МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни «Основи веб-розробки»
для студентів спеціальностей
121 «Інженерія програмного забезпечення» та
122 «Комп'ютерні науки»

Затверджено
редакційно-видавничою
радою університету,
протокол № 1 від 13.02.2025 р.

Харків
НТУ «ХПІ»
2025

Методичні вказівки до лабораторних робіт з дисципліни «Основи веб-розробки» для студентів спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки»// уклад. Літвінова Ю.С.– Харків: НТУ «ХПІ» – 2025. –61 с.

Укладач:
Ю.С. Літвінова

Рецензент:
В.В. Москаленко

Кафедра програмної інженерії та інтелектуальних технологій

ЗМІСТ

Лабораторна робота 1

Тема: Розроблення веб-сторінок із використанням мови HTML

Мета роботи: Під час виконання цієї лабораторної роботи необхідно освоїти базові прийоми використання мови HTML для створення веб-сторінки.

Завдання до роботи

1. Спроектувати структуру веб-сайту за обраною предметною галуз'ю. (обсягом 4-5 сторінок). Передбачити розміщення навігаційних панелей і дублювання елементів навігації. У нижній частині кожної сторінки має міститися інформація про авторів та авторські права.
2. Розробити ескіз оформлення веб-сайту (використовувати будь-який графічний редактор).
3. Виконати верстку макету сторінки з блоковою структурою за розробленим ескізом.
4. Сайт має містити такі структурні елементи HTML-коду:
   - списки,
   - таблиці,
   - картинки,
   - гіперпосилання на сторінки та гіперпосилання на позицію всередині сторінки;
   - змінені колір / розмір / шрифт / колір фону тексту.

У якості веб-технологій у процесі розробки сайта для цієї роботи використовувати тільки HTML!

Вказівки до роботи

Основні поняття

Структура веб-сторінки.

Всі веб-сторінки складаються з двох розділів – заголовка (<HEAD>) і тіла документа (<BODY>). Розділ заголовка може містити текст і теги, але вміст цього розділу не показується безпосередньо на сторінці.

Приклад 1.1 Найпростіший HTML-документ

```
<!DOCTYPE HTML >
<html>
<head>
<!-- Цей розділ призначений для заголовка сторінки та технічної інформації. -->
</head>

<body>
<!—А тут треба розміщувати все, що бажано побачити на сторінці. -->
</body>
</html>
```

Основні елементи веб сторінки.

DOCTYPE

Елемент <! DOCTYPE> призначений для вказівки на тип поточного документа – DTD (document type definition, опис типу документа). Це необхідно, щоб браузер розумів, як слід інтерпретувати поточну веб- сторінку, адже HTML існує в декількох версіях, крім того є XHTML (EXtensible HyperText Markup Language – розширена мова розмітки гіпертексту), схожий на HTML, але відрізняється від нього за синтаксисом; щоб браузер "не плутався" і розумів, згідно з яким стандартом відображати веб-сторінку та як необхідно в першому рядку коду задавати <! DOCTYPE>.

Розділ заголовка документа (<HEAD>)

Теги та тексти, що знаходяться в цьому розділі, не відображуються на веб-сторінці. Цей розділ, як правило, призначений для наступної службової інформації.

Заголовок сторінки (тег <TITLE>)

Використовується для відображення рядка тексту в назві закладки вікна браузера. Такий рядок повідомляє користувачеві назву сайта та іншу інформацію, яку додає розробник.

CSS (Cascading Style Sheets, каскадні таблиці стилів)

Стилі зберігають набір елементів форматування, який застосовується до тексту документа, щоб швидко змінити його зовнішній вигляд.

Метатеги (тег <META>)

Метатеги використовуються для зберігання інформації, призначеної для браузерів і пошукових систем. Наприклад, механізми пошукових систем звертаються до метатегів для отримання опису сайта, ключових слів та інших даних. Хоча тег <META> всього один, він має безліч параметрів. Для відвідувача веб-сторінки інформація, яку несуть мета-теги, буде не видно.

У документі може бути будь-яка кількість тегів <meta>. Усі вони розміщуються у блоці <head>...</head>.

Розглянемо деякі, часто використовувані мета-теги:

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

Використовується для того, щоб браузер міг правильно визначити тип і вміст і кодування веб-сторінки.

<meta http-equiv="Refresh" content="N; url=http://example.org/">

Автоматичне перенаправлення (редирект) через N секунд після відкриття поточної сторінки на вказану адресу .

<meta name="author" content="Ім'я автора сторінки">

Використовується для вказівки імені автора. Пошукові системи можуть знайти потрібну інформацію на ім'я автора.

<meta name="keywords" content="список, ключових, слів">

У мета-тезі keywords вказуються ключові слова присутні у документі. Цей тег спочатку був орієнтований на пошукові машини, але був скомпрометований веб-майстрами, які використовували його для пошукового спаму.

<meta name="description" content="Сюди вписується короткий опис сторінки">

Цей тег визначає фразу, за якою користувач визначає суть вашої сторінки і вирішує, чи відвідувати її. Вписані вирази в цей meta-тег відіграють важливу роль у рейтингу сторінки. Ключові фрази з опису повинні співпадати з основним текстом сторінки, це також відіграє велику роль при індексації сторінки пошуковими роботами.

<meta name="robots" content="index,all">

Управління пошуковим роботом, вказівку йому того, що сторінку потрібно індексувати (або ні, якщо вказано "noindex").

Скрипти

Скриптом традиційно називають програму, яка впроваджується в тіло веб-сторінки та виконує на ній певні дії. Поширеною мовою програмування для написання скриптів є JavaScript.

Порядок тегів в заголовку документа принципового значення не має.

Тіло документа (<BODY>)

Тіло документа призначене для відображення даних на веб-сторінці, зокрема, в тілі розміщується текст, зображення, посилання, таблиці, списки тощо.

Коментарі

Певний текст можна приховати від показу в браузері, зробивши його коментарем. Хоча такий текст користувач не побачить, він буде передаватися в документі, тому, подивившись вихідний код, можна виявити приховану інформацію. Коментарі починаються тегом <!-- і закінчуються тегом -->. Усе, що знаходиться між цими тегами, відображатися на веб-сторінці не буде.

Робота з текстом

Форматування тексту

Форматування тексту – це засоби його зміни – такі, як вибір накреслення шрифту та використання ефектів, що дозволяють змінювати вид тексту. У таблиці 1.1 перелічені основні теги, які застосовуються для зміни оформлення тексту.

Таблиця 1. 1 - Теги для форматування тексту

Код HTML	Опис	Приклад
Текст	Жирне накреслення тексту	Текст
Текст	Курсив тексту	Текст
^{Текст}	Верхній індекс	$e=mc^2$
_{Текст}	Нижній індекс	H_2O
<pre>Текст</pre>	Текст пишеться як є, включаючи всі прогалини	Текст
Текст	Акцентування тексту	Текст
Текст	Важливий текст	Текст

Слід зазначити, що теги <B> і <STRONG>, також як <I> і <EM>, є не зовсім еквівалентними та замінними. Перший тег <B> є тегом фізичної розмітки та встановлює жирний текст, а тег <STRONG> – тегом логічної розмітки та визначає важливість поміченого тексту. Такий поділ тегів на логічне та фізичне форматування спочатку призначався для того, щоб

зробити HTML універсальним, в тому числі не залежним від пристрою виведення інформації.

Будь-які теги форматування тексту можна використовувати спільно.

Розмір тексту

Для зміни розміру тексту існує кілька можливостей – це використання заголовків <H1>,...,<H6>, тегів <BIG> і <SMALL>. У табл. 1.2 перелічені основні варіанти з описом і прикладом.

Теги <BIG> і <SMALL> можна повторювати кілька разів поспіль, тим самим збільшуючи або зменшуючи текст до потрібних розмірів.

Серед поданих у табл. 1.1 тегів переважно застосовуються теги <H1>, <H2> і <H3>. Вони призначені для створення заголовків до розділів і показують їх відносну важливість. Так, за замовчуванням, текст всередині тегу <H1> відображується в жирному накресленні та розміром 24-х пунктів. Вміст тегу <H2> вже має розмір 18 пунктів, а <H3> – 14 пунктів.

Таблиця 1.2 - Теги для зміни розміру тексту

Код HTML	Опис	Приклад
<big>Текст</big>	Збільшує розмір шрифту	Текст
Текст	Зменшує розмір шрифту	Текст
Текст	Пише текст у вигляді великого заголовка	Текст
Текст	Пише текст у вигляді маленького заголовка	Текст

Вирівнювання тексту

Вирівнювання тексту визначає його зовнішній вигляд, орієнтацію країв абзацу та може виконуватися по лівому або правому краю, по центру або по ширині.

Найбільш поширений варіант – вирівнювання по лівому краю, коли зліва текст зсувається до краю, а правий залишається нерівним. Вирівнювання по правому краю та по центру в основному використовується в заголовках і короткому змісті. Слід мати на увазі, що при використанні вирівнювання по ширині в тексті між словами можуть з'явитися великі інтервали, що не дуже гарно.

Для установки вирівнювання тексту зазвичай використовується тег параграфа <P> з параметром align, який визначає спосіб вирівнювання.

Також блок тексту допустимо вирівнювати за допомогою тегу <DIV> з аналогічним параметром align.,

Вирівнювання елементів по лівому краю задано за замовчуванням, тому вказувати його ще раз необхідності немає. Тоді параметр *align = "left"* можна опустити.

Відмінність між параграфом (тег <P>) і тегом <DIV> в тому, що на початку й у кінці параграфа з'являється вертикальний відступ, чого немає у випадку використання тегу <DIV>.

Параметр *align* достатньо універсальний і може застосовуватися не тільки до основного тексту, а й до заголовків, наприклад <H1>.

Робота із зображеннями

Вставка зображень. Для вбудовування зображення в документ використовується тег <IMG>, що має обов'язковий параметр *src*, який визначає адресу файла з картинкою і *alt*, що визначає альтернативний текст. Загальний синтаксис для додавання зображення наступний:

<img src="URL" alt="альтернативний текст">

Закривати тег не потрібно, URL (*Universal Resource Locator*, універсальний покажчик ресурсів) – це шлях до графічного файла. Для вказівки на нього можна використовувати як абсолютну, так і відносну адресу.

Як правило, в якості формату графічного файла виступає GIF і JPEG.

Особливості GIF

Кількість кольорів у зображенні може бути від 2-х до 256-ти, але це можуть бути будь-які кольори з 24-бітової палітри.

Файл у форматі GIF може містити прозорі ділянки. Якщо використовується відмінний від білого кольору фон, він буде просвічувати крізь "отвори" в зображенні.

Підтримує покадрову зміну зображень, що робить формат популярним для створення банерів і простих анімацій.

Використовує вільний від втрат метод стиснення.

Сфера застосування GIF

Текст, логотипи, ілюстрації з чіткими краями, анімовані малюнки, зображення з прозорими ділянками, банери.

Особливості JPEG

Кількість кольорів у зображенні – близько 16-ти мільйонів, чого цілком достатньо для збереження фотографічної якості зображення.

Основна характеристика формату – якість, що дозволяє управляти кінцевим розміром файла.

Підтримує технологію, так званий прогресивний JPEG, в якому версія малюнка з низькою здатністю з'являється у вікні перегляду до повного завантаження самого зображення.

Сфера застосування JPEG

Використовується переважно для фотографій. Недоцільно використовувати для малюнків, які містять прозорі ділянки, дрібні деталі або текст.

Для зміни розмірів зображення засобами HTML передбачені параметри *width* і *height* тегу <IMG>.

Необхідно обов'язково задавати розміри всіх зображень на веб-сторінці. Це дещо прискорює завантаження сторінки, оскільки браузеру нема потреби обчислювати розмір кожного малюнка після його отримання.

Зображення, яке додається на веб-сторінку, можна помістити в рамку різної ширини. Для цього служить параметр *border* тегу <IMG>. За замовчуванням, рамка навколо зображення не відображується, за винятком випадку, коли малюнок є посиланням. Колір рамки в цьому випадку збігається з кольором тексту, заданому за допомогою стилю або параметра *text* тегу <BODY>

Альтернативний текст дозволяє отримати текстову інформацію про малюнок при відключеному в браузері завантаженні зображень.

Для створення альтернативного тексту використовується параметр *alt* тегу <IMG>.

Для зображень можна вказувати їх розташування щодо тексту або інших зображень на веб-сторінці. Спосіб вирівнювання зображень задається параметром *align* тегу <IMG>. Цей параметр може приймати такі значення: *bottom* (нижня межа зображення вирівнюється за базовою лінією текстового рядка; це значення встановлено за замовчуванням), *left* (зображення розташовується по лівому краю батьківського елементу), *middle* (середина зображення вирівнюється за базовою лінією поточного рядка тексту), *right* (зображення вирівнюється по правому краю батьківського елементу), *top* (верхня межа зображення вирівнюється за найвищим елементом поточного рядка).

Щоб текст не прилягав щільно до малюнка, рекомендується в тезі <IMG> додати параметр *hspace* і *vspace*, які задають відстань до тексту у пікселях.

Посилання

Для створення посилання необхідно повідомити браузеру, що є посиланням, а також вказати адресу документа, на який слід зробити посилання. Обидві дії виконуються за допомогою тегу <A>, який має єдиний обов'язковий параметр *href*. Як значення використовується адреса документа (*URL*).

Адреса посилання може бути абсолютною та відносною. Абсолютні адреси працюють скрізь і всюди незалежно від імені сайту або веб- сторінки, де прописано посилання. Починаються вони зі вказівки протоколу передачі даних. Так, для веб-сторінок це зазвичай *HTTP* (*HyperText Transfer Protocol*, протокол передачі гіпертексту), відповідно, абсолютні посилання починаються з ключового слова *http://*

Робота зі списками

Марковані списки

Для установки маркованого списку використовуються теги <UL> і <LI> .

Марковані списки дозволяють розбити великий текст на окремі блоки. Тим самим залучається увага читача до тексту та підвищується його читабельність. Із урахуванням того, що сприйняття тексту з екрану монітора важче, ніж з його друкованого варіанта, це є досить корисним прийомом.

Маркери можуть приймати один з трьох видів: коло (за замовчуванням), коло та квадрат. Для вибору типу маркера використовується параметр type = "..." тегу <UL>. Замість трьох крапок підставляється одне з трьох значень, зазначених у табл. 1.3.

Таблиця 1.3 - Види маркерів

Код HTML	Приклад

<ul type="disc">	Що слід враховувати в процесі тестування сайта: – працездатність усіх посилань; – підтримку різних браузерів; – читабельність тексту
<ul type="circle">	Що слід враховувати при тестуванні сайта: – працездатність усіх посилань; – підтримку різних браузерів; – читабельність тексту
<ul type="square">	Що слід враховувати при тестуванні сайта: – працездатність всіх посилань; – підтримку різних браузерів; – читабельність тексту

Нумеровані списки

Нумеровані списки є набором елементів з їх порядковими номерами. Вид і тип нумерації залежить від параметрів тегу <OL>, який використовується для створення списку. В якості маркерів можуть бути такі значення: арабські цифри; великі латинські літери; малі латинські літери; великі римські цифри; малі римські цифри.

У табл. 1.4 наведені різноманітні параметри тегу <OL> і результат їх застосування.

До і після списку автоматично додаються вертикальні відступи, це є особливістю тегу <OL>.

Таблиця 1.4 - Варіанти нумерованих списків

Код HTML	Приклад
  - текст - текст - текст 	Нумерований список із параметрами за замовчуванням: 1. текст; 2. текст; 3. текст
<ol start="5">	Нумерований список, який починається з п’яти: 5. текст; 6. текст; 7. текст

<ol type="A">	Нумерований список із великими літерами латинського алфавіту: A. текст; B. текст; C. текст
<ol type="a’>	Нумерований список із малими літерами латинського алфавіту: a. текст; b. текст; c. текст
<ol type="I’>	Нумерований список із великими римськими цифрами: I. текст; II. текст; III. текст
<ol type="1">	Нумерований список із арабськими цифрами: 1. текст; 2. текст; 3. текст
<ol type="I" start="7">	Список с римськими цифрами, починаючи з семи: VII. текст; VIII. текст; IX. текст

Таблиці

Таблиця складається з рядків і стовпців осередків, які можуть містити текст і малюнки. Зазвичай таблиці використовуються для впорядкування та подання даних, однак можливості таблиць цим не обмежуються. За допомогою таблиць зручно верстати макети сторінок, розташувавши потрібним чином фрагменти тексту та зображень.

Для додавання таблиці на веб-сторінку використовується тег-контейнер <TABLE>. Таблиця повинна містити хоча б один рядок і колонку (приклад 1.1).

Для додавання рядків використовується тег <TR>. Щоб розділити рядки на колонки, застосовуються теги <TD> і <TH>.

Приклад 1.2 **Створення найпростішої таблиці**
<!DOCTYPE HTML>

```
<html>
<head>
<meta charset="utf-8">
<title>Таблиця</title>
</head>
<body>
<table>
<tr>
<td> Вміст таблиці </td>
</tr>
</table>
</body>
</html>
```

Відмінність між цими тегами у наступному. Тег <TH> призначений для створення заголовків, вміст такої комірки позначається жирним накресленням і вирівнюється по центру.В іншому випадку діють ці теги однаково.

Особливості таблиць

У кожного параметра таблиці є своє значення, встановлене за замовчуванням. Це означає, що якщо якийсь атрибут пропущений, то неявно він все одно присутній, причому з певним значенням, через що вигляд таблиці може виявитися інакшим, ніж припускав розробник. Щоб розуміти, чого можна чекати від таблиць, слід знати їх явні та неявні особливості, зазначені в подальшому тексті.

Одну таблицю допускається поміщати всередину осередків іншої таблиці. Це потрібно для подання складних даних або в тому випадку, коли одна таблиця виступає в ролі модульної сітки, а друга, всередині неї – вже як звичайна таблиця.

Розміри таблиці спочатку не встановлюються й обчислюються на основі вмісту осередків. Наприклад, загальна ширина визначається автоматично, виходячи з сумарної ширини вмісту осередків плюс ширина кордонів між осередками, поля навколо вмісту, що встановлюється через параметр *cellpadding* і відстань між осередками, які визначаються значенням *cellspacing*.

Таблиця, якщо не вказано особливо, завжди вирівнюється по лівому краю. За замовчуванням, таблиця виводиться без рамки. Якщо ширина таблиці не вказана, вона підганяється під зміст осередків.

Можлива ситуація, коли між осередків таблиці виникають зайві порожні проміжки. Це пов'язано з тим, що перенесення рядків у коді HTML автоматично створює і додатковий пробіл у таблиці. Щоб позбутися цього, треба помістити код всередині тегу <TR> в один рядок.

Для зміни вигляду та властивостей таблиці використовується безліч параметрів, які додаються до тегу <TABLE>. Загальний синтаксис наступний:

<table параметр1="..." параметр2="...">

Параметри, які використовуються тільки для тегів <TH> і <TD>, наведено в табл. 1.5.

Таблиця 1.5 - Властивості осередків таблиці

Властивість	Значення	Опис	Приклад
nowrap		Забороняє переноси рядків у тексті	<td nowrap>
colspan	n	Кількість об'єднуваних колонок	<td colspan="3">
rowspan	n	Кількість об'єднуваних рядків	<td rowspan="3">

Опис параметрів таблиці та їх значень наведено в табл. 1.6.

Таблиця 1.6 - Параметри тегу <TABLE>

Властивість	Значення	Опис	Приклад
align	left right center	Вирівнювання таблиці	<table align="center">
background	URL	Визначає зображення, яке буде використовуватися в якості фонового малюнка таблиці	<table background="pic.gif">
bgcolor	#rrggbb	Колір фону таблиці	<table bgcolor="#ff9900">
border	n	Товщина рамки в пікселях	<table border="2">

cellpadding	n	Відстань між осередком і його вмістом	<table cellpadding="7">
cellspacing	n	Дистанція між осередками	<table cellspacing="3">
cols	n	Задає кількість стовпців у таблиці, допомагаючи браузеру в підготовці до її відображення	<table cols="3">
nowrap		Забороняє переноси рядків у тексті	<table nowrap>
frame	void above below lhs rhs hsides vsides box	Задавання типу рамки таблиці	<table frame="hsides">
rules	all groups cols none rows	Визначає, де малювати між осередками	<table rules="cols">
width	n n%	Мінімальна ширина таблиці, можна задавати в пікселях або відсотках	<table width="90%">

Завдяки великій кількості властивостей осередків, можна змінювати вигляд і оформлення таблиць.

Вміст комірок за замовчуванням вирівнюється по лівому краю по горизонталі і по центру – по вертикалі.

Параметри тегу <TD> мають більший пріоритет, ніж параметри тегу <TR>, а властивості осередків вищі за властивості самої таблиці.

Браузер Internet Explorer може не застосовувати деякі параметри, прикладені до тегу <TR>. У цьому випадку треба використовувати ті ж аргументи, але до тегу <TD> або <TH>.

Спеціальні символи

У таблиці1 .8 наведено деякі спеціальні символи HTML, що мають особливе призначення та власний спосіб подання у вигляді мнемонічного чи числового коду.

Таблиця 1.8 - Спеціальні символи HTML

Символ	Мемомокод	Числовий код	Опис
			нерозривна пробіл
¢	¢	¢	цент
£	£	£	фунт стерлінгів
¥	¥	¥	ієна або юань
§	§	§	параграф
©	©	©	знак copyright
«	«	«	ліва подвійна кутова дужка
	-	-	місце можливого перенесення
®	®	®	знак зареєстрованої торгової марки
°	°	°	градус
2	2	2	верхній індекс два (x^2)
3	3	3	верхній індекс три (x^3)
·	·	·	точка по середині
»	»	»	права подвійна кутова дужка
½	½	½	дріб – один другий
×	×	×	знак множення
÷	÷	÷	знак розподілу
σ	Σ	Σ	грецька заголовна буква сигма
λ	λ	λ	грецька мала буква лямбда
μ	μ	μ	грецька мала літера мю
•	•	•	маркер списку
…	…	…	багатокрапка ...
€	€	& # 8364;	валюта євро

Порядок оформлення роботи:

1. Титульний аркуш.
2. Опис лабораторної роботи.
3. Скриншоти та листинг коду.
4. Висновки.

Контрольні питання

1. З яких частин складається документ HTML?

2. У чому полягають правила сумісності синтаксису HTML з XML?

3. Чим відрізняються логічне та фізичне форматування тексту в документі HTML?

4. Як використовується службова інформація в блоці заголовка документа HTML?

5. Назвіть основні елементи структури сторінки.

6. Які типи списків можуть зустрічатися в документі HTML?

7. За яких умов для зображення краще вибрати один з графічних форматів GIF або JPG?

8. Чому слід завжди визначати альтернативний текст для зображень?

Лабораторна робота 2

Тема: Каскадні таблиці стилів. Практичне використання CSS

Мета роботи: Вивчити способи використання стильової розмітки Навчитися створювати та застосовувати таблиці стилів для керування представленням вмісту веб-сторінок.

Завдання до роботи

Переробити оформлення розробленого сайту на використання CSS, тобто розробити зовнішню таблицю стилів, яку підключити до всіх сторінок сайту. Зовнішній вигляд сторінок має бути ідентичним. (див. завдання до лабораторній роботі №1).

Підключити створені таблиці до веб-сторінки.

Перевірити коректність відображення веб-сторінок у різних браузерах.

Розробити 1 документ, де за допомогою абсолютного позиціонування або плаваючих блоків визначити інформаційну область і область меню. У нижній частині екрана має міститися інформація про автора та авторські права.

Оформити виділення ключових слів всередині тексту розробленого документа за допомогою тега SPAN і завдання відповідного класу в CSS.

Вказівки до роботи

Каскадні таблиці стилів(Cascading Style Sheets, CSS) - це стандарт, що визначає представлення даних у браузері. Якщо HTML надає інформацію про структуру документа, то таблиці стилів повідомляють, як він має виглядати.

Стиль - це сукупність правил, що застосовуються до елемента гіпертексту та визначають спосіб його відображення. Стиль включає всі типи елементів дизайну: шрифт, фон, текст, кольори посилань, поля та розташування об'єктів на сторінці.

Каскадування — це порядок застосування різних стилів до веб-сторінки. Браузер, що підтримує таблиці стилів, буде послідовно застосовувати їх відповідно до пріоритету: спочатку пов'язані, потім

впроваджені і, нарешті, вбудовані стилі. Інший аспект каскадування - успадкування (inheritance), - означає, що якщо не зазначено інше, то конкретний стиль буде застосований до всіх дочірніх елементів гіпертекстового документа. Наприклад, якщо ви застосуєте певний колір тексту в тезі , всі теги всередині блоку будуть відображатися таким же кольором.

Використання каскадних таблиць дає можливість розділити вміст та його представлення та гнучко керувати відображенням гіпертекстових документів шляхом зміни стилів.

Загальний синтаксис CSS

Таблиці стилів будуються відповідно до певного порядку (синтаксисом), інакше вони не можуть нормально працювати. Таблиці стилів складаються з певних елементів (рис. 2.1):

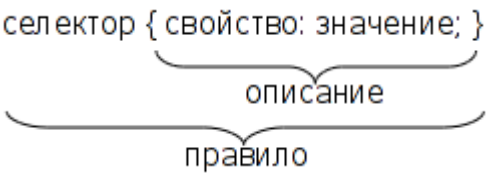

Рисунок 2.1. - Синтаксис опису стилю CSS

– *Селектор (Selector)*. Селектор - це елемент, до якого будуть застосовуватися стилі, що призначаються. Це може бути тег, клас або ідентифікатор гіпертекстового об'єкта документа.

– *Властивість (Property)*. Властивість визначає одну чи кілька характеристик селектора. Властивості задають формат відображення селектора: відступи, шрифти, вирівнювання, розміри тощо.

– *Значення (Value)*. Значення – це фактичні числові чи рядкові константи, що визначають властивість селектора.

– *Опис (Declaration)*. Сукупність властивостей та його значень.

– *Правило (Rule)*. Повний опис стилю (селектор + опис).

– Таким чином, таблиця стилів - це набір правил, що визначають значення властивостей селекторів, перерахованих в цій таблиці. Загальний синтаксис опису правила виглядає так:

– селектор[, селектор[, ...]] {властивість: значення;}

– Регістр символів не має значення, порядок перерахування селекторів у таблиці та властивостей у визначенні не регламентований.

– Правила CSS.

Отже, каскадна таблиця стилів це набір правил форматування тегів HTML. Наведемо кілька прикладів написання таких правил:

1. Основний текст з вирівнюванням по ширині, абзацний відступ 30px, гарнітура (шрифт) – Serif, кегль (розмір шрифту) – 14px:

```
p {
        text-align: justify;
        text-indent: 30px;
        font-family: Serif;
        font-size: 14px;
}
```

Це правило буде застосоване до всіх тегів.

2. Синій колір для заголовків з першого до третього рівня:

```
h1, h2, h3 {
        color: blue; /*те саме, що і #0000FF */
}
```

3. Таблиці та зображення виводити без обрамлення:

```
table, img {border: none;}
```

4. Посилання в елементах списків показувати без підкреслення:

```
li a {text-decoration: none;}
```

5. Внутрішні відступи ліворуч і праворуч для блоків ( ), заголовків таблиць та осередків таблиць встановити в 10px та залити фон жовтим кольором:

```
div, th, td {
        padding-left: 10px;
        padding-right: 10px;
        background-color: yellow;
}
```

6. Всі посилання в документі відображати чорним кольором та напівжирним шрифтом, а в основному тексті та списках - звичайним, а

також виділяти їх зеленим кольором та підкреслювати лише при наведенні курсору.

```
a {color: black; font-weight: bold;}
pa, li a {font-weight: normal; text-decoration: none;}
pa:hover*, li a:hover {
        color: #00FF00; text-decoration: underline;
        }
```

** - В описі правила використаний псевдоелемент (a:hover) - елемент, який не є частиною структури дерева тегів, але має свої методи та властивості*

Стильові класи

Стандарт CSS представляє можливості створення іменованих стилів - стильових класів. Це дозволяє відповісти на таке, наприклад, питання: Як застосувати різні стилі до одного і того ж селектора?

Припустимо, що в документі вам потрібні два різні види основного тексту – один без відступу, другий – з лівим відступом та шрифтом червоного кольору. Для цього потрібно створити правила для кожного з них, наприклад:

```
p {margin-left: 0;}
p.warn {margin-left: 40px; color: #FF00;}
```

Для застосування створеного класу його ім'я потрібно вказати в атрибуті class для вибраних абзаців:

```
<p class=”warn”>Червоний текст із відступом зліва</p>
```

Загальний синтаксис опису класу:

селектор.ім'я_класу {опис}

При створенні класу селектор можна не вказувати, тоді це правило можна застосовувати до будь-якого селектора, що підтримує той самий набір властивостей.

Ось кілька прикладів:

Правило:

```
.solid_blue {color: blue;}
```

Використання:

```
<p class=”solid_blue”>Синій текст абзацу</p>
<li class=”solid_blue”>Синій текст списку</li>
```

Правило:

```
h1.bigsans        {font-family:Sans; font-size: 1.5em;}
h1.smallserif     {font-family:Serif; font-size: .84em;}
```

Використання:

```
<h1 class=”bigsans”>Великий, але рубаний</h1>
<h1 class=”smallserif”>Маленький, але із засічками</h1>
```

Ідентифікатори

Як селектор може бути ідентифікатор елемента гіпертексту, зазначений в атрибуті id. Для призначення стилів таким елементам використовується синтаксис, аналогічний до опису класів, але замість точки ставиться знак # (“решітка”).

Наприклад:

```
div#content {
        position: absolute;
        top: 10px;
        left: 10%;
        right: 10%;
        border: solid 1px silver;
        }
        ...

<div id="content">Текст</div>
```

Слід пам'ятати, що ідентифікатори елементів мають бути унікальними у межах документа.

Угруповання властивостей

*Угруповання (grouping)*полягає в поєднанні значень родинних якостей. При цьому таблиця стилів стає компактнішою, але пред'являються більш жорсткі вимоги до опису правил. Нижче наведено приклад звичайного стилю, що задає відступи:

```
div {
        margin-left: 10px;
        margin-top: 5px;
        margin-right: 40px;
        margin-bottom: 15px;
}
```

Це правило можна переписати з угрупованням у такому вигляді:

div {margin: 5px 40px 15px 10px;} /*порядок: top right bottom left*/

Обидва стилі будуть відображатися однаково.

Угруповання може застосовуватися для таких властивостей, як margin, padding, font, border, background та ін.

Підключення стилів

Існує три способи застосування таблиці стилів до документу HTML:

– *Вбудовування (Inline)*. Цей метод дозволяє застосувати стиль до заданого тегу HTML.

– *Впровадження (Embedded)*. Використання дозволяє керувати стилями сторінки повністю.

– *Зв'язування (Linked або External)*. Пов'язана таблиця стилів дозволяє винести опис стилів у зовнішній файл, посилаючись на який можна контролювати відображення всіх сторінок сайту.

Вбудовані стилі

Вбудовування стилів забезпечує максимальний контроль над усіма елементами веб-сторінки. Вбудований стиль застосовується до будь-якого тегу HTML за допомогою атрибуту style таким чином:

<p style="font: 12pt Courier">Це текст із кеглем 12 точок та гарнітурою Courier</P>

Приклад:

<div style="font-family: Garamond; font-size: 18 pt;>"

Весь текст у цьому розділі має розмір 18 пікселів і шрифт Garamond.

А цей фрагмент ще й виділений червоним кольором.

Вбудовані стилі корисні, коли необхідне тонке настроювання відображення певного елемента сторінки або невеликої веб-сторінки.

Впроваджені стилі

Впроваджені стилі використовують тег <style>, який розміщують у заголовку HTML-документа (<head>...</head>):

<html>

<head>

```
        ...
        <style>
                правила CSS
        </style>
        ...
</head>
<body>
...
```

Приклад використання впроваджених стилів із ЛР №1.

Пов'язані таблиці стилів.

Пов'язані (linked), або зовнішні (external) таблиці стилів - найзручніше рішення, коли йдеться про оформлення цілого сайту. Опис правил міститься в окремий файл (зазвичай, але не обов'язково, з розширенням .css). За допомогою тега <link> виконується зв'язування цієї таблиці стилів з кожною сторінкою, де її потрібно застосувати, наприклад:

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

Будь-яка сторінка, що містить такий зв'язок, буде оформлена відповідно до стилів, зазначених у файлі sample.css. Слід зазначити, що файл зі стилями фізично може знаходитися на іншому веб-сервері, тоді href потрібно вказати абсолютний шлях до нього.

Каскадування.

Якщо вам потрібна сотня-друга-третя сторінок HTML - використовуйте зовнішню, глобальну таблицю стилів. Якщо деякі з цих сторінок вимагають коригування загального оформлення, використовуйте впроваджений стиль. А якщо на сторінці потрібно явно змінити оформлення одного-двох елементів, застосовуйте вбудовані стилі. Саме в такому порядку відбувається перекриття стилів при каскадуванні, схематично це можна так: пов'язані стилі -> впроваджені стилі -> вбудовані стилі

Проблеми із браузерами

Обов'язково переглядайте сторінки з таблицями стилів у різних браузерах. Це пов'язано з тим, що різні браузери можуть по-різному інтерпретувати те саме правило, а деякі властивості та/або значення взагалі не підтримувати. Слід також тестувати сторінки з відключеними стилями (наприклад, текстових браузерах), щоб переконатися, що сторінка читальна.

Апаратно-залежні стилі

Таблиці стилів можуть застосовуватися для керування відображенням вмісту в залежності від пристрою виводу (монітор, проектор, пристрій друку, звуковий синтезатор тощо). Для цього в опис стилів включити тип пристрою, наприклад:

```
@media print {/* друкувальний пристрій */
        BODY { font-size: 10pt; }
        }
@media screen { /* монітор */
        BODY { font-size: 12pt; }
        }

@media screen, print {
        BODY { line-height: 1.2; }
        }
@media all {
        BODY { margin: 1pt; }
        }
```

Як видно з прикладу, вся таблиця розбивається на секції, кожна з яких починається зі слова @media, за яким слідує назва класу пристроїв і далі, у фігурних дужках, безпосередньо опис стилів.

Можна розділити таблиці стилів інакше, вказавши тип пристрою у тезі <link>:

<link rel=stylesheet href="sample.css" type="text/css" media=”screen”>

Властивості CSS

У табл. 2.1 перелічені деякі часто використовувані властивості CSS та їх призначення.

Таблиця 2.1 - Властивості CSS

Ім'я	**Значення**	**Опис**
background	[background-color || background-image || background-repeat || background-attachment || background-position] | inherit	Елемент управління фоном
background-color	<color> | transparent | inherit	Колір фону

background-image	<uri> | none | inherit	Фонове зображення
background-position	[[<%] | <length>]{1,2} | [[top | центр | bottom] || [Left | центр | right]]] | inherit	Положення фонової картинки
background-repeat	Repeat | repeat-x | Repeat-y | no-repeat | inherit	Повторення фонової картинки
border	[border-width || border-style || <color>] | inherit	Кордони елемента
border-collapse	collapse | separate | inherit	Об'єднання/поділ суміжних кордонів
border-color	<color>{1,4} | transparent | inherit	Колір кордону
border-style	<border-style> {1,4} | inherit	Стиль лінії кордону
border-top border-right border-bottom border-left	[Border-top-width || border-style || <color>] | inherit	Управління стилем заданої межі
border-width	<border-width>{1,4} | inherit	Товщина лінії кордону
bottom	<length> | <процент> | auto | inherit	Низ елемента
clear	none | left | right | both | inherit	Заборона заповнення вільного простору поруч із елементом
clip	<Shape> | auto | inherit	Обрізання вмісту елемента
color	<color> | inherit	Колір вмісту
cursor	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| Text | wait | help]] | inherit	Форма курсору
display	inline | block | List-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-	Спосіб відображення елемента

	column-group | table-column | table-cell | table-caption | none | inherit	
empty-cells	шоу | hide | inherit	Відображення порожніх осередків таблиці
float	left | right | none | inherit	Вільне розміщення елемента
font	[[font-style || font-variant || font-weight]? font-size [/line-height]? font-family] | caption | icon | menu | message-box small-caption | status-bar | inherit	Керування шрифтом
font-family	[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>] | inherit	Гарнітура
font-size	<absolute-size> | <relative-size> | <length> | <процент> | inherit	Кегль
font-style	normal | italic | oblique | inherit	Стиль шрифту
font-variant	normal | small-caps | inherit	Варіанти відображення шрифту
font-weight	normal | Болд | Болдер | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	Товщина шрифту
height	<length> | <процент> | auto | inherit	Ширина елемента
left	<length> | <процент> | auto | inherit	Положення лівої межі елемента
line-height	normal | <number> | <length> | <процент> | inherit	Висота рядка
list-style	[list-style-type || list-style-position || list-style-image] | inherit	Стиль списку
margin	<margin-width>{1,4} | inherit	Зовнішній відступ
margin-top margin-right margin-bottom margin-left	<margin-width> | inherit	Зовнішній відступ по заданій стороні

padding	<padding-width>{1,4} | inherit	Внутрішній відступ
padding-top padding-right padding-bottom padding-left	<padding-width> | inherit	Внутрішній відступ по заданій стороні
position	static | relative | absolute | fixed | inherit	Позиціонування елемента
right	<length> | <процент> | auto | inherit	Положення правого кордону
text-align	left | right | центр | Justify | <string> | inherit	Вирівнювання текстового блоку
text-decoration	none | [underline || overline || line-through || blink] | inherit	Текстові ефекти
text-indent	<length> | <процент> | inherit	Абзацний відступ
text-transform	capitalize | uppercase | lowercase | none | inherit	Накреслення тексту
top	<length> | <процент> | auto | inherit	Положення верхньої межі елемента
vertical-align	Baseline | sub | Super | top | text-top | middle | bottom | text-bottom | <процент> | <length> | inherit	Вертикальне вирівнювання в межах блоку
visibility	visible | hidden | collapse | inherit	Керування видимістю елемента
white-space	normal | pre | nowrap | inherit	Управління пробілами між словами
width	<length> | <процент> | auto | inherit	Ширина елемента
z-index	auto | <integer> | inherit	Порядок переходу на клавішу Tab

Позиціювання елементів

Властивість position у поєднанні з властивостями left, top, right, bottom, display, clear та інших дозволяє керувати положенням елементів на сторінці і порядком їх виведення. Властивість position може приймати такі значення:

1) static - нормальне становище. Цей блок є звичайним блоком, він відображається відповідно до загальних правил. Властивості 'left' та 'top' не застосовуються.

2) relative - відносне позиціонування. Положення блоку розраховується відповідно до нормального потоку виведення. Потім блок зміщується щодо свого нормального (static) положення.

3) absolute - абсолютне позиціонування. Положення блоку (можливо і розмір) вказується за допомогою властивостей 'left', 'right', 'top' та 'bottom'. Вони вказують величину усунення щодо контейнера блоку. Абсолютно блоки, що позиціонуються, вилучаються з нормального потоку. Це означає, що вони впливають розміщення наступних елементів тієї самої рівня.

4) fixed - фіксоване положення.Положення блоку розраховується відповідно до моделі абсолютного позиціонування, а потім він фіксується щодо області перегляду або сторінки.

Два оголошення можуть бути відокремлені один від одного за допомогою правила @media, як це показано у прикладі:

@media screen { H1#first { position: fixed; } }

@media print { H1#first { position: static; } }

Керуючи позиціонуванням, можна по-різному розміщувати блоки інформації на сторінці, аж до створення ефектів накладання, перетікання, градієнта тощо.

Контрольні питання

1. Чим відрізняються дії властивостей display:none та visiblity:hidden?
2. На веб-сторінці розміщено зображення завширшки 200px. Як задати йому обтікання текстом з правого боку?
3. Як помістити елемент веб-сторінки (наприклад, <p>) за видиму область екрана?

Лабораторна робота 3

Тема: Робота з веб-формами

У рамках цієї лабораторної роботи ми розглянемо процес створення веб-форм засобами HTML.

Метою даної лабораторної роботи є навчання таким основам роботи з веб-формами, як:

- створення веб-форм;
- стилізація форм;
- *валідація* значень, що вводяться;
- завдання маски введення.

Завдання до лабораторної роботи:

- створити сторінку реєстрації з веб-формами зробити власну форму. Необхідно також створити відповідні стилі для оформлення веб-форми і валідацію значень, що вводяться.

Вказівки до роботи

На рисунку 3.1 наведено проста форма реєстрації.

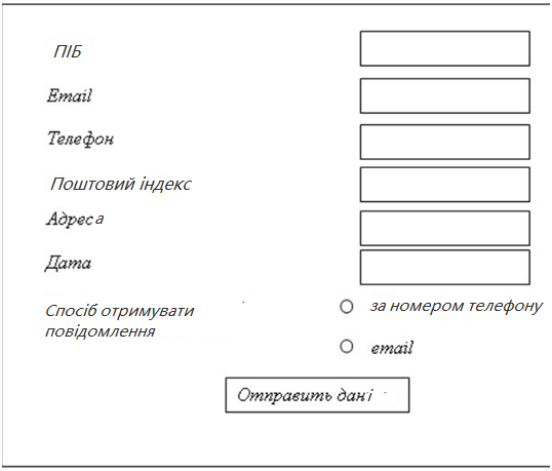

Рисунок 3.1 - Зразковий макет веб-сторінки для виконання завдання

1. Створення веб-форм

Розглянемо кроки створення необхідної сторінки з реєстраційною формою.

Крок 1

Безпосередньо створимо форму реєстрації.

```
<form id="registration">
 <fieldset>
 <legend>Форма реєстрації</legend>
 </fieldset>
</form>
```

Тег <fieldset> використовується для логічного угруповання об'єктів форми.

Тег <legend> визначає заголовок.

Крок 2

Додамо поле для введення ПІБ користувача:

```
<fieldset>
 <legend>Форма реєстрації</legend>

 <label for=fio>ПІБ</label>
 <input id=fio name=fio type=text>
</fieldset>
```

На даному етапі форма буде виглядати в браузері таким чином:

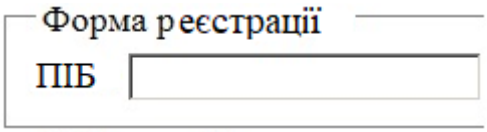

Рисунок 3.2 - Проміжний результат на кроці 2

Крок 3

Створимо поля для введення email та номери телефону, поштового індексу та міста:

```
<fieldset>
 <legend>Форма реєстрації</legend>

 <label for=fio>ПІБ</label>
```

```
<input id=fio name=fio type=text>

<label for=email>Email</label>
<input id=email name=email type=email>

<label for=phone>Номер телефону</label>
<input id=phone name=phone type=tel>
</fieldset>
```

Як можна бачити по наступному малюнку, результат трохи інший, ніж ми очікували:

Рисунок 3.3 - Проміжний результат на кроці 3

Крок 4

Для вирівнювання елементів форми рядково і один відносно одного необхідно створити таблицю з двома колонками і без видимих кордонів. Змінимо код наступним чином:

```
<fieldset>
<legend>Форма реєстрації</legend>
<table class="alignment">
<tr>
<td> <label for=fio>ПІБ</label> </td>
<td> <input id=fio name=fio type=text></td>
</tr>
<tr>
<td><label for=email>Email</label></td>
<td> <input id=email name=email type=email> </td>
</tr>
<tr>
<td> <label for=phone>Номер телефону</label> </td>
<td> <input id=phone name=phone type=tel></td>
</tr>
</table>
</fieldset>
```

В результаті отримаємо наступне:

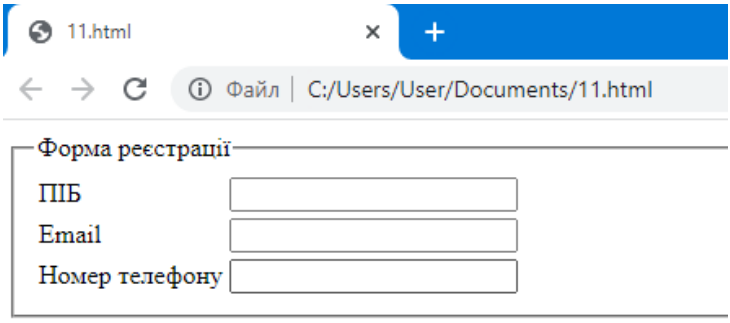

Рисунок 3.4 - Проміжний результат на кроці 4

Крок 5

Додамо поле для введення дати народження, для цього використовуємо type=date (не працює з усіма браузерами). Додамо до таблиці наступний рядок:

```
<tr>
 <td><label for=dateofbirth>Дата народження</label></td>
 <td><input id=dateofbirth name=dateofbirth type="date" /></td>
 </tr>
```

Крок 6

Для логічного угруповання, додамо ще один fieldset, що об'єднує елементи, що вказують адресу та поштовий індекс:

```
<fieldset>
 <legend>Параметри доставки</legend>
 <table>
 <tr>
 <td><label for=address>Адреса</label></td>
 <td><textarea id=address name=address rows=5 ></textarea></td>
 </tr>
 <tr>
 <td><label for=postcode>Поштовий індекс</label></td>
 <td><input id=postcode name=postcode type=text ></td>
 </tr>

 </table>
```

</fieldset>

Оскільки написання повної адреси, швидше за все, займе не один рядок, задамо атрибут *rows*, зі значенням рівним 5 для відповідного тегу.

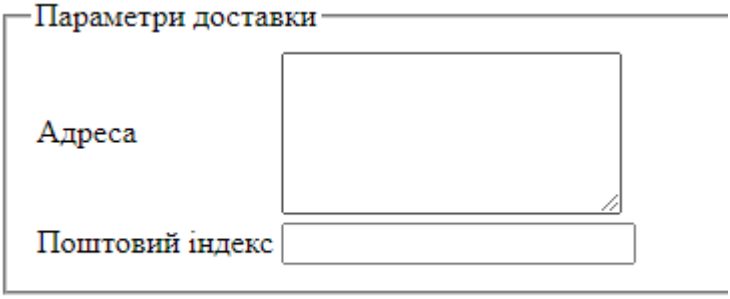

Рисунок 3.5 -Проміжний результат на кроці 6

Крок 7

Також в окрему логічну групу виділимо елемент управління, за допомогою якого користувач вкаже бажаний спосіб отримання повідомлень:

```
<fieldset>
 <legend>Уподобаний спосіб отримання повідомлень</legend>
 <table>
 <tr>
 <td><input id=emailmessage name=message type=radio></td>
 <td><label for=emailmessage>По Email</label></td>
 </tr>
 <tr>
 <td><input id=phonemessage name=message type=radio></td>
 <td><label for=phonemessage>По телефону</label></td>
 </tr>
 <tr>
 <td><input id=nomessage name=message type=radio></td>
 <td><label for=nomessage>Не повідомляти мене</label></td>
 </tr>
 </table>
 </fieldset>
```

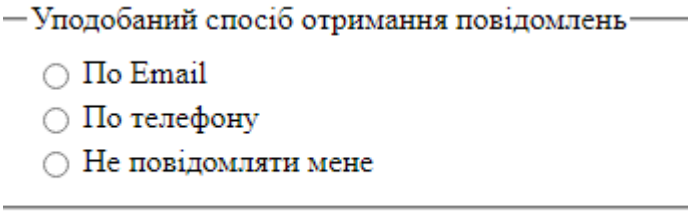

Рисунок 3.6 - Проміжний результат на кроці 7

Крок 8
Залишилось тільки додати на сторінку кнопку:
```
<fieldset>
 <button type=submit>Надіслати дані</button>
 </fieldset>
```

На цьому створення форми та розміщення елементів управління закінчується. Можна перейти до наступного пункту завдання.

Стилізація форм
Крок 1
Створимо стиль для всієї форми - задамо колір фону, ширину і т.д.
```
form#registration {
 background: #1E90FF;
 -moz-border-radius: 5px;
 -webkit-border-radius: 5px;
 -khtml-border-radius: 5px;
 border-radius: 5px;
 counter-reset: fieldsets;
 padding: 20px;
 width: 400px;
}
```

Крок 2
Приберемо кордони біля fieldset і додамо відступ:
```
form#registration fieldset {
 border: none;
```

```
 margin-bottom: 10px;
}
```

Крок 3

Стилізуємо наші*legend*Ти зробимо їх жирними та застосуємо до них світло синій *text-shadow*:

```
form#registration legend {
 color: #384313;
 font-size: 16px;
 font-weight: bold;
 padding-bottom: 10px;
 text-shadow: 0 1px 1px #C1F7FF;
}
```

Крок 4

Залишилось стилізувати елементи *label,inputibutton*. Усе*label*повинні виглядати однаково, крім*label*я який прописаний для елементів radio. Вирівняємо їх по лівому краю і надамо ширину:

```
form#registration label {
 float: left;
 font-size: 13px;
 width: 110px;
 }

form#registration fieldset label {
 background:none no-repeat left 50%;
 line-height: 20px;
 padding: 0 0 0 30px;
 width: auto;
 }

 form#registration button {
 background: #87CEEB;
 border: none;
 -moz-border-radius: 20px;
 -webkit-border-radius: 20px;
 -khtml-border-radius: 20px;
 border-radius: 20px;
```

```
color: #ffffff;
display: block;
font: 18px Georgia, "Times New Roman", Times, serif;
letter-spacing: 1px;
margin: auto;
padding: 7px 25px;
text-shadow: 0 1px 1px #C1F7FF;
text-transform: uppercase;
```

В результаті форма виглядатиме таким чином:

Подальші експерименти зі стилізацією форми залишаються на самостійний розгляд.

Валідація значень, що вводяться

Розглянемо наступну ділянку нашого HTML коду:

```
<input id=email name=email type=email />
```

Атрибут типу дорівнює "email",а не"text". Найцінніше у нових HTML-типах *input* в тому, що ви можете використовувати їх зараз, і вони будуть працювати на тому чи іншому рівні в будь-якому браузері.

Коли браузер зустрічає один із цих типів і не підтримує нові input-типи, оголошення типу не розпізнається. У такому разі браузер коректно скорочує функціональність та інтерпретує елемент як type="text".

Щоб зробити елемент управління обов'язковим до заповнення, достатньо вставити в тег, що створює його, атрибут *REQUIRED*. Це атрибут тега без значення.

```
<input id=name name=name type=text required>
```

Задати мінімальне та максимальне значення та крок числових значень можна тільки для полів введення числових величин. Інші елементи керування, зокрема звичайні поля введення, цю можливість не підтримують.

Для завдання мінімального значення числа використовується атрибут тега *MIN*, а для завдання максимального значення – атрибут тега MAX. Як їх значення вказуються числа.

```
<INPUT TYPE="number" ID="txtAge" REQUIRED MIN="1" MAX="100" >
```

Завдання маски введення

*Маска*введення задає формат, якому має відповідати значення, що вводиться. Як правило, вона вказується для звичайних полів уведення.

Для вказівки маски використовується атрибут стилю *PATTERN*. Як його значення вказується регулярне вираження, що власне, і задає маску введення.

<input id=phone name=phone type=tel pattern=" [(]\d{3,4}[)]\s\d{2,3}[-]\d{2}[-]\d{ 2}" >

Ця веб-форма містить поле введення, де вказується номер телефону у форматі([x]xxx) [x]xx-xx-xx, дех- Цифра.

Якщо відвідувач введе в поле введення з вказаною маскою значення, що не відповідає даній масці, побачить спливаюче повідомлення з попередженням про неправильний формат введеного значення.

Для успішного завершення виконання завдання самостійно необхідно зробити таке:

1) Поля ПІБ, email та номер телефону, визначити як обов'язкові до заповнення.
2) Створити маску введення поля Номер телефону.
3) Здійснити перевірку значень поля Поштовий індекс, що вводяться.
4) Ознайомитись з матеріалами для самостійного вивчення.

Контрольні питання.

1. За допомогою якого атрибуту можна проконтролювати ввід даних в форму?
2. Які атрибути є в HTML5 для форм?
3. Як захистити веб-форми від спама?
4. Які є інструменти розробки форм?
5. Як створити простий ковзанок засобами HTML?

Лабораторна робота № 4

Тема: Розроблення динамічних веб-сторінок за допомогою мови JavaScript

Метою даної лабораторної роботи є закріплення теоретичного матеріалу з попереднього лекційного матеріалу, вивчення основ мови сценаріїв JavaScript для подальшого застосування під час розробки веб-сайтів.

Завдання до лабораторної роботи

1. Для реалізації завдань додати на сайт, створений в рамках лабораторних робіт № 1,2,3, сторінку з назвою "JavaScript.html".
2. Створити форму з полем уведення та двома кнопками (типу ОК і ВІДМІНИТИ). Натисканням на кнопку типу ОК у формі заповнити текстове поле інформацією (а) відповідно до варіанту.
3. Створити файл "lr4.js", в якому реалізувати завдання (б) відповідно до варіанту, і підключити цей файл до JavaScript.html.
4. Додати гіперпосилання, при натисканні на яке у вікні вивести інформацію (С) відповідно до варіанту.

ВАРІАНТИ

Варіант 1

А. Кількість днів (обчислити в скрипті), що минули від дня вашого народження.

Б. Написати скрипт, який визначає кількість днів до найближчої неділі, і вивести це число під час завантаження сторінки у вікні повідомлення.

С. Повідомлення назви поточного місяця.

Варіант 2

А. Кількість тижнів (обчислити в скрипті), які пройшли від дня вашого народження.

Б. Написати скрипт, який визначає, скільки днів минуло з Нового року, і вивести це число при завантаженні сторінки у вікні повідомлення.

С. Повідомлення кількості годин і хвилин, що пройшли з початку доби.

Варіант 3

А. Кількість годин і хвилин, що пройшли з початку поточного місяця.

Б. Написати скрипт, який визначає кількість годин (за ґрінвічським часом) до настання Нового року, і вивести це число при завантаженні сторінки у вікні повідомлення.

С. Повідомлення поточних координат курсора.

Варіант 4

А. Повна інформація щодо поточної дати та часу. Наприклад, "14 травня 2021 року, вівторок, 2:53:44 pm".

Б. Написати скрипт, який визначає, скільки тижнів залишилося до 1 вересня поточного року.

С. Повідомлення значення поточного елемента списку.

Варіант 5

А. Кількість тижнів, що минули з останнього 1 вересня.

Б. Написати скрипт, який визначає, чи є поточний рік роком проведення літньої Олімпіади (високосним), і вивести це число під час завантаження сторінки у вікні повідомлення.

С. Повідомлення значення поточного елемента списку.

Варіант 6

А. Кількість годин (обчислити в скрипті), що залишилися до початку літа.

Б. Написати скрипт, який визначає, скільки діб залишилось до днів весняного й осіннього рівнодення (22 березня та 22 вересня),і вивести це число при завантаженні сторінки у вікні повідомлення.

С. Повідомлення щодо кількості спрацьовувань обробника даної події в поточну хвилину.

Варіант 7

А. Кількість годин і хвилин, що пройшли з початку поточного місяця.

Б. Написати скрипт, який визначає: півріччя (перше або друге); квартал (перший, другий, третій чи четвертий); сезон (зима, весна, літо або осінь); сторіччя; тисячоліття; і вивести цю інформацію під час завантаження сторінки у вікні повідомлення.

С. Повідомлення поточної секунди.

Варіант 8

А. Скільки днів залишилося до найближчої п'ятниці, яка випадає на 13-те число.

Б. Написати скрипт, який визначає, скільки днів залишилось до

найближчого 23 серпня.

С. Повідомлення поточного часу.

Варіант 9

А. Кількість днів, що залишились до літніх канікул (уточніть дату початку канікул в деканаті).

Б. Написати скрипт, який визначає кількість годин, що залишились до кінця поточного місяця, і вивести це число при завантаженні сторінки у вікні повідомлення.

С. Повідомлення випадкового числа від 1 до кількості днів у поточному місяці.

Варіант 10

А. Кількість хвилин, що минули з початку пари.

Б. Написати скрипт, який визначає, скільки днів минуло з останнього 14 лютого.

С. Повідомлення щодо кількості спрацьовувань обробника даної події в поточному сеансі.

Варіанти обирати за списком групи. Після 10 номера буде знову 1.

Вказівки до роботи

JavaScript – це мова управління сценаріями перегляду гіпертекстових сторінок Web на боці клієнта. Якщо бути більш точним, JavaScript – це не тільки мова програмування на боці клієнта. Liveware, предок JavaScript, є засобом підстановок на боці сервера Netscape. Однак найбільшу популярність JavaScript забезпечило програмування на боці клієнта.

Розміщення коду JavaScript на HTML-сторінці

У загальному випадку можна виділити чотири способи функціонального застосування JavaScript:

1) гіпертекстове посилання (схема URL);
2) обробник події (в атрибутах, що відповідають подіям);
3) вставка (контейнер <SCRIPT>).

Коментарі в HTML і JavaScript

У програмі на JavaScript можна залишати коментарі, які ігноруються JavaScript-інтерпретатором і слугують поясненням для розробників. Однорядкові коментарі починаються з символів //. Текст, починаючи з цих символів і до кінця рядка, вважається коментарем.

Багаторядковий коментар укладається між символами/* і */і може займати кілька рядків.

Типи даних і оператори

Як і будь-яка інша мова програмування, JavaScript підтримує вбудовані структури та типи даних. Усе їх різноманіття поділяється на: літерали; змінні; масиви; функції; об'єкти.

Вони розрізняються на: вбудовані та визначені програмістом.

Літерали

Літералом називають дані, які використовуються безпосередньо в програмі. При цьому під даними розуміють числа або рядки тексту. Усі вони розглядають в JavaScript як елементарні типи даних.

Літерали використовуються в операціях присвоювання значень змінним або в операціях порівняння:

```
var a=10;
var str = 'Рядок';
if(x=='test') alert(x);
```

Оператор присвоювання (змінна = вираз) повертає результат обчислення виразу, тому ніщо не заважає отримане значення надати ще й іншій змінній.

Крім рядкових літералів (послідовностей символів, укладених в лапки) є ще рядкові об'єкти; вони створюються конструктором: **var s = new String()**. Цей об'єкт має багато методів. Слід розуміти, що рядковий літерал та рядковий об'єкт – далеко не те ж саме.

Змінні

Змінна – це область пам'яті, що має своє ім'я та зберігає певні дані. Змінні в JavaScript оголошуються за допомогою оператора var, при цьому можна надавати чи не надавати їм початкові значення:

```
var k;
var h='Привіт!';
```

Тип змінної визначається за присвоєним їй **значенням.** Мова JavaScript – **слабо типізована**: у різних частинах програми можна привласнювати тій самій змінній значення різних типів, і інтерпретатор буде "на льоту" змінювати тип змінної. Дізнатися тип змінної можна за допомогою оператора **typeof().**

Змінна, оголошена оператором **var** поза функціями, є глобальною – її "видно" повсюди в скрипті. Змінна, оголошена оператором **var** всередині функції, є локальною – її видно тільки в межах цієї функції.

Оголошувати змінні можна і без оператора **var**, просто привласнюючи змінній початкове значення. Однак опускати var не рекомендується.

Масиви

Масиви розділяються на вбудовані (document.links [], document.images [] тощо – їх ще називають колекціями) і визначені користувачем (автором документа). Для масивів призначено декілька методів: **join(), reverse(), sort()** та інші, а також властивість **length**, яка дозволяє отримати кількість елементів масиву.

Для визначення масиву користувача існує спеціальний конструктор **Array**. Якщо йому передається один аргумент, причому ціле невід'ємне число, то створюється незаповнений масив відповідної довжини. Якщо ж передається один аргумент, який не є числом, або більше одного аргументу, то створюється масив, заповнений цими елементами:

a **= new** Array**();** // порожній масив (довжини 0) b **= new** Array**(**10**);** // масив довжини 10

c **= new** Array**(**10**,**'Привіт'**);** // масив з двох елементів: числа та рядки

//Короткий спосіб створити масив з 4-х елементів d **= [**5**,** 'Тест'**,** 2.71828**,** 'Кількість e'**];**

Елементи масиву нумеруються від нуля. Тому в останньому прикладі значення **d [0]** одне **5**, а значення **d [1]** одне **'Тест'**. Отже, масив може складатися з різнорідних елементів. Масиви не можуть бути багато вимірюваними, проте можна завести масив, елементами якого також будуть масиви.

Метод join() дозволяє об'єднати елементи масиву в один рядок. Він є зворотним до розглянутого вище методу split(), який розрізає об'єкт типу String на шмати та складає з них масив. До речі, метод split() демонструє той факт, що масив можна отримати і без конструктора масиву.

Метод reverse() застосовується для зміни порядку елементів масиву на протилежний. Впорядкувати його у зворотному порядку можна, викликавши метод a.reverse().

Метод sort() інтерпретує елементи масиву як рядкові літерали та сортує масив у алфавітному (лексикографічному) порядку. Слід зауважити, що метод sort() змінює масив. Метод sort() інтерпретує елементи масиву як рядки (і здійснює лексикографічне сортування), але не

перетворює їх на рядки. Якщо в масиві були числа, то вони залишаться числами.

Оператори умови

Основна увага приділяється операторам декларування й управління потоком обчислень. Без них не може бути написана жодна JavaScript- програма.

Перелік основних операторів виглядає наступним чином:

{...}
if... else...
()?
while for
break
continue return

Оператор {...}

Фігурні дужки визначають складений оператор JavaScript-блок. Основне призначення блоку – визначення тіла циклу, тіла умовного оператора або функції.

Оператор if... else...

Умовний оператор застосовується для розгалуження програми за певною логічною умовою. Є два варіанти синтаксису:

if (логічний_вираз**)** оператор_1**;**
if (логічний_вираз**)** оператор_1**; else** оператор_2**;**

Логічний вираз – це вираз, який приймає значення **true** або **false**.

Оператор()?

Цей оператор, званий умовним виразом, видає одне з двох значень залежно від виконання певної умови. Синтаксис його такий:

логічний_вираз**)?** значення_1 **:** значення_2

Якщо логічний_вираз одне **true**, то повертається значення_1, в іншому випадку – значення_2. Умовний вираз легко імітується оператором **if... else**, проте він дозволяє зробити більш компактним і легко сприйнятним код програми.

Оператор **while** задає цикл, який визначається у загальному випадку наступним чином:

while (умова_продовження_циклу) тіло циклу**;**

Тіло циклу може бути як простим, так і складним оператором. Складний оператор, як правило, вкладається у фігурні дужки. Рекомендується і простий оператор укладати в них, щоб програму можна

було легко модифікувати. Умова_продовження_циклу є логічним виразом. Тіло виповнюється доти, доки правильною є логічна умова.

Оператор **for** – це ще один оператор циклу. У загальному випадку він має вигляд:

for (ініціалізація_змінних_циклу; умова_продовження_циклу; модифікація_змінних_циклу)

тіло_циклу**;**

Тіло циклу може бути як простим, так і складним оператором (складні необхідно укладати у фігурні дужки). Оператори ініціалізація_змінних_циклу і модифікація_змінних_циклу можуть складатися з декількох простих операторів, у цьому випадку прості оператори повинні бути розділені комою. Умова_ продовження_циклу є логічним виразом.

Оператор **break** дозволяє достроково покинути тіло циклу.

Оператор **continue** дозволяє перейти до наступної ітерації циклу, пропустивши виконання всіх нижчерозташованих операторів в тілі циклу.

Оператор **return** використовують для повернення значення з функції або обробника події. Зверніть увагу: оператор **return** не тільки вказує, яке значення має повернути функція, а й припиняє виконання подальших операторів в тілі функції. Під час використання в обробниках подій оператор **return** дозволяє скасувати або не скасовувати дію за замовчуванням, яка створює браузер під час виникнення даної події.

Контрольні запитання

1. Назвіть способи розміщення коду JavaScript на HTML-сторінці.
2. Як додати коментарі до коду JavaScript?
3. Чому JavaScript є слабоуніверсальна мова?
4. Як дізнатися тип змінної в JavaScript?
5. Що таке літерали?
6. Для чого використовують оператор ***var***?
7. Назвіть особливості використання оператора ***var*** усередині функцій.
8. Які види масивів існують в JavaScript?
9. Назвіть методи об'єкта "масив" і призначення цих методів.
10. Які вам відомі оператори в JavaScript?

Лабораторна робота 5

Тема: Розроблення динамічних веб-сторінок за допомогою мови JavaScript та DOM API

Мета лабораторної роботи - закріплення теоретичного матеріалу з попереднього лекційного матеріалу, вивчення об'єктної моделі документа та засобів роботи з нею в мові сценаріїв JavaScript при розробленні динамічних веб-сайтів.

Завдання до роботи

1. У сайт, створений в рамках попередніх лабораторних робіт, додати наступну функціональність:

- Додати блок ,у якому під час завантаження сторінки вставити поточну дату.
- Після наведення миші на область з датою в ній починає відображатися поточний час (год; хв; с), який оновлюється кожну секунду. Після відведення курсора миші з даної області в ній знову відображується поточна дата.
- Додати на сайт логотип або інше (більш слушне, на ваш погляд) зображення, реалізоване декількома зображеннями. Після наведення курсору миші на логотип частини зображення, з яких він складається, повинні замінюватися на більш яскраві (більш цікаві рішення вітаються).
- Розробити функцію, яка при наведенні миші на гіперпосилання "Про сайт" має показувати під ним блок із короткою інформацією про авторів. Це має виглядати як спливаюче вікна, але без додаткових вікон, а з використанням тільки блоків HTML.
- Завантаження зображень повинно бути оптимізоване.

Вказівки до роботи

Для реалізації інтерактивних, динамічних веб-сторінок застосовується так званий Dynamic HTML, або DHTML.

Dynamic HTML – це спосіб створення інтерактивного веб-сайту, який використовує поєднання статичної мови розмітки HTML, вбудованої (і виконуваної на боці клієнта) зі скриптовою мовою JavaScript, CSS (каскадних таблиць стилів) і DOM (об'єктною моделлю документа).

Об'єкти в JavaScript

Об'єкт – це головний тип даних JavaScript. Тип даних Object сам визначає об'єкти. Об'єкт в JavaScript є звичайним асоціативним масивом ("хеш"). Він зберігає будь-які відповідності "ключ => значення" і має кілька стандартних методів. У сценарії JavaScript можуть використовуватися об'єкти декількох видів:

– клієнтські об'єкти входять до моделі DOM, тобто відповідають тому, що міститься або відбувається на Web-сторінці у вікні браузера. Вони створюються браузером під час розбору (парсингу) HTML-сторінки. Приклади: window, document, location, navigator тощо;

– серверні об'єкти відповідають за взаємодію клієнт-сервер. Приклади: Server, Project, Client, File тощо. Серверні об'єкти в цьому курсі не розглядаються;

– вбудовані об'єкти – це різноманітні типи даних, властивостей, методів, притаманних самій мові JavaScript незалежно від вмісту HTML- сторінки. Приклади: вбудовані класи об'єктів Array, String, Date, Number, Function, Boolean, а також вбудований об'єкт Math;

– користувальницькі об'єкти створюються програмістом у процесі написання сценарію з використанням конструкторів типу об'єктів (класу).

Оператори роботи з об'єктами for... in...

Оператор for (змінна in об'єкта) дозволяє "пробігтися" по властивостям об'єкта.

Оператор with задає об'єкт за замовчуванням для блоку операторів, визначених у його тілі. Синтаксис його такий:

with (об'єкт) оператор;

Властивості та методи, притаманні тілу оператора, повинні бути записаними повністю, інакше вони вважатимуться властивостями та методами об'єкта, зазначеного в операторі with. Оператором with корисно користуватися під час роботи з об'єктом Math, використовуваним для доступу до математичних функцій і констант.

Клієнтські об'єкти

Для створення механізму управління сторінками на клієнтському боці використовується об'єктна модель документа (DOM – Document Object Model). Сутність моделі в тому, що кожному HTML-контейнеру відповідає об'єкт, який характеризується трійкою: властивості; методи; події.

Об'єктну модель можна уявити як спосіб зв'язку між сторінками та браузером. Об'єктна модель документа – це подання об'єктів, їх методів, властивостей і подій, які присутні і відбуваються в програмному забезпеченні браузера, у вигляді зручного для роботи з ними з коду HTML і вихідного тексту сценарію на сторінці. З її допомогою можна ставити будь-які вимоги до браузера та передавати їх на клієнські сторінки. Браузер виконає команди і, відповідно, змінить сторінку на екрані.

Об'єкти з однаковим набором властивостей, методів і подій об'єднуються в класи однотипних об'єктів. *Класи* – це описи можливих об'єктів. Самі об'єкти з'являються тільки після завантаження документа браузером або як результат роботи програми. Про це треба завжди пам'ятати, щоб не звертатися до неіснуючого об'єкта.

Ієрархія класів DOM

Об'єктно-орієнтована мова програмування передбачає наявність ієрархії класів об'єктів. У JavaScript така ієрархія починається з класу об'єктів **Window**, тобто кожен об'єкт приписаний до того чи іншого вікна. Для звернення до будь-якого об'єкта або його властивості вказують повне або часткове ім’я цього об'єкта або його властивості, починаючи з імені об'єкта, старшого за ієрархією, до якої належить даний об'єкт

Однак JavaScript не є класичною об'єктною мовою (її ще називають полегшеною об'єктною мовою). У неї відсутні успадкування та поліморфізм. Є лише тези "об'єкт А містить об'єкт В". Вона не є ієрархією класів у буквальному сенсі. У об'єктів DOM певні властивості обов'язково присутні, тоді як наявність інших залежить від веб-сторінки. Наприклад, об'єкт Window завжди має в якості своїх властивостей об'єкти location і history, тобто це обов'язкові властивості. Якщо HTML-сторінка містить контейнер <BODY>, то в об'єкті Window буде присутній як властивість об'єкт document. Якщо HTML-сторінка містить контейнер <FRAMESET> з вкладеними в нього контейнерами <FRAME>, то в об'єкті Window будуть присутні в якості властивостей імена фреймів, наприклад window.f1. Останні самі є об'єктами класу Window, і для них, в свою чергу, справедливо все вищесказане.

Примітка. Кожен браузер, будь то Internet Explorer, Mozilla Firefox або Opera, має свою об'єктну модель. Об'єктні моделі різних браузерів (і навіть різні версії одного) відрізняються між собою, але мають принципово однакову структуру. Тому немає сенсу зупинятися на кожній з них окремо.

Колекції

Колекція – це структура даних JavaScript, подібна до масиву. Відмінність колекції від масивів полягає в тому, що масиви програміст створює сам в коді програми і заповнює їх даними. Колекції створюються браузером і "заселюються" об'єктами, пов'язаними з елементами веб-сторінки. Колекцію можна розглядати як більш зручний спосіб доступу до об'єктів веб-сторінки.

Наприклад, якщо на сторінці є форми з іменами f, g5 і h32, то в об'єкта document є відповідні властивості-об'єкти document.f, document.g5 тощо. Окрім цього, в об'єкта document є властивість forms, яка є колекцією (масивом) усіх форм. Отже, до тих самих об'єктів форм можна звернутися як document.forms[0], document.forms[1] тощо. Це зручно, при виконанні дій з усіма об'єктами форм на даній сторінці. При визначенні властивостей того чи іншого об'єкта колекції будуть писатись з дужками: forms[], images[], frames[], щоб підкреслити, що це не звичайні властивості, а колекції.

Нумеруються елементи колекції, починаючи з нуля, в порядку їх появи в початковому HTML-файлі. Доступ до елементів колекцій здійснюється або за індексом (в круглих або квадратних дужках), або за ім’ям (теж в круглих або квадратних дужках чи через точку). Як і у звичайних масивів, у колекцій є властивість length, яка дозволяє дізнатись про кількість елементів в колекції.

Властивості

Більшість HTML-контейнерів мають атрибути. Як вже відомо, кожному контейнеру відповідає об'єкт. Отже, відповідним атрибутам відповідають властивості об'єкта. Відповідність між атрибутами HTML-контейнерів і властивостями DOM-об'єктів не завжди пряма.

Зазвичай кожному атрибуту відповідає певна властивість об'єкта. Але, по-перше, назву цієї властивості не завжди легко зрозуміти за назвою атрибута, а по-друге, у об'єкта можуть бути властивості, що не мають аналогів серед атрибутів. Крім того, атрибути є регістрозалежні або регістронезалежні, як і вся мова HTML, тоді як властивості об'єктів потрібно писати в точно визначеному регістрі символів. До властивостей можна також звертатися за допомогою дужкової нотації: об'єкт ['властивість']. У об'єктів, що відповідають гіперпосиланнями, також є властивості, які не мають аналогів серед атрибутів. Повний перелік властивостей об'єктів класу URL розміщений в довіднику з JavaScript.

Методи

У термінології JavaScript методи об'єкта визначають функції, за допомогою яких виконуються дії з цим об'єктом, наприклад: зміна його властивостей, відображення їх на веб-сторінці, відсилання даних на сервер, перезавантаження сторінки тощо.

Події

Окрім методів і властивостей, об'єкти характеризуються подіями. Власне, сутність програмування на JavaScript полягає в написанні обробників цих подій. Наприклад, з об'єктом типу *button* (контейнер INPUT типу *button* – "кнопка") може відбуватися подія **Click**, тобто користувач може натиснути на кнопку. Для цього атрибути контейнера INPUT розширені атрибутом обробляння цієї події – **onClick**. Як значення цього атрибута вказується програма обробляння події, яку повинен написати на JavaScript автор HTML-документа.

Обробники подій вказуються в спеціально створених для цього атрибутах у тих контейнерах, з якими ці події пов'язані. Наприклад, контейнер BODY визначає властивості всього документа, тому обробник події "Завершено завантаження всього документа" вказується в цьому контейнері як значення атрибута **onLoad**.

Прототип

Зазвичай розробники використовують вбудовані об'єкти JavaScript (крім ієрархії об'єктів DOM), такі, як: **Data, Array і String**. У цьому сенсі цікавою є властивість об'єктів під назвою prototype. *Прототип* – це інша назва конструктора об'єкта конкретного класу. Є один істотний нюанс: новими методами та властивостями будуть володіти тільки ті об'єкти, які породжуються після зміни прототипу об'єкта. Усі вбудовані об'єкти створюються до того, як JavaScript- програма отримає управління, що істотно обмежує застосування властивості **prototype**.

Об'єкт window

Клас об'єктів Window – це найстарший клас в ієрархії об'єктів JavaScript. Об'єкт window, що відноситься до поточного вікна (тобто в якому виконується скрипт), є об'єктом класу Window. Клас об'єктів Frame міститься в класі Window, тобто кожен фрейм – це теж об'єкт класу Window.

Об'єкт window створюється тільки в момент відкриття вікна. Усі інші об'єкти, які породжуються під час завантаження сторінки, є властивостями об'єкта window. Більш того, всі глобальні змінні, визначені в даному вікні, теж є властивостями об'єкта window. Таким чином, у об'єкта window можуть бути різні властивості під час завантаження різних

сторінок. Крім того, в різних браузерах властивості об'єктів і поведінка об'єктів і браузера при оброблянні подій може бути різним.

Оскільки об'єкт window є найстаршим, то в більшості випадків, звертаючись до його властивостей і методів, приставку "window." можна опускати (зрозуміло, в разі, якщо треба звернутися до властивості або методу поточного вікна, де працює скрипт; якщо ж це інше вікно, то необхідно вказати його ідентифікатор). Так, наприклад, можна писати alert ('Привіт') замість window.alert ('Привіт'), або location замість window.location. Винятками з цього правила є виклики методів open() і close(), у яких потрібно вказувати ім'я вікна, з яким працюють (батьківське в першому випадку та дочірнє – у другому).

Методи об'єкта ***window***

Які операції можна виконувати з вікном? Відкрити (створити), закрити (видалити), покласти його поверх всіх інших відкритих вікон (передати фокус). Крім того, можна управляти властивостями вікна та властивостями підлеглих йому об'єктів. Слід зосередитись на простих і найбільш популярних методах управління вікнами.

Метод alert() дозволяє створити вікно попередження, яке має тільки кнопку "OK":

<A HREF="javascript:window.alert('Увага')"> Повторіть запит!**</A>**

Потрібно мати на увазі, що повідомлення виводяться системним шрифтом, отже, для отримання попереджень російською мовою потрібна локалізована версія ОС.

Метод confirm() дозволяє задати користувачеві питання, на яке можна відповісти позитивно (натиснувши кнопку "OK") або негативно (натиснувши кнопку "Скасування" або "Cancel" чи просто закривши вікно запиту). Відповідно до дій користувача метод confirm() повертає значення true або false.

Метод prompt() дозволяє прийняти від користувача рядок тексту.

prompt**(**"Рядок питання","Рядок відповіді за замовчуванням"**)**

Коли користувач введе свою відповідь (або залишить незмінним відповідь за замовчуванням) і натисне кнопку OK, метод prompt() поверне отриманий рядок у якості значення, яке може далі набути будь- яка змінна і яке можна в подальшому розбирати в JavaScript- програмі.

Метод open() призначений для створення нових вікон. У загальному випадку його синтаксис виглядає наступним чином:

myWin = window**.**open**(**"URL"**,** " ім'я_вікна "**,**

" параметр = значення, параметр = значення,... ", замінити**);**

Перший аргумент задає адресу сторінки, яка завантажується в нове вікно (можна залишити порожній рядок, тоді вікно залишиться порожнім). Другий аргумент задає ім'я вікна, яке можна буде використовувати в атрибуті TARGET контейнерів <A> і <FORM>. Як значення допустимі також зарезервовані імена _blank, _parent, _self, _top, сенс яких такий же, як у аналогічних значень атрибута TARGET. Якщо ім'я_вікна збігається з ім'ям вже існуючого вікна (або кадру), то нове вікно не створюється, а всі наступні маніпуляції зі змінною myWin застосовуватимуться до цього вікна (або фрейму).

Третій аргумент не містить пробілів у рядках і є списком параметрів і їх значень, поданих через кому. Вказівка на кожний з параметрів необов'язкова, однак значення за замовчуванням можуть залежати від браузера, тому завжди треба вказувати саме ті параметри, які очікуються.

Метод window.open() повертає посилання на знову відкрите вікно, тобто об'єкт класу Window. Його можна надати до змінної (що й було зроблено вище), з тим щоб надалі можна було управляти відкритим вікном (писати в ньому, читати з нього, передавати та прибирати фокус, закривати).

Метод close() дозволяє закрити вікно. Найчастіше виникає питання, яке з вікон, власне, слід закрити. Якщо необхідно закрити поточне, то:

window.close(); self.close();

Якщо вікно відкрите за допомогою методу window.open(), то зі скрипта, працюючого в новому вікні, послатися на вікно-батько можна за допомогою window.opener (тут window посилається на об'єкт нового, створеного вікна, оскільки воно використано в скрипті, працюючому в новому вікні). Тому, якщо необхідно закрити батьківське вікно, тобто вікно, з якого було відкрито поточне, то:

window**.**opener**.**close**();**

Якщо необхідно закрити вільне вікно, то спочатку потрібно отримати його ідентифікатор:

```
id=window.open();
...
id.close();
```

Як видно з останнього прикладу, закривають вікно не за ім’ям (значення атрибута TARGET не враховується), а використовують покажчик на об'єкт.

focus() и blur()

Метод **focus()** застосовується для передавання фокуса в вікно, з яким він використовувався. Передавання фокуса корисне як під час відкриття вікна, так і за його закритті, не кажучи вже про випадки, коли потрібно вибирати вікна. Як приклад можна навести наступне.

Відкрити вікно і, не закриваючи його, знову відкрити вікно з таким же ім'ям, але з іншим текстом. Нове вікно не з'явилося поверх основного вікна, бо фокус йому не був переданий. Тепер повторити відкриття вікна, але вже з передачею фокуса . Оскільки зміст нового вікна записується з вікна старого (батька), то в якості покажчика на об'єкт використовується значення змінної myWin. Щоб відвести фокус з певного вікна myWin, необхідно застосувати метод myWin.blur().

Наприклад, щоб відвести фокус з поточного вікна, де виконується скрипт, потрібно викликати window.blur(). Ефект буде такий, ніби користувач сам згорнув вікно натисканням кнопки в правому верхньому кутку вікна.

Метод **setTimeout()** використовується для створення нового потоку обчислень, виконання якого відкладається на час (в мілісекундах), вказаний іншим аргументом:

```
idt = setTimeout("JavaScript_код",Time);
```

Типове застосування цієї функції – організація періодичної зміни властивостей об'єктів. Наприклад, можна запустити годинник в поле форми.

Метод **clearTimeout()** дозволяє знищити потік, викликаний методом setTimeout(). Його застосування дозволяє більш ефективно розподіляти ресурси обчислювальної установки.

Події об'єкта ***window***

Доцільно зупинитися на подіях, пов'язаних з об'єктом window. Обробники цих подій зазвичай поміщають як атрибут контейнера <BODY>.

Load – подія відбувається в момент, коли завантаження документа в даному вікні повністю закінчене. Якщо поточним вікном є фрейм, то подія Load його об'єкта window відбувається, коли в даному фреймі завантаження документа закінчилось, незалежно від стану завантаження документів в інших фреймах. Використовувати обробник даної події можна, наприклад, наступним чином:

<BODY onLoad="alert('Документ повністю завантажений.');">

Unload

Unload – подія відбувається в момент вивантаження сторінки з вікна. Наприклад, коли користувач закриває вікно або переходить з цієї веб-сторінки на іншу в разі посилання або набору адреси в адресному рядку чи у випадку зміни адреси сторінки (властивості window.location) скриптом. Наприклад, у разі виходу користувача з нашої сторінки можна подбати про його зручність і закрити відкрите раніше нашим скриптом вікно:

<BODY onUnload="myWin.close();">

Error – подія відбувається у разі виникнення помилки в процесі завантаження сторінки. Якщо ця подія відбулася, можна, наприклад, вивести користувачу повідомлення за допомогою alert() або спробувати перезавантажити сторінку за допомогою window.location.reload().

Focus – подія відбувається в момент, коли вікну передається фокус. Наприклад, коли користувач "розкриває" згорнуте раніше вікно або (у Windows) вибирає це вікно браузера за допомогою Alt + Tab серед вікон інших додатків. Ця подія відбувається також при програмному передаванні фокусу даних вікна шляхом виклику методу window.**focus**(). Приклад використання:

<BODY onFocus="alert('Дякую, що повернулися!');">

Blur – подія, протилежна попередній, відбувається в момент, коли дане вікно втрачає фокус. Це може статися в результаті дій користувача або програмними засобами – викликом методу window.**blur**().

Resize – подія відбувається під час зміни розмірів вікна користувачем або сценарієм.

Об'єктна модель документа (DOM)

Більшість дій в JavaScript виконується з HTML-сторінкою. У JavaScript сторінка подана у вигляді об'єктної моделі DOM (Document Object Model).

Будь-які дії зі сторінкою вимагають виклику відповідного методу DOM.

Об'єкт document

Об'єкт document є найважливішою властивістю об'єкта window (тобто повністю до нього потрібно звертатися як window.document). Усі елементи HTML-розмітки, присутні на веб-сторінці, – текст, абзаци, гіперпосилання, малюнки, списки, таблиці, форми тощо – є властивостями об'єкта document. Можна сказати, що технологія DHTML (Dynamic HTML), тобто динамічна зміна вмісту веб-сторінки, полягає саме в роботі з властивостями, методами та подіями об'єкта document.

Уміння працювати з документом в моделі DOM є наріжним каменем у JavaScript-програмуванні.

Доступ до елементів

Будь-який доступ і зміни DOM беруть свій початок від об'єкта document.

Вершина дерева

document.documentElement

Самий верхній тег. У разі коректної HTML-сторінки, це буде <html>. document.body. Тег <body>, якщо є в документі (зобов'язаний бути).

Типи DOM-елементів

У кожного елемента в DOM-моделі є тип. Його номер зберігається в атрибуті elem.nodeType. Всього в DOM розрізняють 12-ть типів елементів. Зазвичай використовується тільки один: Node.ELEMENT_NODE, номер якого дорівнює 1. Елементам цього типу відповідають HTML-теги.

Іноді корисний ще тип Node.TEXT_NODE, який дорівнює 3. Це текстові елементи.

Решта типів в JavaScript-програмуванні не використовуються.

Дочірні елементи

З вершини дерева можна піти далі вниз. Для цього кожен DOM-вузол містить масив усіх нащадків, окремо – посилання на першого й останнього нащадка та ще ряд корисних властивостей.

Усі дочірні елементи, включаючи текстові, знаходяться в масиві **childNodes**. Властивості **firstChild** і **lastChild** показують на перший і останній дочірні елементи та дорівнюють *null*, якщо нащадків немає.

Властивість **parentNode** вказує на батька. Наприклад, для <body> таким елементом є <html>.

Властивості **previousSibling** і **nextSibling** вказують на лівого та правого братів вузла.

Властивості елементів

У DOM-елементів є маса властивостей. Зазвичай використовується максимум третина з них. Деякі з них можна читати та встановлювати, інші – тільки читати.

Є ще й третій варіант, що зустрічається в IE, коли встановлювати властивість можна тільки під час створення елемента.

Слід розглянути ще деякі (не всі) властивості елементів, корисні при роботі з DOM.

tagName

Атрибут є у елементів-тегів і містить ім'я тега в верхньому регістрі, призначений тільки для читання.

style

Зміна індивідуальних стилів елементів. В усіх HTML-елементів є властивість style. З її допомогою можна легко міняти стилі, які вказані для елемента в HTML-атрибуті style.

innerHTML

Цю властивість містить весь HTML-код усередині вузла, і його можна змінювати. Властивість innerHTML застосовується в основному для динамічної зміни змісту сторінки.

className

Ця властивість задає клас елементу. Вона повністю аналогічна html-атрибуту "class", наприклад:

elem.className = 'newclass'

Об'єкт Image

Найбільш видовищні ефекти у програмуванні на JavaScript досягаються під час роботи з графікою. При цьому в арсеналі програміста не так вже й багато інструментів: вбудовані в документ малюнки,

можливість генерації об'єкта **Image**, комбінування малюнків з гіпертекстовими посиланнями та таблицями. Проте кількість різноманітних ефектів, які досягаються цими нехитрими засобами, вражає.

Програмування графіки в JavaScript спирається на об'єкт **Image**, який характеризується наступними властивостями, методами та подіями Незважаючи на значну кількість властивостей, їх абсолютну більшість можна тільки читати, але не змінювати. Про це свідчить, перш за все, відсутність методів. Але дві властивості все ж можна змінювати: src і lowSrc. Цього виявляється достатньо для безлічі ефектів з малюнками.

Зміна малюнка

Змінити малюнок можна, тільки присвоївши властивості *src* вбудованого об'єкта *Image* нове значення. Вище було показано, як це можна зробити в простому випадку. Очевидно, що повільне перезавантаження малюнка з сервера не дозволяє реалізувати швидке перегортання.

Рішення полягає в розведенні за часом підкачки малюнка та його відображення. Для початку створюється зображення, до якого прив'язуються обробники подій onMouseOver і onMouseOut. Після наведення покажчика миші на кожен з малюнків він замінюється іншим (кольоровим), а при відведенні мишки малюнок знов стає чорно-білим

```
<IMG NAME=m0 src="images/mapb000.gif" border=0
onMouseOver="document.m0.src=color[0].src;"
onMouseOut="document.m0.src= mono[0].src;">
```

Мультиплікація

Природним продовженням ідеї заміщення значення атрибута SRC у контейнері IMG є мультиплікація, тобто послідовна зміна значення цього атрибута в часі. Для реалізації мультиплікації використовують метод setTimeout() об'єкта window.

Існує два способи запуску мультиплікації: по закінченні завантаження сторінки (**onLoad**) і при діях користувача (**onClick**, **onChange** тощо). Найбільш популярний – перший, а саме – використання **onLoad**() і **setTimeout**().

Обробник події **onLoad**

Подія Load наступає в момент закінчення завантаження документа браузером. Оброблювач даної події (onLoad) вказується в контейнері BODY:

<BODY onLoad="програма JavaScript">

Оптимізація відображення

Під час програмування графіки слід враховувати безліч чинників, які впливають на швидкість відображення сторінки та швидкість зміни графічних образів. При цьому звичайна дилема оптимізації програм – швидкість або розмір займаної пам'яті – вирішується тільки на користь збільшення швидкості. Розмір пам'яті у програмуванні на JavaScript на увагу не береться.

З усіх способів оптимізації відображення малюнків слід зупинитися тільки на декількох:

– оптимізація відображення під час завантаження;

– оптимізація відображення за рахунок попереднього завантаження;

– оптимізація відображення за рахунок нарізки зображення.

Якщо перші дві позиції належать як до відображення статичних малюнків, так і до мультиплікації, то третій пункт характерний головним чином для мультиплікації.

Контрольні запитання

1. Що таке DHTML?
2. Що таке DOM?
3. Назвіть основну структуру DOM, опишіть її основні об'єкти.
4. Які типи об'єктів існують в JavaScript?
5. Назвіть види використання ключового слова this.
6. Чим відрізняється спосіб зміни індивідуальних стилів елемента від способу доступу до чинних стилів елемента?
7. Як знайти елемент за його ідентифікатором?
8. Як сформувати колекцію елементів одного тегу?
9. Як сформувати колекцію елементів одного класу?

Список джерел інформації

1. Роман Мельник. Програмування веб-застосувань (фронт-енд та бек-енд).- Львівська політехніка, 2018.-248с.
2. Wagner G. Building Front-End Web Apps with Plain JavaScript.,2020.-333с.
3. Duckett Jon. HTML and CSS: Design and Build Websites, 2020. - 514 с.
4. Herron D. Node.js Web Development – 4 edition., 2018. – 492с.
5. Mike McGrath. CSS in Easy Steps, 2020.- 192с.
6. Бородкіна І. Л., Бородкін Г. О. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів.- Ліра-К, 2020.- 212с.
7. Ерік Фрімен, Елізабет Робсон Head First. Програмування на JavaScript. – Фабула., 2022.-450с.
8. Haverbeke M. Eloquent JavaScript 3rd edition., 2018.-436с.

Навчальне видання

Методичні вказівки
до лабораторних робіт з
дисципліни «Основи веб -розробки»
для студентів спеціальностей
121 «Інженерія програмного забезпечення»
та 122 «Комп'ютерні науки»

Укладач:
ЛІТВІНОВА Юлія Сергіївна

Відповідальний за випуск Копп А.М.
Роботу до видання рекомендував Гамаюн І.П.

В авторській редакції

План 2025 р., поз. 174

Підп. до друку 13.02.2025
Гарнітура Times New Roman.
Ум. друк. арк. 0,35.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Самостійне електронне видання