСКАДНІ ТАБЛИЦІ СТИЛІВ (CSS) 27
Лекція 3. Вступ. Визначення CSS-стилю. Форматування тексту засобами CSS. 27
Каскадні таблиці стилів, їх можливості і призначення 27
Додавання CSS-стилів в документ 28
Внутрішні стилі 28
Глобальні стилі (стилі рівня документу) 29
Визначення стилів через класи та ідентифікатори 30
Імпортування зовнішніх таблиць стилів 31
Зв’язані стилі 32
Пріоритет стилів 32
Форматування тексту засобами CSS 34
Колірне оформлення 34
Робота з шрифтами 35
Робота з текстом 36
Робота з блоками тексту 38
Визначення полів документа 40
Створення відступів та інтервалів у заголовків 41
Абзацний відступ 41
Контрольні питання 41
КАСКАДНІ ТАБЛИЦІ СТИЛІВ (CSS) 43
Лекція 4. Оформлення HTML-документів засобами CSS 43
Оформлення HTML-документів засобами CSS 43
Позиціонування елемента 43
Генерований вміст 46
Автоматична нумерація і лічильники 47
Форматування списків 48
Форматування таблиць 49
Колонки *49*
Ширина таблиці *50*
Висота таблиці *50*
Положення та вирівнювання заголовка *50*
Вертикальне вирівнювання *51*
Горизонтальне вирівнювання *51*
Динамічні ефекти рядків і колонок *51*
Границі *51*
Стилі границь *52*
Порожні коміркі *53*
Псевдостилі гиперпосилань 53
Контрольні питання 53

Для довідок