**Розробка уроку**

**Тема:** каскадні таблиці стилів  
  
**Мета:**

* ознайомити:
  + з мовою розмітки даних HTML;
  + з правилами форматування даних;
  + з властивостями стилів та їх значеннями;
  + з тегами й селекторами каскадних таблиць стилів;
* навчити:
  + правилам форматування даних;
  + оперувати властивостями стилів та їхніми значеннями;
  + форматувати документи HTML.

Після вивчення теми учень

* має уявлення про:
  + правила форматування даних;
  + властивості стилів та їхні значення;
* уміє:
  + розрізняти властивості стилів;
  + форматувати документи HTML.

**Обладнання:** комп’ютери зі встановленими ОС і браузером.  
  
**Структура уроку**

1. Організаційний момент.
2. Актуалізація опорних знань.
3. Вивчення нового матеріалу.
4. Інструктаж з ТБ.
5. Закріплення вивченого матеріалу.
6. Підведення підсумків уроку.
7. Домашнє завдання.

**Хід уроку**  
  
**1. Організаційний момент**  
Вітання з класом. Перевірка присутності і готовності учнів до уроку. Перевірка виконання домашнього завдання.  
  
**2. Актуалізація опорних знань**

1. Яка структура документу HTML?
2. Що таке гіперпосилання?
3. Назвіть теги форматування тексту.
4. Які є види списків?
5. Назвіть теги, які використовують для створення таблиць.
6. Назвіть атрибути, які використовують для тегу <div>.
7. Назвіть атрибути, які використовують для тегу <img>.

**3. Вивчення нового матеріалу**  
  
**Каскадні таблиці стилів** (англійською Cascading Style Sheets, CSS) — *спеціальна мова, яку використовують для запису оформлення сторінок, написаних мовами розмітки даних.*  
  
Таблиці стилів дають змогу спростити процес створення сторінок і поліпшити їх зовнішній вигляд. Концепція стилів подібна до ідеї стилів, яку реалізовано в сучасних текстових редакторах — текст спочатку вводять, а потім форматують, користуючись стилями. Застосування стилів дає змогу вводити на сторінку потрібні тексти та інші елементи, не задумуючись над їхнім зовнішнім виглядом і розташуванням.  
  
Таблиці стилів зазвичай створють окремо від html-файлу. Під час створення html-файлу увагу концентрують на змісті сторінки, а не на її зовнішньому вигляді, а під час створення таблиці стилів — навпаки. Отже, стилі дають змогу розділити етапи створення html-файлу й удосконалення зовнішнього вигляду сторінки.  
  
**Cпособи взаємодії таблиці стилів з html-файлом:**

* *зв'язування;*
* *імпортування;*
* *вбудовування (використання стильових класів).*

**Зв'язування**   
Таблицю стилів створюють і зберігають в окремому файлі з розширенням css. Таку таблицю називають *зовнішньою.* Щоб зв'язати основний файл з такою таблицею стилів, у середині тега <head> застосовують одинарний тег <link> з інформацією про таблицю.

<head>

<link href="*адреса таблиці стилів*"

type="text/css"

rel="stylesheet">

</head>

Тут  
href="*адреса таблиці стилів*" — адреса розташування файлу,  
type="text/css" — опис типу файлу,  
rel="stylesheet" — вказівка на відношення основного файлу з файлом таблиці стилю.  
  
**Імпорт**  
Це те саме, що і зв'язування, але взаємодію файлів забезпечують засобами тегу <style> і такої вказівки:  
  
@import URL(*адреса таблиці стилів*)  
  
**Вбудовування (використання стильових класів)**  
Якщо таблицю створено лише для деякого конкретного html-файлу, то її розташовують у цьому файлі в контейнері <style>. Таку таблицю стилів називають *внутрішньою* або *вбудованою.* Також стиль можна вбудувати безпосередньо в html-тег за допомогою атрибуту <style>. Наприклад, контейнер  
  
<span style="color: blue">Blue </span>  
  
створює блок з текстом синього кольору: Blue.  
  
Таблиця стилів складаєтся з правил, а правило — з назви тега (*селектора*) чи списку назв тегів і описів стилів, які діятимуть у межах деякого html-файлу.  
  
**Опис стилю** — *це послідовність пар властивість: значення, які записують через крапку з комою у фігурних дужках.*  
  
**Загальний вигляд правила** такий:

*Селектор*{*властивість*1: *значення*1;

*властивість*2: *значення*2; …

*властивістьN*: *значенняN*;}

Наприклад, для виведення усіх абзаців червоним кольором застосовують таке правило:  
  
p { color: red;}  
  
Проаналізуємо приклад складнішого правила для списку тегів.

h3, li{ color: green;

font-family: pragmatica;

font-size: 16pt;

text-align: left;

border-style: ridge;

border-width: 2mm;}

Браузер виведе всі заголовки третього рівня h3 і елементи списків <li> зеленим кольором, гарнітурою pragmatica розміру 16 пунктів, з вирівнюванням за лівим краєм вікна, рамкою товщиною 2 мм з видавленим контуром (ridge).  
  
Назва властивості складається з одного чи декількох англійських слів, які записують через риску. Властивість діє лише у межах тега, зазначеного у відповідному правилі.  
  
Властивості стилів подамо за такою схемою:  
  
**назва властивості**  
значення (перелік або приклади)  
*— пояснення.*

**Властивості стилів, їхні значення та призначення**

**background-attachment**  
fixed, scroll  
*— тло стале чи прокручуєтся;*  
  
**background-color**  
red, green, #343434  
*— колір тла;*  
  
**background-image**  
*— url-адреса графічного файлу для тла;*  
  
**background-position**  
0% 0%  
*— початкове розташування тла;*  
  
**background-repeat**  
repeat,  
repeat-x,  
repeat-y,  
no-repeat  
*— повторювання зображення тла;*  
  
**border-top,  
border-right,  
border-bottom,  
border-left,  
border**  
4mm  
*— товщина лінії рамки;*  
  
**border-color**  
red, green, #343434  
*— колір рамки;*  
  
**border-style**  
none, dotted, dashed, solid, double, groove, ridge, inset, outset  
*— стиль рамки*  
  
**font-family**  
arial, sans-serif  
*— гарнітура;*  
  
**font-size**  
24px  
*— розмір шрифту;*  
  
**font-style**  
normal, italic, oblique  
*— стиль шрифту;*  
  
**font-variant**  
normal, small-caps  
*— варіант шрифту;*  
  
**font-weight**  
normal (400), bold (700), bolder, lighter, 100-900  
*— жирність шрифту;*  
  
**word-spacing**  
1mm, 2mm  
*— відстань між словами;*  
  
**letter-spacing**  
1mm  
*— відстань між символами;*  
  
**line-height**  
2mm, 4mm  
*— відстань між рядками;*  
  
**text-align**  
left, right, center, justify  
*— вирівнювання тексту;*  
  
**text-decoration**  
none, underline, blink, overline, line-through  
*— оформлення тексту;*  
  
**text-indent**  
2cm  
*— абзацний відступ;*  
  
**text-transform**  
none, uppercase, lowercase, capitalize  
*— трансформація тексту;*  
  
**vertical-align**  
baseline, sub, super, top, text-top, middle, bottom, text-bottom  
*— вертикальне вирівнювання;*  
  
**white-space**  
normal, pre, nowrap, pre-wrap, pre-line  
*— пробіли та перенесення між словами;*  
  
**margin-top  
margin-right  
margin-bottom  
margin-left  
margin**  
4mm  
*— зовнішній відступ;*  
  
**padding-top  
padding-right  
padding-bottom  
padding-left  
padding**  
4mm  
*— внутрішній відступ;*  
  
**height**  
4cm  
*— висота елемента;*  
  
**width**  
14cm  
*— ширина елемента;*  
  
**float**  
none, left, right  
*— обтікання об'єкта текстом;*  
  
**clear**  
none, left, right, both  
*— заборона обтікання;*  
  
**color**  
red, green, #343434  
*— колір елемента;*  
  
**list-style-image**  
url(*aдреса графічного файлу маркера списку*)  
*— зображення для маркера;*  
  
**list-style-position**  
inside, outside  
*— позиція маркера;*  
  
**list-style-type**  
disc, circle, square, decimal, lower-alpha, lower-roman, upper-alpha, upper-roman  
*— вигляд маркера;*  
  
**position**  
static, relative, absolute, fixed  
*— спосіб позиціонування елемента;*  
  
**top  
right  
bottom  
left**  
px, in, pt, %, auto тощо  
*— позиціонування елемента;*  
  
**z-index**  
ціле число, auto  
*— положення елемента за віссю z;*  
  
**display**  
block, inline-block, inline, list-item, none, table, table-row, table-cell  
*— відображення елемента в документі;*  
  
**visibility**  
visible, hidden, collapse  
*— відобразити чи сховати елемент;*  
  
**overflow**  
visible, hidden, scroll, auto  
*— спосіб відображення вмісту елемента, якщо він повністю не вміщується в задану область;*  
  
**clip**  
shape(rect), auto  
*— визначає прямокутну область, де буде показано частину елемента*  
  
**outline-color  
outline-style  
outline-width  
outline**  
4mm  
*— зовнішня рамка*  
  
**cursor**  
default, pointer, help, wait, progress, text, move, crosshair, auto  
*— вигляд вказівника*.  
  
Правила можуть починатися або з назви тега без кутових дужок, або з деякого слова користувача, перед яким є крапка. Це слово стає назвою типу, який можна застосувати в контейнері body до іншого текстового блока, фрагменту тексту чи деякого елемента. Наприклад,

.mystyle{color: red;

font-size: 40pt;

margin-top: 30px;}

**Внутрішні таблиці стилів** описують у головному файлі в контейнері head за допомогою контейнера style.  
  
**Виокремлення текстового блоку**, до якого буде застосовано стиль, здійснюють за допомогою контейнера  
  
<div class=*назва стилю*>…</div>  
  
де значення параметра class задає конкретний стиль.  
  
**Виокремлення у блоці фрагмента**, для якого буде переозначено стиль, здійснюють за допомогою контейнера  
  
<span>…</span>  
  
**Селектор class**  
Нехай поставлено завдання зробити сторінку, на якій буде два різновиди абзаців, причому обидва будуть постійно чергуватися і повторюватися. Наприклад це текст, в якому чергуются питання вчителя і відповіді учня. При створені такої сторінки бажано візуально відділити питання і відповіді одне від одного. Можна створити створити дві різні таблиці стилів. На щастя, це робити не обов'язково: можна створити в одній таблиці стилів два різних класи абзаців за допомогою селектора класу. Проаналізуйте відповідний [прикладі](http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/139/1.html) та проаналізуйте опис властивостей.  
  
Якщо немає гарантії наявності певного шрифту, бажано через кому вказати інші прийнятні значення font-family. Буде вибраний перший із наявних у переліку.  
  
**Селектор id**  
Розглянемо випадок, коли потрібно створити на сторінці унікальні елементи, до яких у майбутньому будуть звертатися за допомогою мови JavaScript. Можливо, ці елементи будуть повторюватися на інших сторінках, і бажано задати їм єдине оформлення за допомогою CSS. На цей випадок у таблицях стилів є можливість надання унікальним елементам ідентифікаторів (id) — див. [приклад](http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/139/2.html) призначення ідентифікатора і правил CSS.  
  
Розглянемо найпростіший випадок використання CSS щодо відображення тексту — вкажемо правила для заголовків.

h2 {font-family: arial, verdana, ubuntu;

font-weight: bold;

font-style: italic;}

h1 {font-variant: small-caps;}

Для контейнерів h1 і h2 встановлено пріоритетний шрифт, а потім перераховано альтернативні і схожі за властивостями. Заголовок буде відображено першим наявним у списку шрифтом.  
  
**Родина шрифту [font-family]** (гарнітура)  
Властивість font-family вказує пріоритетний список шрифтів, що використовують для відображення даного елемента. Якщо перший шрифт списку не встановлено на комп'ютері, з якого виконують доступ до сайту, шукають наступний шрифт списку, поки не буде знайдено відповідний. У поданому вище прикладі заголовки h2 буде відображено гарнітурою Arial. Якщо її не встановлено, буде використано гарнітуру Verdana. Якщо недоступно обидва шрифти, для показу заголовків буде використано ubuntu. Якщо назву гарнітури задано окремими словами, то її потрібно обмежити подвійними лапками. Наприклад, "Times New Roman".  
  
**Стиль шрифту [font-style]** (написання) може мати такі значення: normal, italic (курсив) або oblique (похилий). У поданому прикладі заголовки h2 буде показано курсивом.  
  
**Варіант шрифту [font-variant]** може мати такі значення:

* normal — звичайні малі літери;
* small-caps — малі заголовні літери замість звичайних малих літер.

Якщо font-variant має значення small-caps, а відповідний шрифт недоступний, то браузер, швидше за все, відобразить текст літерами верхнього регістру.  
  
**Жирність шрифту [font-weight]** описує, наскільки жирним (важким) має бути шрифт: normal або bold. Деякі браузери підтримують навіть числові значення 100..900 (у сотнях) для опису жирності шрифту.  
  
**Розмір шрифту [font-size]** можна встановити, використавши одну з таких одиниць вимірювання:

* px — фіксована величина піксель (pixels), екранна точка;
* pt — фіксована величина (points), 1/72 дюйма;
* % — за 100% беруть поточний розмір шрифта;
* em — поточний розмір шрифта.

px і pt дають абсолютне значення розміру шрифту, а % і em — відносні. Багато користувачів не можуть читати дрібний текст. Щоб зробити документ доступним для всіх, бажано використовувати саме ті, що задають відносні розміри — % або em.  
  
**Скорочений запис [font]** використовують, щоб вказати всі властивості шрифту в одному стильовому правилі. Наприклад, чотири рядки опису властивостей шрифту для абзацу:

p{

font-style: italic;

font-weight: bold;

font-size: 30px;

font-family: arial, sans-serif;

}

можна замінити таким записом:  
  
p{font: italic bold 30px arial, sans-serif;}  
  
Порядок властивостей font такий:  
font-style | font-variant | font-weight | font-size | font-family.  
  
**Відступ першого рядка** задають значенням властивості text-indent (про одиниці вимірювання див. вище).  
  
**Вирівнювання тексту** задають значенням властивості text-align:

* left — вирівняти за лівим краєм;
* right — вирівняти за правим краєм;
* center — центрувати;
* justify — вирівняти по ширині.

**Декоративні ефекти** задають значенням властивості text-decoration:

* underline — підкреслити текст;
* overline — провести лінію під текстом;
* line-through — перекреслити текст.

**Інтервал між літерами** задають як значення властивості letter-spacing в одиницях вимірювання довжини. Наприклад, відстань 3px між літерами у абзацах задають таким кодом:  
  
p{letter-spacing: 3px;}   
  
**Трансформація тексту** буде здійснена, якщо надати певних значень властивості text-transform:

* capitalize — кожне слово з великої літери — john doe буде подано як John Doe;
* uppercase — конвертує всі символи у верхній регістр — john doe буде подано як JOHN DOE;
* lowercase — конвертує всі символи в нижній регістр — JOHN DOE буде подано як john doe;
* none — трансформації немає — текст буде відображено так само, як у коді html.

Для гіперпосилань можна використовувати зміну шрифту, кольору, підкреслення. Новим є те, що в CSS ці властивості можна визначати по-різному в залежності від того, відвідали вже посилання, чи воно активне, чи знаходиться вказівник над посиланням.  
  
**Клас** — *одна із форм селекторів html, який можна використати для визначення декількох елементів на відміну від*ідентифікатора (id), *які можуть визначати один элемент. Клас описують таким чином:*.*назва класу* {*опис властивостей*}  
  
**Псевдоклас** — *це властивість, яка дозволяє змінювати стиль елемента у залежності від дій користувача, а також розташування елемента (тега) у загальному потоці документа, що додає у дизайн сторінки деяку динаміку і логіку.*  
  
Класичним прикладом застосування псевдокласу є посилання, що змінює свій колір при наведенні на нього вказівника. Посилання в коді html задають контейнером a. У CSS можна використати цей тег як селектор:  
  
a{color: blue;}  
  
Псевдокласи можна використати для встановлення різних стилів для відвіданих і невідвіданих посилань відповідно. Перелічимо ці *псевдокласи* посилання:

* a:visited — відвідані;
* a:link — невідвідані;
* a:active — активні;
* a:hover — під вказівником.

Наприклад, якщо ми хочемо, щоб при наведенні на них вказівника посилання ставали помаранчевими, їх було подано курсивом без підкреслення, то CSS потрібно записати таким чином:  
  
a:hover {color: orange; font-style: italic; text-decoration: none;}  
  
**Групування за допомогою контейнера span** у поєднанні з CSS використовують для створення візуальних ефектів *в окремих блоках* тексту. Наприклад, у реченнях: «Стиглі сливи високо на дереві. Зірви, з'їж, природі дякую скажи» потрібно виділити дієслова. Для цього можна використати контейнер span:  
  
<p>Стиглі сливи високо на дереві. <span class="benefit">Зірви</span>, <span class="benefit">з'їж</span>, природі дякую <span class="benefit">скажи</span>.</p>  
  
У кожному блоці span властивості class буде надано значення benefit, яке потрібно визначити у таблиці стилів. Наприклад, для виділення червоним кольором це можна зробити таким чином:  
  
span.benefit{color:red;}  
  
**Групування за допомогою контейнера div** застосовують для групування *одного або більше блоків* — див. [приклад](http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/139/3.html).  
  
**Правило @media** дозволяє вказати для певного типу носія свій набір стилів. Перелічимо ці типи:

* all — усі типи;
* aural — мовні синтезатори і програми для відтворення тексту;
* braille — пристрої, створенні по системі Браєля;
* handheld — планшети;
* print — принтер;
* projection — проектор;
* screen — екран монітора;
* tty — телетайпи, термінали;
* tv — телевізор.

Описують правило таким чином:  
  
@media *тип1 [, тип2]* { *Опис стилю* }  
  
Після ключового слова @media пишуть один або декілька типів носіїв. Якщо їх більше одного, писати потрібно через кому. А у фігурних дужках — опис стилю.  
  
**4. Закріплення вивченого матеріалу**

1. Назвіть три способи взаємодії СSSтаблиці з html-файлом
2. Що таке псевдоклас?
3. Які властивості форматування та встановлення стилю тексту ви знаєте?
4. Яка послідовність дій при встановлені шрифту?
5. Що таке class?
6. Назвіть відомі вам селектори CSS

**5. Підбиття підсумків уроку**  
Виставлення оцінок.  
  
**6. Домашнє завдання**  
Повторити вивчений матеріал щодо створення й редагування каскадних таблиць стилів. Потренуватися у редагуванні й форматуванні html використовуючи CSS

Текст упорядкував *Круць Олег Петрович*, учитель інформатики СШ № 280 Дарницького району міста Києва під час виконання випускної роботи на курсах підвищення кваліфікації з 16 березня по 3 квітня 2015 року.