

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ
УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисципліни
«Web-дизайн»

Ухвалено
Вченою радою ФПМ
НТУУ «КПІ»
Протокол № 5 від 26.11.2011 р.

Київ
НТУУ «КПІ»

2012

Методичні вказівки до виконання лабораторних робіт з дисципліни «Web-дизайн» /
Уклад. А.В.Петрашенко, Д.С.Зам'ятін. – К.: ВПК «Політехніка», 2012. – 75 с.

Методичні вказівки призначені для студентів, що навчаються за спеціальностями
«Комп'ютерна інженерія» та «Програмна інженерія»

*Гриф надано Вченою радою ФПМ НТУУ «КПІ»
(Протокол №5 від 26.11.2011 р.)*

Навчально-методичне видання

**МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисципліни «Web-дизайн»**

Укладачі	<i>Петрашенко Андрій Васильович, канд. техн. наук, доц. Зам'ятін Денис Станіславович, канд. техн. наук, доц.</i>
Відповідальний за випуск	<i>Заболотня Тетяна Миколаївна, канд. техн. наук, ст.викл.</i>
Рецензент	<i>Жабін Валерій Іванович, д-р техн. наук, проф.</i>

Підп. до друку Формат 60×84¹/₁₆ . Папір офс. Гарнітура Times
Спосіб друку – ризографія. Ум. друк. арк. Обл.-вид. арк. Зам. №
Наклад 300 прим.

НТУУ «КПІ» ВПІ ВПК «Політехніка»
03056, Київ, вул. Політехнічна, 14, корп. 15

Зміст

ВСТУП	4
Порядок виконання лабораторних робіт	6
Особливості оформлення протоколу виконання лабораторних робіт	6
Лабораторна робота №1	7
Створення статичного Web-сайту засобами мов XHTML та CSS	7
Основні теоретичні відомості	8
Мова розмітки Web-документів XHTML	8
Мова CSS	15
Методичні вказівки	18
Етапи розробки статичного Web-сайту	18
Кодування сторінок сайту мовою XHTML та CSS	19
Варіанти завдання	30
Контрольні запитання	38
Рекомендована література	38
Лабораторна робота №2	39
Доступ до об'єктної моделі Web-документа засобами мови Javascript	39
Основні теоретичні відомості	39
XHTML DOM – об'єктна модель Web-документа	39
Особливості програмування мовою Javascript	41
Методичні вказівки	42
Оброблювачі подій мовою Javascript	42
Модифікація елементів XHTML DOM	45
Обробка форм XHTML	50
Варіанти завдання	53
Контрольні запитання	57
Рекомендована література	57
Лабораторна робота №3	58
Основні теоретичні відомості	58
Бібліотека JQuery як інструмент розробки інтерактивних Web-додатків	58
Методичні вказівки	65
Використання серверних Web-технологій	65
Застосування Ajax для асинхронного завантаження XML-документа	66
Застосування Ajax для асинхронного завантаження об'єкта Javascript	68
Функції \$.get(), \$.post() та \$(selector).load()	70
Варіанти завдання	72
Контрольні запитання	76
Рекомендована література	76

ВСТУП

Дисципліна «Web-дизайн» є важливою складовою підготовки кваліфікованих бакалаврів за освітніми напрямками 6.050102 «Комп'ютерна інженерія» та 6.050103 «Програмна інженерія». Web-дизайн сьогодні – це основа створення інтерфейсів користувача широкого спектру Web-орієнтованих додатків: Web-порталів, інформаційних, довідкових, пошукових, корпоративних, освітніх та інших типів сайтів та систем. Крім того, велика кількість програмного забезпечення, яке функціонує із використанням так званого віконного інтерфейсу користувача, поступово переводиться у Web-орієнтований спосіб взаємодії із користувачем для підвищення ефективності розробки та супроводження існуючого програмного забезпечення.

Дисципліна Web-дизайн включає такі важливі складові, як розмітка Web-документу, його зовнішнє оформлення, інтерактивність елементів управління сторінки, а також зручність експлуатації Web-сайтів в цілому. Розглядаючи технічний аспект Web-дизайну, слід відмітити, перш за все, мови розмітки документів – XHTML (*eXtensible HyperText Markup Language*) та XML (*eXtensible Markup Language*), які дозволяють вирішити задачу структуризації інформації на Web-сторінках у чіткий формалізований спосіб, зокрема, встановлення ієрархічних та інших видів зв'язків між такими елементами сторінки, як блоки навігації, заголовочні елементи, інформаційні фрагменти тощо. Використання підмножини XHTML універсальної мови розмітки XML дозволяє створювати сторінки для виведення на різних пристроях відображення інформації: моніторі персонального комп'ютера, екрані кишенькового комп'ютера, мобільного телефону тощо. Наступним технічним аспектом є мова стильового оформлення документів CSS (*Cascading Style Sheets*), яка призначена для надання єдиного візуального стилю усім сторінкам Web-сайту або Web-додатка. Саме мова CSS призначена для винесення функцій форматування текстових, табличних, графічних та інших видів даних з HTML-сторінки, інакше кажучи, відокремлення даних від опису їх виведення. За

динамічну (інтерактивну) складову інтерфейсу користувача Web-сторінки відповідає мова програмування Javascript, інтерпретатор якої є вбудованим у більшість сучасних Web-браузерів. Javascript дозволяє модифікувати вміст та зовнішній вигляд сторінки із можливістю завантаження нових даних з сервера, використовуючи традиційний програмний спосіб доступу до елементів Web-сторінки. Модель програмування Javascript базується на подіях, а також на використанні об'єктної моделі Web-документа.

Крім технічних задач, які постають перед розробниками, важливими є також питання естетичного сприйняття інформації, наприклад, композиції елементів, кольорової гами, єдності та контрасту елементів тощо, а також питання зручності експлуатації (*Usability*), в основі якої – дисципліна організації людино-машинних інтерфейсів. При цьому, критеріями зручності експлуатації є швидкість оволодіння новим інтерфейсом, продуктивність використання, запам'ятовуваність інтерфейсу, а також задоволеність користувачів даним інтерфейсом.

Відповідно до наведених аспектів Web-дизайну та згідно із робочою навчальною програмою дисципліни «Web-дизайн» завданням даних методичних вказівок є набуття студентами практичних навичок створення сучасних Web-орієнтованих інтерфейсів користувача з використанням загальноприйнятих стандартів, сучасного програмного інструментарію, технологій, мов розмітки та програмування.

Порядок виконання лабораторних робіт

1. Ознайомлення з метою, загальним завданням та теоретичними відомостями відповідної лабораторної роботи.
2. Визначення варіанту завдання та розробка плану виконання роботи.
3. Виконання завдання, використовуючи вказані у завданні мови, технології, інструментальне програмне забезпечення тощо.
4. Тестування окремих програмних модулів та програми в цілому.
5. Підготовка відповідей на контрольні запитання до роботи.
6. Підготовка протоколу виконання лабораторної роботи.
7. Демонстрація викладачеві результатів функціонування програми.

Особливості оформлення протоколу виконання лабораторних робіт

Протокол має містити наступні складові частини, які розміщують кожний з нової сторінки: титульний аркуш, загальне завдання до лабораторної роботи, завдання за варіантом, діаграма структури програми із призначенням кожного модуля, текст програмного коду (2-4 сторінки), екранні форми («screenshots») візуального інтерфейсу програмних модулів, відповіді на контрольні запитання.

ЛАБОРАТОРНА РОБОТА №1

Створення статичного Web-сайту засобами мов XHTML та CSS

Метою лабораторної роботи є оволодіння навичками створення статичних Web-сайтів у відповідності до загальноприйнятих стандартів.

Загальне завдання на лабораторну роботу:

1. Вивчити особливості розмітки Web-документів із використанням мови XHTML та CSS.
2. Розробити макет та виконати кодування сторінок статичного Web-сайту відповідно до варіанта.
3. Виконати інформаційне (графічне та текстове) наповнення сторінок Web-сайту.

Деталізоване завдання полягає у наступному:

1. Згідно з макетом головної сторінки Web-сайту виконати її розмітку, а саме, кодування мовами XHTML та CSS. Для розмітки забороняється використовувати таблиці, лише базові теги: <div>, .
2. Меню сайту створювати за допомогою списків (,) з відповідним форматуванням CSS.
3. Розмітку документу виконувати згідно із правилами створення XHTML-документів (із синтаксисом XML). Перевірити відповідність стандартам XML на сайті <http://validator.w3.org>.
4. Текстові та графічні фрагменти сторінок заповнювати обов'язково.
5. Заголовок сторінки оформлювати у вигляді зображення або текстової назви.
6. Нижній колонтитул має містити інформацію про студента та дату створення сайту.
7. З кожної сторінки має бути доступний перехід на головну сторінку.
8. Максимальна кількість кольорів та шрифтів на сайті – 4.
9. CSS-стили створити у окремому файлі, який підключається до усіх сторінок сайту.

Основні теоретичні відомості

Мова розмітки Web-документів XHTML

HTML – мова розмітки гіпертекстових документів (Web-документів, Web-сторінок) призначена для вирішення декількох задач: структурування інформації на Web-сторінці, логічного об'єднання Web-сторінок у єдиний сайт завдяки гіперпосиланням, а також форматування окремих елементів сторінки (тексту, зображень тощо). Слід зауважити, що вирішення останньої задачі на сьогоднішній день вирішується завдяки використанню спеціалізованої мови стильового оформлення Web-документів CSS. Хоча HTML містить засоби так званого «фізичного» форматування фрагментів документа (декорування тексту, вирівнювання елементів тощо), використовувати їх не рекомендується, оскільки це значно ускладнює процес модифікації дизайну сторінки, а також призводить до недоцільного дублювання HTML-коду. Особливу увагу слід звернути на те, що HTML – не є мовою програмування, ця мова не містить управляючих конструкцій типу циклів або умовних операторів. В той же час HTML містить спеціальні інструкції для включення в сторінку програм мовою Javascript, або інших об'єктів, які можуть бути виконані для даної сторінки.

Мова XHTML – це модифікація HTML для підтримки сумісності з правилами формування XML-документів. Ця модифікація дозволяє використовувати семантику мови HTML (призначення інструкцій залишається таким самим) та строгий та однозначний синтаксис XML, який гарантує відсутність синтаксичних помилок у вихідному коді сторінки, а також можливість відображення того ж інформаційного наповнення сторінки на різних пристроях, зокрема, персональних, кишенькових комп'ютерах або мобільних телефонах завдяки засобам автоматичної трансформації XML-документів.

Розглянемо синтаксис XHTML. Документ XHTML є текстовим файлом, який за своєю структурою складається з елементів, що визначають правила форматування його фрагментів. Кожний елемент записується як початкова

частина, вміст та кінцева частина (рис.1.1). Початкова та кінцева частини визначаються тегами. Тег – це запис виду `<назва>` для початкової частини та `</назва>` - для кінцевої, де *назва* відповідає призначенню елемента.

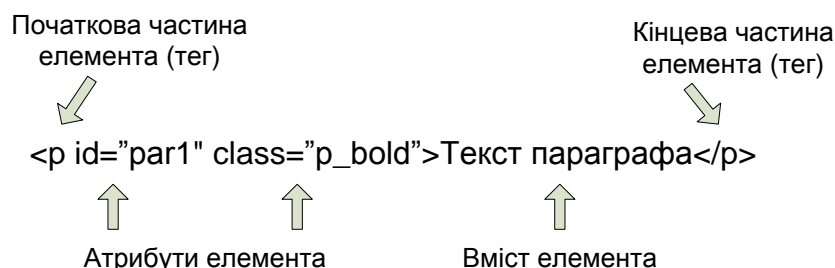


Рис.1.1. Структура елемента XHTML

З рис.1.1 видно, що початкова частина елемента може містити додаткові параметри – атрибути, які уточнюють деякі характеристики елемента в залежності від призначення елемента, наприклад, для запису

`Гіперпосилання`

атрибут *href* вказує на цільову адресу гіперпосилання.

Якщо вміст елемента відсутній, допустимим є запис виду: `<назва/>`.

Наприклад, `
`, `<p/>`, `<hr/>`.

Елементи сторінки об'єднуються у строгий ієрархічний порядок, причому кожний XHTML-документ повинен мати обов'язкову базову структуру (рис.1.2):

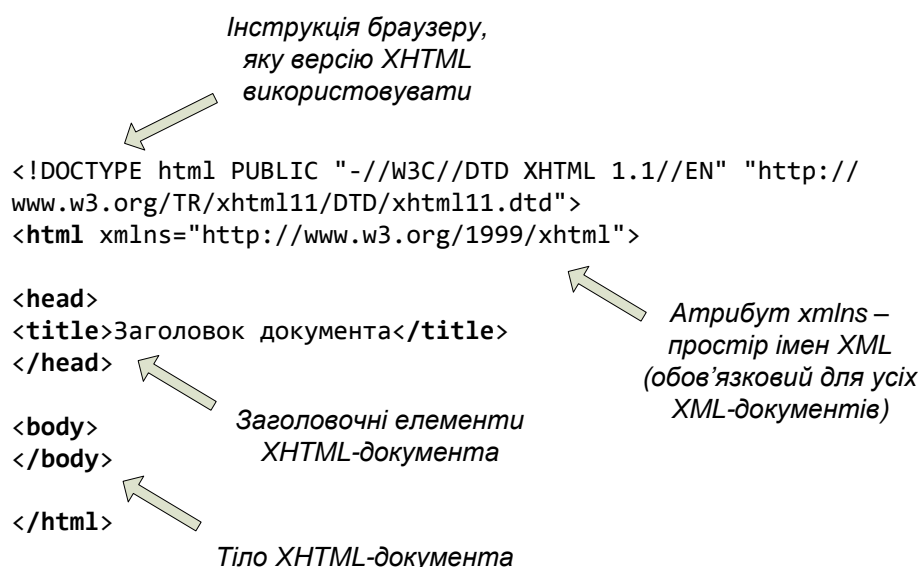


Рис.1.2. Мінімальний вміст XHTML-документа

Важливою особливістю базової структури XHTML-документа є наявність в ньому інструкції `<!DOCTYPE>`, яка вказує браузеру, який саме діалект мови використав автор сторінки і, відповідно, що саме вважати помилками на сторінці. На практиці використовують декілька варіацій HTML та XHTML. Переважно їх відмінності полягають у забороні використання фреймів та застарілих тегів «фізичного» форматування тексту (наприклад, ``, `<center>`). Проте усі діалекти XHTML мають задовольняти правилам побудови так званих коректно сформованих (well-formed) XML-документів. Найголовніші правила полягають у наступному:

1. Документ обов'язково повинен мати один або більше елементів.
2. В документі має бути один і лише один кореневий елемент. Решта елементів ієрархічно підпорядковуються йому.
3. Елементи мають бути ієрархічно коректно підпорядковані між собою.
4. Елементи мають бути записаними в нижньому регістрі.
5. Елементи мають включати початкову та кінцеву частину (рис.1.1).
6. Назви атрибутів елементів мають бути в нижньому регістрі.
7. Значення атрибутів мають бути записані у подвійних лапках.

В обов'язковій секції `<head>` документа, як правило, записують елементи посилань на зовнішні файли, зокрема, програм мовою Javascript:

```
<script type="text/javascript" src="myfile.js"></script>
```

або файлів стильового оформлення CSS:

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

Крім того, в даній секції можуть бути мета-теги, що описують сам документ, наприклад, заголовок, автор, ключові слова для пошукової системи тощо.

В обов'язковій секції `<body>` міститься тіло документа, власне те, що бачить користувач: текст документа, посилання на зовнішні графічні та інші види мультимедійних ресурсів, а також посилання на програмні модулі.

Розглянемо основні типи елементів HTML (табл.1.1).

Групи елементів XHTML

Група елементів	Опис	Приклади елементів
1	2	3
Базові	Блокові контейнери: тіла документа, абзаців, заголовків, а також коментарі.	<body>, <p>, <h1>.. <h6>, </h6>, , <hr/>, <div> <!--коментар -->
«Фізичне» форматування	Рядкові елементи декорування тексту: шрифту, кольору, інтервалів тощо.	<s>, , <i>, <sub>, <sup>, <u>, <q>, <strike>,
Елементи форм	Описують елементи введення даних від користувача (кнопки, поля уведення, перемикачі тощо)	<form>, <input>, <textarea>, <select>, <option>, <label>, <button>, <fieldset>
Фрейми	Елементи відображення декількох сторінок у одному вікні браузера	<frame>, <frameset>, <noframe>, <iframe>
Зображення	Вказують на зовнішні файли зображень (jpg, gif тощо)	, <map>, <area>
Посилання на зовнішні файли	Дозволяють додавати гіперпосилання, включати зовнішні програми мовою Javascript, файли стилів CSS та інші	<a>, <link>
Списки	Організують текст у вигляді списків (нумерованих або декорованих)	, ,
Таблиці	Структурують дані в табличну форму	<table>, <tr>, <td>, <th>

1	2	3
Стили CSS	Застосовують стилі для документа	<style>
Програмування	Застосування можливостей управління сторінкою, зокрема, підключення Flash	<script>, <noscript>, <object>, <param/>
Елементи секції <head>	Описують службову інформацію про сторінку: автор, ключові слова, назва, опис та інші	<meta>, <title>, <base>

Одиниці виміру, прийняті в XHTML

Для виміру розмірів, відступів, інтервалів тощо можуть бути застосовані абсолютні та відносні одиниці виміру (табл.1.2).

Таблиця 1.2

Одиниці виміру XHTML

Відносні	Абсолютні
<i>px</i> – піксель (залежить від пристрою виведення)	<i>in</i> – дюйми
<i>%</i> – відсоток від можливого розміру	<i>cm</i> – сантиметри
<i>em</i> – розмір відносно ширини стандартної літери «m»	<i>mm</i> – міліметри
<i>ex</i> – розмір відносно висоти стандартної літери «x»	<i>pt</i> – пункти (1/72 дюйма)
	<i>pc</i> – піка (12 пунктів)

Атрибути елементів XHTML

Атрибути визначають додаткові характеристики елементів в залежності від призначення того чи іншого елемента. Проте існує група атрибутів, які є допустимими майже до будь-яких з них (табл.1.3). Виключення складають елементи: `<base>`, `<head>`, `<html>`, `<meta>`, `<param>`, `<script>`, `<style>` та `<title>`.

Таблиця 1.3

Базові атрибути XHTML

Назва атрибута	Призначення	Приклади
id	Унікальний ідентифікатор елемента. Застосовується при визначенні стилю CSS або при здійсненні доступу через Javascript	<code><p id="p1">text</p></code>
style	Дозволяє визначити стиль CSS для поточного елемента	<code><h1 style="color:black"></code> чорний заголовок <code></h1></code>
title	Описує додаткову інформацію про елемент	<code><div title="Меню"></code> ... <code></div></code>
class	Задає клас CSS, до якого належить елемент	<code><div class="bold-text"></code> Текст <code></div></code>

Колір у XHTML задається декількома способами:

1. Англійські ідентифікатори для основних тонів (black, white, red, green, yellow тощо). Наприклад, `<body bgcolor="white"> ... </body>`

2. 16-кове значення компонент у кольоровому просторі RGB (червоний-зелений-синій), наприклад: *#ff0000* (червоний), *#0000ff* (синій), *#cccccc* (сірий).

Адресація ресурсів у XHTML

Для унікальної ідентифікації ресурсів прийнято використовувати URL-схему. Структура URL має наступний вигляд:

`<схема>://<логін>:<пароль>@<хост>:<порт>/<URL - шлях>?<параметр1>=<значення1>&<параметр2>=<значення2>...#<id елем.>`, де

`<схема>` – назва протоколу, зокрема `http`, `https`, `ftp` тощо (обов'язковий компонент);

`<логін>` – ім'я користувача;

`<пароль>` – пароль користувача;

`<хост>` – доменне ім'я сервера або його IP-адреса (обов'язковий компонент);

`<порт>` – TCP-порт;

`<URL - шлях>` – локальна адреса ресурсу на сервері відносно кореневого каталогу Web-сервера, заданого в конфігурації;

`<параметр1>=<значення1>` – додаткові параметри запита (може бути декілька);

`<id елем.>` – елемент сторінки, який буде активним на сторінці (вказується значення атрибута *id* елемента).

Деякі приклади URL-адресації:

`http://www.example.com/index2.html#p123`

`http://www.example.com/path/to/file/page1.php?a=1&b=abc&c=q1`

`http://10.111.123.231:8080/page/to/resource/`

Крім того, допустимим є використання так званих відносних шляхів URL:

`./index3.html`, де «./» означає, що ресурс знаходиться у поточному каталозі;

/file.php, де «/» означає, що *file.php* необхідно адресувати з кореневого каталогу сайту;

../img/il.jpg, де «../» означає, що для знаходження файлу *il.jpg* треба піднятися на один каталог вище, і потім шукати файл у каталозі *img*.

Відносно адресацію рекомендується використовувати для знаходження локальних ресурсів. Це спрощує подальшу реструктуризацію сайту.

Мова CSS

Призначенням CSS є застосування властивостей форматування елементів у єдиний стандартизований спосіб та відокремлення засобів структурування елементів (XHTML) від їх візуального подання. Стилі CSS можна задати декількома способами:

1. Як атрибут *style* елемента. В такому випадку властивості форматування будуть застосовані лише для даного елемента. Використання цього способу на практиці можна вважати виключенням, оскільки він майже тотожний фізичному форматуванню засобами HTML. Наприклад: `<p style="color:red">Червоний текст</p>`.

2. Як елемент `<style>` записаний, як правило, у секції `<head>`. Такий спосіб задає стилі, що будуть застосовані для всього даного документа. Цей спосіб характерний для форматування деякої окремої сторінки, яка стилізована інакше, ніж решта. Наприклад:

```
<head>
<style>
p {color:red; margin:5px}
#div1 {border: 1px red solid}
</style>
</head>
```

3. Використання директиви `@import` мови CSS. Цей спосіб дозволяє розділяти інструкції CSS на окремі файли (модулі), зокрема, відокремлюючи окремі секції сайту: заголовки, основний вміст, меню тощо. Наприклад:

`<style>`

`@import url("menu.css")`

`@import url("content.css")`

`</style>`

4. Як зовнішній файл CSS, що підключається завдяки елементу `<link>` для усіх сторінок сайту. Цей спосіб є найбільш вживаним, оскільки дозволяє формувати усі сторінки єдиним способом, скорочує обсяги HTML-сторінок та відокремлює структуру документа від засобів її відображення. Наприклад:

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

Пріоритетність застосування стилів у випадку наявності декількох способів віддається першому способу, інакше другому, інакше третьому, інакше четвертому. Якщо жодний зі стилів не визначено для елемента, підставляються налаштування за замовчуванням, прийняті в браузері.

Синтаксис CSS

Синтаксично інструкції CSS, які називають селекторами, записуються у вигляді правил, що застосовуються для одного, групи або усіх елементів сторінки. Приклад:

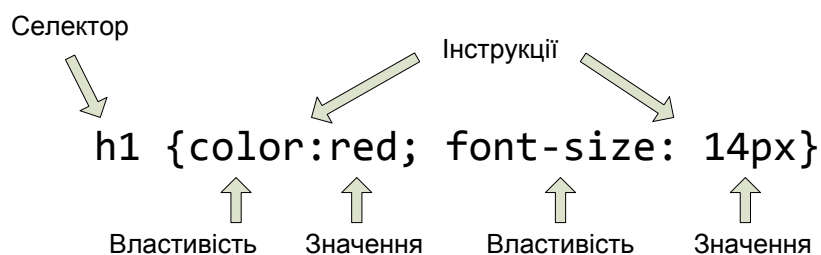


Рис. 1.3. Приклад запису правила CSS

Селектори

Основою CSS є селектори, які дозволяють визначати один елемент, групу елементів або усі елементи сторінки, для яких буде застосовано те чи інше правило форматування. Існують десятки правил, які дозволяють визначати цільові елементи форматування, причому допустимими є правила як CSS, так і

правила, що описані в стандарті мови XPath (спеціально розробленої для пошуку елементів у XML-документах). Наведемо приклади найбільш вживаних селекторів.

1. «*» застосувати для усіх елементів сторінки:

** {margin: 10px}*

2. «#X» застосувати для елемента з атрибутом *id* рівним X:

#myid {border: 0}

У XHTML: `<p id="myid">Text</p>`

3. «.X» застосувати для елементів, в яких атрибут *class* має значення X:

.myclass {top:15px}

У XHTML:

`<div class="myclass">Text1</div>`

`<p class="myclass">Text2</p>`

4. «X Y» застосувати для Y, який знаходиться ієрархічно всередині X (як безпосередньо під X, так і глибше за ієрархією):

ul a {text-decoration: none}

У XHTML:

` <a>Link `

5. «X» застосувати для усіх елементів, назва тегу яких є X.

p {font-weight:bold}

6. «X, Y, Z» застосувати для всіх перерахованих селекторів:

p, a, #myid, .myclass {border: 1px solid black}

7. «X:visited», «X:hover» та інші. Застосувати для селектора X зі специфічним станом: посилання, яке вже відвідане, курсор миші над елементом тощо. Запис після двокрапки називається *псевдокласом*.

a: visited {color: #ffffff}

8. «X+Y» застосувати для Y, який знаходиться на тому ж рівні ієрархії та розміщено безпосередньо за X.

h1+p {padding: 5px}

У XHTML:

`<h1>Z1</h1>`

`<p>P1</p>`

9. «X>Y» застосувати для Y, який знаходиться ієрархічно *безпосередньо* всередині X (порівняти з п. 4):

`ul a {text-decoration: none}`

У XHTML:

` <a>Link`

Наступні правила запозичені з мови XPath.

10. «X[Y]» застосувати для X, у якого наявний атрибут Y:

`h4[title] {color: 0}`

У XHTML:

`<h4 title="abcd">Header</h4>`

11. «X[Y="V"]» застосувати для X, у якого атрибут Y має значення V:

`a[href="example.com"] { text-decoration: none }`

12. «X[Y\$="V"]» застосувати для X, у якого значення атрибута Y закінчується на V:

`a[href$=".jpg"] { text-decoration: none }`

Методичні вказівки

Етапи розробки статичного Web-сайту

На практиці процес створення статичного Web-сайту спрощено можна розділити на такі етапи.

1. Розробка структури Web-сайту.

На даному етапі проводиться аналіз задач, для вирішення яких призначено Web-сайт, формується список сторінок, які складатимуть інформаційне наповнення сайту та відповідну систему навігації. В рамках лабораторної роботи список сторінок та структурні (ієрархічні) зв'язки між ними задано попередньо, але в реальній ситуації необхідно узгодити структуру

сайту із замовником в залежності від інформаційного наповнення, яке планується розмістити на Web-сайті.

2. Створення графічного дизайну сайту, що включає розробку макетів головної та підпорядкованих сторінок у вигляді прямокутних блоків, які відповідають функціональному призначенню секції сторінки (наприклад, меню, заголовок, інформаційний блок тощо). Крім цього, визначаються стилеві особливості сайту: кольорова схема, набір шрифтів, їх розміри для різних структурних елементів сторінки та інші елементи дизайну з урахуванням питань зручності експлуатації (usability). В лабораторній роботі ці параметри задаються варіантом.

3. Кодування сторінок сайту мовою XHTML та CSS. На цьому етапі власне і відбувається створення сторінок у вигляді XHTML та CSS файлів, які визначають візуальне подання сторінки. При виконанні лабораторної роботи необхідно дотримуватись загальноприйнятих стандартів: задовольняти правилам формування структурних елементів сторінок, слідкувати за коректним іменуванням елементів (атрибут *id*), використанням DOCTYPE та мета-тегів заголовку сторінки. Крім того, сторінки мають проходити валідацію W3C (<http://validator.w3.org>).

4. Наповнення сторінок сайту інформаційним вмістом. На даному етапі формується текстовий та мультимедійний вміст сторінки (графіка, відео тощо).

5. Тестування. На цьому етапі виконується тестування системи навігації, гіперпосилань, розмітки сторінки, інформаційного наповнення, а також зручності експлуатації.

Кодування сторінок сайту мовою XHTML та CSS

Згідно з наведеними вище етапами фактичним завданням на лабораторну роботу є виконання третього пункту, а також частково другого (для розробки структури підлеглих сторінок сайту). Тому розглянемо більш детально процес підготовки та власне кодування сторінок сайту, зупинившись на створенні

головної сторінки. Вхідною інформацією є макет сторінки, наприклад, як на рис.1.4:

Головне меню	
Заголовок	
Основний вміст	Бічна панель
Нижній колонтитул	

Рис.1.4. Макет сторінки

Будемо вважати, що сайт має фіксовані розміри у вікні браузера і його буде розміщено у центрі по горизонталі. Встановимо розміри кожного із структурних елементів сторінки.

1. Головне меню: ширина – 800 пікселів (далі px), висота – 50px.
2. Заголовок: ширина – 800px, висота – 150px.
3. Основний вміст: ширина – 500px, висота змінюється в залежності від наповнення.
4. Бічна панель: ширина – 300px, висота змінюється в залежності від наповнення.
5. Нижній колонтитул: ширина – 800px, висота – 70px.

Наступним кроком є створення шаблону XHTML-сторінки, який містить визначення документу DOCTYPE, параметри кодування, а також інші мета-теги, необхідні для застосування пошуковими системами (рис.1.5). Слід зауважити, що у заголовку шаблону присутній елемент <link>, який підключає зовнішній файл стилів *main.css*, з папки *css*. Цей файл містить налаштування форматування елементів сторінки, його необхідно використовувати для усіх сторінок сайту для підтримки єдиного стилю відображення (шрифти, кольори тощо).

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
    <title>Назва сторінки</title>
    <meta http-equiv="Content-Language" content="uk" />

    <meta name="description" content="Опис" />
    <meta name="keywords" content="Ключові слова" />

    <meta name="author" content="Студент" />

    <link type="text/css" media="all" rel="stylesheet" href="css/main.css" />
</head>

<body>

</body>
</html>

```

Рис.1.5. Шаблон XHTML-сторінки

Оскільки сторінки сайту мають фіксовану ширину (800px), доцільним є створення контейнера для решти елементів сторінки. Тому в середину тегу `<body>` внесемо наступний запис:

```

<body>
<div id="container">
Вміст контейнера
</div>
</body>

```

Контейнер реалізовано як елемент *div* (універсальний блок) з унікальним ідентифікатором *container*. У файл *main.css* внесемо запис виду:

```

#container {
    width: 800px;
    margin: auto;
    border: 1px solid black
}

```

Цим записом встановлено ширину контейнера 800 пікселів, границю блока, а також горизонтальне вирівнювання «по центру». Саме властивість *margin* (поле) зі значенням *auto* вказує, що поля треба встановлювати

автоматично з усіх боків – за замовчуванням це визначає центральне вирівнювання. При цьому висота буде обраховуватись в залежності від вмісту блока (рис.1.6).



Рис.1.6. Контейнер сторінки

З рис.1.6 видно, що верхній бік контейнера має відступ від вікна браузера, щоб його позбавитись необхідно встановити відступи для елементів *html* та *body* в 0:

```
html, body {  
    margin: 0;  
    padding: 0;  
}
```

Наступним кроком є ініціалізація основних блоків сторінки: головного меню, заголовку, основного вмісту, бічної панелі та нижнього колонтитула. Як правило, ці блоки записують тегами `<div>` у XHTML-сторінці всередині контейнера:

```
<body>  
<div id="container">  
    <div id="nav">Головне меню</div>  
    <div id="header">Заголовок</div>  
    <div id="sidebar">Бічна панель</div>  
    <div id="content">Вміст</div>  
    <div id="footer">Нижній колонтитул</div>  
</div>  
</body>
```

Додамо базові стилі для цих блоків у файл `main.css`:

```
#container {  
    width: 800px;  
    margin: auto;  
}
```

```
#nav {  
    height: 50px;  
}  
#header {  
    height: 150px;  
}  
#sidebar {  
    float: right;  
    width: 300px;  
}  
#footer {  
    height: 70px;  
}  
div {  
    border: 1px solid black;  
}
```

Висоту меню та нижнього колонтитула встановлено в абсолютних одиницях. Для зручності сприйняття усі елементи `<div>` виводяться з рамкою (останній запис). Особливим блоком є *sidebar*, оскільки за завданням його треба розмістити праворуч від основного вмісту. Крім того, оскільки *sidebar* – це додаткова панель, текст основного вмісту має «обтікати» його збоку. Це досягається шляхом використання властивості *float: right*. Результат ілюструє рис.1.7.

Головне меню	
Заголовок	
Вміст	Бічна панель
Нижній колонтитул	

Рис.1.7. Результат форматування базових блоків сторінки

У випадку, коли бічна панель буде за висотою більшою, ніж основний вміст, спостерігатиметься ефект накладання тексту бічної панелі на нижній колонтитул. Щоб позбавитись цього, необхідно відмінити «обтікання» для нижнього колонтитула в файлі CSS:

```
#footer {
    height: 70px;
    clear: both;
}
```

Наступним кроком є створення головного меню. Для прикладу оберемо популярні в Web пункти: «Про автора», «Послуги», «Контакти». Для меню будемо використовувати список HTML:

```
<div id="nav">
    <ul>
        <li><a href="#">Про автора</a></li>
        <li><a href="#">Послуги</a></li>
        <li><a href="#">Контакти</a></li>
    </ul>
</div>
```

Застосуємо наступні правила CSS для меню:

```
#nav ul {
```



```
margin: 0 auto;
padding: 2px 0 0 0;
border-top: 1px dotted #AFAFAF;
border-bottom: 1px dotted #AFAFAF;
text-align: center;
width: 100%;
}
```

У наведеному блоці CSS встановлюються правила для списку (*ul*) всередині контейнера *#nav*: центральне вирівнювання, відступ зліва на 2px, верхню та нижню границю шириною 1px, сірим кольором пунктирним стилем. Ширина списку – 100%.

Визначимо стиль елементів списку *li*:

```
ul li
{
    list-style-type:none;
    margin:0;
    padding:3px 0;
    display:inline;
}
```

В цьому блоці CSS: маркер – відсутній, відступ – 0, поле зліва – 3px, тип елемента – рядковий (щоб розмістити список у вигляді рядка).

```
#nav ul li a
{
    padding:0 20px;
    line-height:40px;
    color:#afafaf;
    text-transform:uppercase;
    font-weight:bold;
    text-decoration: none;
}
```

Для гіперпосилань меню встановлено відступ з правого та лівого боку 20px, колір – сірий, усі символи – у верхньому регістрі, напівжирні.

При наведенні та натисканні лівої кнопки миші колір тексту пункту меню буде замінено на темно-сірий:

```
#nav ul li a:hover, #nav ul li a.active  
{  
    color: #333;  
}
```

Вміст заголовку сторінки складається з наступного коду:

```
<div id="header">Привітання <br/> Головна сторінка</div>
```

Стилеве оформлення CSS при цьому матиме вигляд:

```
#header {  
    text-align: center;  
    font-size: 3em;  
    color: #AFAFAF;  
    line-height: 1.1em;  
    letter-spacing: -0.04em;  
    background: white;  
}
```

Особливістю наведеного блоку CSS є збільшений втричі шрифт, на 4% зменшена відстань між символами, та збільшена на 10% висота рядка. Ілюструє вигляд головного меню та заголовку рис. 1.8:

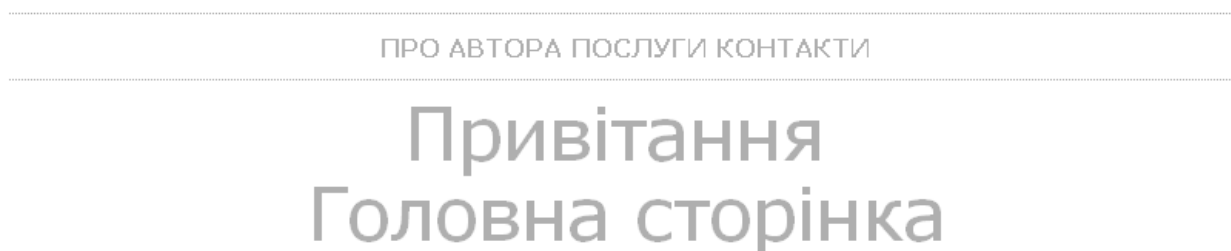


Рис. 1.8. Зовнішній вигляд меню та заголовку сторінки

Наступний елемент сторінки – бічна панель, стилеве оформлення якої наведено вище. Вміст XHTML виглядатиме наступним чином:

```
<div id="sidebar">
```

```
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc lacinia, odio in viverra  
varius, nisl quam laoreet leo, eu pretium nibh sapien eu diam. Etiam interdum lorem eget turpis vehicula euismod.  
Phasellus vestibulum placerat risus fermentum mollis. Fusce eget pretium eros. Aenean sed laoreet magna. Vestibulum  
ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;
```

```
    </p>
```

```
</div>
```

Після бічної панелі розміщено основний вміст. Для прикладу візьмемо 2 інформаційні повідомлення. XHTML-код блоку основного вмісту наведено нижче:

```
<div id="content">
```

```
    <ul >
```

```
        <li>
```

```
            <a href="#" class="info">
```

```
                <h2>Інформаційне повідомлення №1</h2>
```

```
                <p>Lorem Ipsum is simply dummy text of the printing and typesetting  
industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer  
took a galley of type and scrambled it to make a type specimen book.
```

```
            </p>
```

```
        </a>
```

```
    </li>
```

```
    <li>
```

```
        <a href="#" class="info">
```

```
            <h2>Інформаційне повідомлення №2</h2>
```

```
            <p>Lorem Ipsum is simply dummy text of the printing and typesetting  
industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer  
took a galley of type and scrambled it to make a type specimen book.
```

```
        </p>
```

```
    </a>
```

```
    </li>
```

```
</ul>
```

```
</div>
```

Контейнер *content* містить список, елементи якого складаються з гіперпосилання, заголовку та тексту повідомлення.

Відповідні стилі вмісту наведено нижче:

a.info

{

background: url('../images/info.gif') no-repeat left center;

```
}
```

Даним правилом встановлюємо фон гіперпосилання.

```
h2 {  
font-size: 1.4em;  
font-weight: bold;  
color: #333;  
margin: 10px;  
}
```

Зовнішній вигляд заголовку *h2* включає налаштування шрифту, кольору та відступу, які розглядались вище.

Наступним правилом знімаємо підкреслення з гіперпосилання, перетворюємо його на блок, дещо збільшуємо розмір шрифту та робимо колір світлішим (в чорно-білій гамі).

```
#content LI a  
{  
text-decoration: none;  
color:#333;  
padding:10px 0 10px 40px;  
display:block;  
color:#afafaf;  
font-size:103%;  
border-bottom: 1px dotted #CCC  
}
```

Зовнішній вигляд бічної панелі та основного вмісту наведено на рис.1.9.

Інформаційне повідомлення №1



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc lacinia, odio in viverra varius, nisl quam laoreet leo, eu pretium nibh sapien eu diam. Etiam interdum lorem eget turpis vehicula euismod. Phasellus vestibulum placerat risus fermentum mollis. Fusce eget pretium eros. Aenean sed laoreet magna. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Інформаційне повідомлення №2



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

Рис.1.9. Зовнішній вигляд блоків основного вмісту та бічної панелі

Останній елемент сторінки – нижній колонтитул. Як правило, його фон виділяють іншим кольором, а з інформаційної точки зору він містить інформацію про автора сторінки та, за необхідності, дублює деякі частини головного меню:

```
<div id="footer">
  <div id="altnav">
    <a href="#">Про автора</a> -
    <a href="#">Послуги</a> -
    <a href="#">Контакти</a>
  </div>
</div>
```

Стилеве оформлення нижнього колонтитула:

#footer a

{

color:#B86443;

margin-top:25px;

```
display: inline-block;  
font-size: 90%;  
}
```

При цьому гіперпосилання перетворено на рядковий елемент із властивостями блоку (*inline-block*), а шрифт зменшено на 10%.

Таким чином, сторінка у цілому виглядатиме як на рис.1.20:

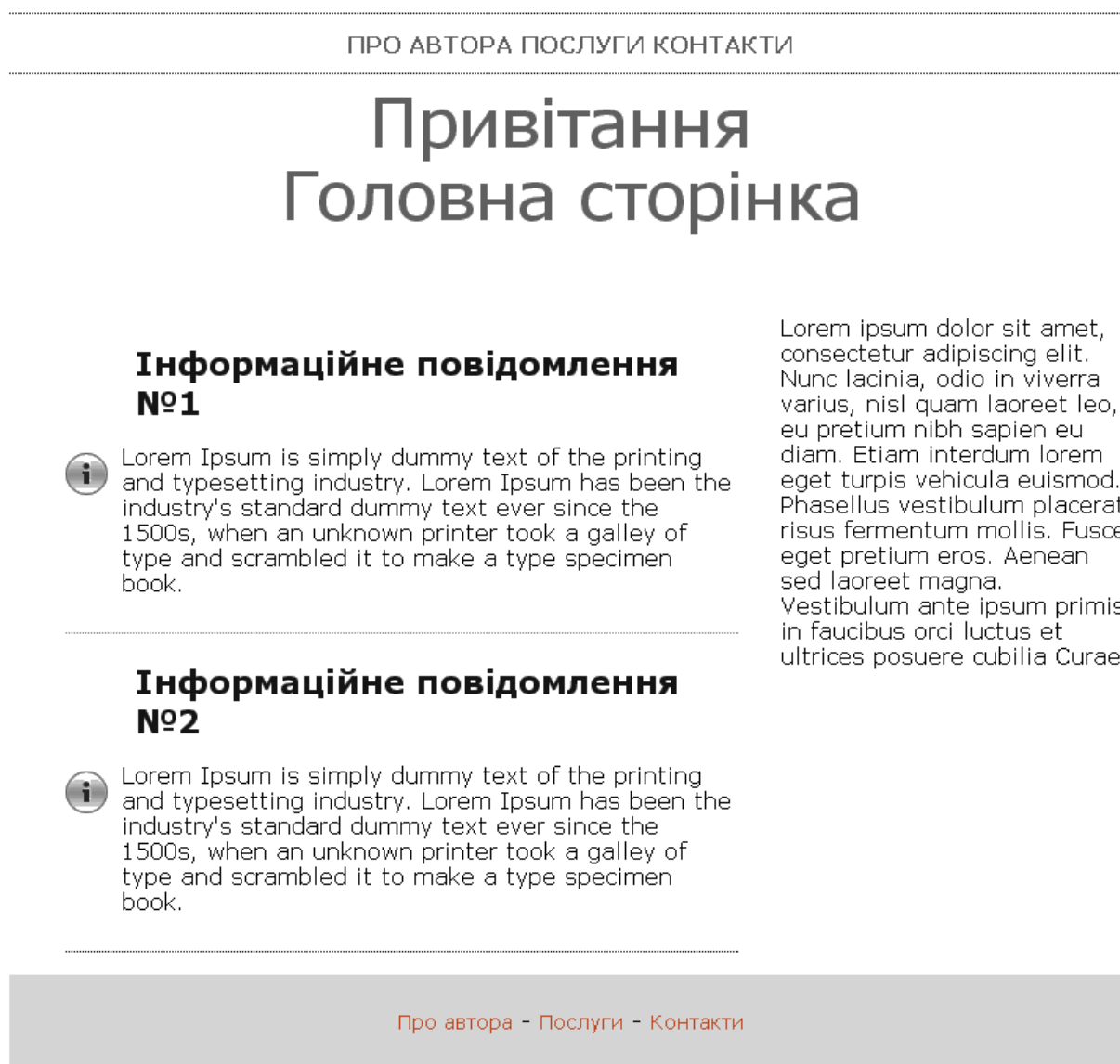


Рис.1.20. Результат розмітки сторінки

Варіанти завдання

Роботу виконувати самостійно. Номер варіанта завдання визначити, взявши останні дві цифри номера залікової книжки студента (табл.1.1).

Таблиця 1.1

Варіанти завдань

№	Макет головної сторінки (Заголовок – місце логотипу або назви, Меню – навігація по сайту, Вміст – вміст сторінки, Додаткова секція – рекламний блок, Нижній колонтитул – контактна інформація)	Призначення сайту та набір сторінок (склад меню)	Фіксована чи плаваюча ширина сторінки								
1	2	3	4								
1.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Портфоліо автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	фіксована
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
2.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул		Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	фіксована		
Заголовок	Меню										
Вміст	Додаткова секція										
Нижній колонтитул											
3.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Додаткова секція</td><td>Вміст</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Додаткова секція	Вміст	Нижній колонтитул		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована		
Заголовок	Меню										
Додаткова секція	Вміст										
Нижній колонтитул											
4.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул	Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	плаваюча			
Заголовок	Меню										
Вміст	Додаткова секція										
	Нижній колонтитул										

Продовження табл.1.1

1	2	3	4									
5.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Сайт про програмний продукт: 1. Опис програмної розробки 2. Сторінка завантаження із зображеннями - прикладами роботи 3. Контакти автора 4. Сторінка зворотного зв'язку	фіксована	
Заголовок												
Меню												
Вміст	Додаткова секція											
Нижній колонтитул												
6.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	плаваюча		
Заголовок												
Меню												
Вміст	Додаткова секція											
	Нижній колонтитул											
7.	<table><tr><td colspan="3">Заголовок</td></tr><tr><td rowspan="2">Меню</td><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок			Меню	Вміст	Додаткова секція	Нижній колонтитул	Сайт про котеджне містечко: 1. Про містечко 2. Фотоальбом будівництва 3. Розташування на карті 4. Соціальні програми	плаваюча		
Заголовок												
Меню	Вміст	Додаткова секція										
		Нижній колонтитул										
8.	<table><tr><td colspan="2">Заголовок</td><td rowspan="4">Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню	Вміст		Додаткова секція		Нижній колонтитул		Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	фіксована
Заголовок		Меню										
Вміст												
Додаткова секція												
Нижній колонтитул												

9.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	плаваюча
	Меню	Заголовок							
	Вміст								
	Додаткова секція	Нижній колонтитул							

Продовження табл.1.1

1	2	3	4								
10.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована			
Меню	Заголовок										
	Вміст										
	Додаткова секція										
	Нижній колонтитул										
11.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Портфоліо автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	плаваюча
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
12.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td>Додаткова секція</td><td>Вміст</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Додаткова секція	Вміст	Нижній колонтитул		Сайт-«візитка» компанії: 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення	плаваюча		
Заголовок	Меню										
Додаткова секція	Вміст										
Нижній колонтитул											
13.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Портфоліо автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	фіксована		
Меню	Заголовок										
Вміст											
Додаткова секція	Нижній колонтитул										

14.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>		Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Персональна сторінка студента: <div><div>1. Біографія</div><div>2. Інформація про дисципліни в університеті</div><div>3. Розклад занять</div><div>4. Досягнення</div></div>	плаваюча
	Заголовок											
	Меню											
	Вміст	Додаткова секція										
Нижній колонтитул												

Продовження табл.1.1

1	2	3	4								
15.	<table><tr><td colspan="3">Заголовок</td></tr><tr><td rowspan="2">Меню</td><td rowspan="2">Вміст</td><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок			Меню	Вміст	Додаткова секція	Нижній колонтитул	Сайт про котеджне містечко: 1. Про містечко 2. Фотоальбом будівництва 3. Розташування на карті 4. Соціальні програми	плаваюча	
Заголовок											
Меню	Вміст	Додаткова секція									
		Нижній колонтитул									
16.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	фіксована			
Меню	Заголовок										
	Вміст										
	Додаткова секція										
	Нижній колонтитул										
17.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Персональна сторінка студента: 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення	фіксована
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											

18.	<table><tr><td>Заголовок</td><td rowspan="4">Меню</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст	Додаткова секція	Нижній колонтитул	Сайт про котеждне містечко: 1. Про містечко 2. Фотоальбом будівництва 3. Розташування на карті 4. Соціальні програми	плаваюча			
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											
19.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Портфолію автора: 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання	плаваюча
Заголовок											
Меню											
Вміст	Додаткова секція										
Нижній колонтитул											

Продовження табл.1.1

1	2	3	4						
20.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	фіксована
Меню	Заголовок								
Вміст									
Додаткова секція	Нижній колонтитул								
21.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Інформаційний сайт про культурну подію: 1. Інформація про концерт 2. Новини культурного життя 3. Сторінка із розташуванням місця проведення 4. Інформація про виконавця	фіксована	
Меню	Заголовок								
	Вміст								
	Додаткова секція								
	Нижній колонтитул								

22.	<table><tr><td colspan="2">Заголовок</td><td>Меню</td></tr><tr><td>Додаткова секція</td><td colspan="2">Вміст</td></tr><tr><td colspan="3">Нижній колонтитул</td></tr></table>	Заголовок		Меню	Додаткова секція	Вміст		Нижній колонтитул			Персональна сторінка викладача: <div>1. Біографія</div> <div>2. Інформація про дисципліни</div> <div>3. Розклад занять викладача</div> <div>4. Список наукових публікацій</div>	плаваюча
Заголовок		Меню										
Додаткова секція	Вміст											
Нижній колонтитул												
23.	<table><tr><td colspan="2">Заголовок</td><td rowspan="4">Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню	Вміст		Додаткова секція		Нижній колонтитул		Персональна сторінка студента: <div>1. Біографія</div> <div>2. Інформація про дисципліни в університеті</div> <div>3. Розклад занять</div> <div>4. Досягнення</div>	плаваюча
Заголовок		Меню										
Вміст												
Додаткова секція												
Нижній колонтитул												

Продовження табл.1.1

1	2	3	4								
24.	<table><tr><td rowspan="4">Меню</td><td>Заголовок</td></tr><tr><td>Вміст</td></tr><tr><td>Додаткова секція</td></tr><tr><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст	Додаткова секція	Нижній колонтитул	Сайт про програмний продукт: <ul style="list-style-type: none">1. Опис програмної розробки2. Сторінка завантаження із зображеннями - прикладами роботи3. Контакти автора4. Сторінка зворотного зв'язку	фіксована			
Меню	Заголовок										
	Вміст										
	Додаткова секція										
	Нижній колонтитул										
25.	<table><tr><td colspan="2">Заголовок</td></tr><tr><td colspan="2">Меню</td></tr><tr><td>Вміст</td><td>Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок		Меню		Вміст	Додаткова секція	Нижній колонтитул		Персональна сторінка студента: <ul style="list-style-type: none">1. Біографія2. Інформація про дисципліни в університеті3. Розклад занять4. Досягнення	фіксована
Заголовок											
Меню											
Вміст	Додаткова секція										
Нижній колонтитул											

26.	<table> <tr> <td rowspan="4">Меню</td> <td colspan="2">Заголовок</td> </tr> <tr> <td colspan="2">Вміст</td> </tr> <tr> <td colspan="2">Додаткова секція</td> </tr> <tr> <td colspan="2">Нижній колонтитул</td> </tr> </table>	Меню	Заголовок		Вміст		Додаткова секція		Нижній колонтитул		Персональна сторінка студента: <ol style="list-style-type: none"> 1. Біографія 2. Інформація про дисципліни в університеті 3. Розклад занять 4. Досягнення 	плаваюча
Меню	Заголовок											
	Вміст											
	Додаткова секція											
	Нижній колонтитул											
27.	<table> <tr> <td colspan="3">Заголовок</td> </tr> <tr> <td rowspan="2">Меню</td> <td rowspan="2">Вміст</td> <td>Додаткова секція</td> </tr> <tr> <td>Нижній колонтитул</td> </tr> </table>	Заголовок			Меню	Вміст	Додаткова секція	Нижній колонтитул	Портфоліо автора: <ol style="list-style-type: none"> 1. Про автора 2. Контакти 3. Основні роботи 4. Корисні посилання 	Фіксована		
Заголовок												
Меню	Вміст	Додаткова секція										
		Нижній колонтитул										
28.	<table> <tr> <td colspan="2">Заголовок</td> <td>Меню</td> </tr> <tr> <td>Додаткова секція</td> <td colspan="2">Вміст</td> </tr> <tr> <td colspan="3">Нижній колонтитул</td> </tr> </table>	Заголовок		Меню	Додаткова секція	Вміст		Нижній колонтитул			Сайт-«візитка» компанії: <ol style="list-style-type: none"> 1. Про компанію 2. Види продукції 3. Контакти 4. Історія створення 	плаваюча
Заголовок		Меню										
Додаткова секція	Вміст											
Нижній колонтитул												

Продовження табл.1.1

1	2	3	4								
29.	<table><tr><td>Меню</td><td>Заголовок</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td>Додаткова секція</td><td>Нижній колонтитул</td></tr></table>	Меню	Заголовок	Вміст		Додаткова секція	Нижній колонтитул	Персональна сторінка викладача: 1. Біографія 2. Інформація про дисципліни 3. Розклад занять викладача 4. Список наукових публікацій	фіксована		
Меню	Заголовок										
Вміст											
Додаткова секція	Нижній колонтитул										
30.	<table><tr><td>Заголовок</td><td>Меню</td></tr><tr><td colspan="2">Вміст</td></tr><tr><td colspan="2">Додаткова секція</td></tr><tr><td colspan="2">Нижній колонтитул</td></tr></table>	Заголовок	Меню	Вміст		Додаткова секція		Нижній колонтитул		Сайт про програмний продукт: 1. Опис програмної розробки 2. Сторінка завантаження із зображеннями - прикладами роботи 3. Контакти автора 4. Сторінка зворотного зв'язку	плаваюча
Заголовок	Меню										
Вміст											
Додаткова секція											
Нижній колонтитул											

Контрольні запитання

1. Визначення елемента XHTML та його атрибуту.
2. Правила формування коректно сформованого XML-документа.
3. Визначення DOM-моделі документа.
4. Недоліки використання таблиць XHTML для створення розмітки сторінки.
5. Універсальні атрибути XHTML.
6. Поняття блокової моделі CSS.
7. Основні селектори CSS.

Рекомендована література

1. Пауэлл, Т. Web-дизайн. Наиболее полное руководство [Текст] / Т. Пауэлл.
— СПб. : БХВ-Петербург, 2002. — 1024 с. — ISBN 0-07-212297-8.
2. Ши, Д. Философия CSS-дизайна [Текст] / Дейв Ши, Молли Е. Хольцшлаг.
— М. : НТ Пресс, 2005. — 312 с. — ISBN 5-477-00122-4.

ЛАБОРАТОРНА РОБОТА №2

Доступ до об'єктної моделі Web-документа засобами мови Javascript

Метою лабораторної роботи є оволодіння навичками маніпулювання інформаційним вмістом Web-документа засобами мови Javascript.

Завдання на лабораторну роботу:

1. Ознайомитись із основами XHTML DOM та синтаксисом мови Javascript.
2. Набути практичних навичок програмування мовою Javascript.
3. Розробити програму маніпулювання вмістом XHTML-сторінки за вказаним варіантом.

Основні теоретичні відомості

XHTML DOM – об'єктна модель Web-документа

XHTML DOM – це стандартизований підхід до пошуку та маніпулювання вузлами та елементами Web-документа. DOM дозволяє виконувати такі задачі, як пошук вузлів за визначеним критерієм, додавати, вилучати та замінювати вузли Web-документа, звертатись до атрибутів та змінювати вміст елементів XHTML, маніпулювати стилями CSS, а також під'єднувати оброблювачі подій. Практично, в браузерях XHTML DOM реалізується у вигляді об'єктно-орієнтованого програмного інтерфейсу (API), де об'єктами є вузли Web-документа, властивостями – їх атрибути, а методи визначають логіку обробки даних вузлів.

Вузли у XHTML DOM розділені на наступні типи:

1. Увесь документ – це вузол-документ.
2. Кожний елемент XHTML – вузол-елемент.
3. Текстовий вміст елемента XHTML – це текстовий вузол.
4. Атрибут елемента XHTML – це вузол-атрибут.
5. Коментар – це вузол-коментар.

Відповідно до типів вузлів надається визначений список властивостей та методів щодо їх обробки. Особливим вузлом є вузол-елемент, стандартні властивості та методи якого варіюються в залежності від його призначення. Наприклад, елементи XHTML-форми мають такі додаткові властивості, як *value*, *selectedIndex* тощо. Як відомо, XHTML-документ являє собою ієрархічну структуру, яка дозволяє виконувати доступ до всіх вузлів Web-документа. Наведемо приклад:

```
<html>
<head>
  <title>Заголовок</title>
</head>
<body>
  <h1>Привітання</h1>
  <p id="p1">Текстовий <b>Напівжирний</b>фрагмент</p>
</body>
</html>
```

Тоді, подання цього документа у вигляді дерева буде наступним (рис.2.1).

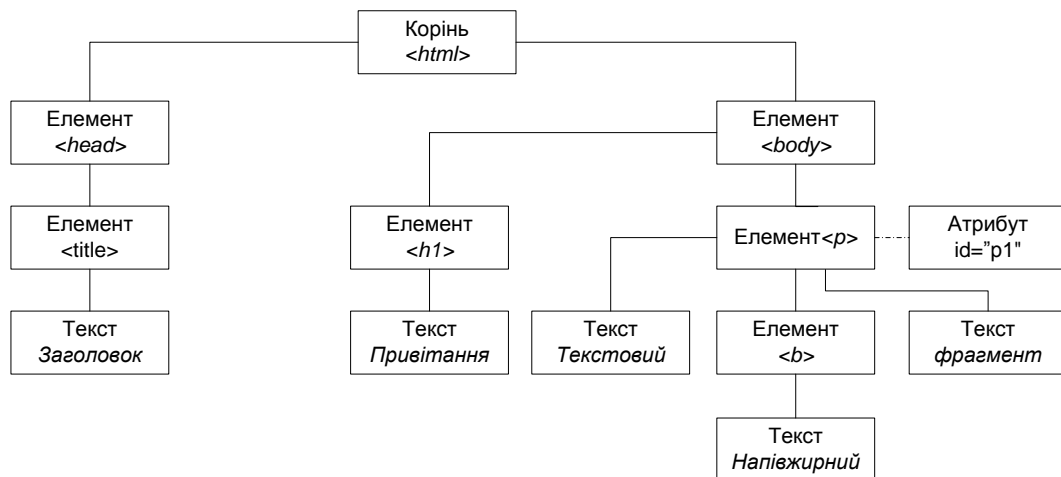


Рис.2.1. Подання Web-документа у вигляді XHTML DOM

Виходячи із деревовидної моделі, доступними є такі дії, як отримання атрибута (*attributes*) елемента, його батьківського елемента (*parentNode*), списку дочірніх елементів (*childNodes*), попереднього того ж рівня (*previousSibling*), наступного того ж рівня (*nextSibling*) та інші.

Серед методів відбору елементів документа можна виділити наступні:

1. *getElementById* – отримати елемент за його атрибутом *id*.
2. *getElementsByName* – отримати елементи за назвою тега.
3. *getElementsByTagName* – отримати елементи за їх атрибутом *name*.

Для маніпуляції елементами документа використовують наступні методи:

1. *appendChild* – додати елемент у кінець списку.
2. *removeChild* – вилучити елемент.
3. *replaceChild* – замінити елемент.
4. *getAttribute/setAttribute* – отримати/встановити значення атрибута та інші.

Особливості програмування мовою Javascript

Мова Javascript – це інструмент маніпулювання вмістом XHTML-документа. Ця мова має об'єктно-орієнтовані можливості, реалізована у вигляді інтерпретатора, вбудована у більшість сучасних Web-браузерів. Це означає, що немає необхідності у використанні інших засобів для виконання програм, крім браузера.

Javascript має декілька важливих аспектів, які характеризують її як спеціалізовану мову для програмування Web-документів:

1. Мовний аспект. Javascript має спрощений синтаксис, тому її відносять до так званих «скриптових» мов. Програми мовою Javascript можуть бути розміщені як всередині XHTML-документа, так і в окремому файлі. Синтаксис операторів Javascript схожий на синтаксис мов сімейства C, типи даних орієнтовані на специфіку Web-документа: рядки, масиви, об'єкти та інші. Більше того, усі типи даних реалізовані як вбудовані об'єкти із можливістю доступу до їх властивостей та методів маніпулювання даними цих типів. Прикладами вбудованих типів є: *Boolean*, *String*, *Array*, *Object*, *Number*.

2. Аспект DOM. Javascript надає програмний інтерфейс доступу до усіх складових Web-сторінки: вузлів, елементів, атрибутів тощо. Цей інтерфейс реалізовано у вигляді вбудованих об'єктів таких, як *Document*, *Event*,

HTMLElement та об'єктів, специфічних для тегів XHTML: *Form*, *Input*, *Body*, *Button*, *Frame*, *Image*, *Anchor* тощо. Таким чином, реалізація DOM дозволяє працювати з вмістом Web-документа з одного боку як з XML-документом, з іншого – як з HTML-документом, враховуючи специфіку тегів та їх атрибутів.

3. Аспект браузера. Оскільки Web-документ відображається та оброблюється безпосередньо браузером, мова Javascript має вбудовані об'єкти для роботи зі специфічними для браузера компонентами, зокрема *Window* (вікно документа в цілому), *Navigator* (інформація про поточний браузер), *Screen* (інформація про екран користувача), *History* (дані про вже відвідані сторінки) та *Location* (інформація про поточну адресу URL сторінки). Ці об'єкти дозволяють більш ефективно пристосувати відображення сторінки для конкретного браузера користувача.

Методичні вказівки

Оброблювачі подій мовою Javascript

Більшість дій, які виконує програма мовою Javascript, реалізуються у вигляді реакції на визначені типи подій від користувача або браузера. Виділяють такі типи подій:

1. Події DOM. Такі події спрацьовують при натисканні, підведенні/відведенні миші над визначеним елементом, натисканні клавіш клавіатури над елементом, який знаходиться в фокусі, власне при отриманні/втраті фокусу, зміні вмісту елементів форм XHTML тощо.

2. Події завантаження. Виникають при закінченні завантаження сторінки, фрейму або зображення, а також при закритті сторінки.

3. Події вікна та інші події. До цієї групи відносять такі події, як зміна розміру екрану, закінчення завантаження даних з сервера тощо.

Додати оброблювач подій в Javascript можна декількома способами:

1. Використання атрибуту *onПодія* безпосередньо для елементу DOM, наприклад: `Гіперпосилання`. В

такому випадку значення атрибуту містить фрагмент коду Javascript. Іншими атрибутами подій можуть бути, зокрема: *ondblclick*, *onblur*, *onmousemove* та інші. Такий спосіб є найпростішим, але використовується найчастіше у випадках, коли треба виконати невеликий за обсягом код, наприклад, викликати функцію. Проте його використання призводить до небажаного змішування XHTML та Javascript коду, що ускладнює подальшу підтримку програм.

2. Використання відповідної властивості об'єкта DOM, наприклад:

```
<body>
  <p id='myEl'>Параграф</p>
  <script>
    document.getElementById('myEl').ondblclick = function() {alert('Подвійне
натискання')}
  </script>
</body>
```

В наведеному прикладі слід звернути увагу на два важливі моменти. По-перше, оброблювач події в даному випадку являє собою саме функцію, а не рядок, як це було зроблено в попередньому випадку. По-друге, фрагмент Javascript події описується після опису елемента параграфа, інакше елемент не буде знайдено.

3. Використання спеціальних методів. Особливістю цього підходу є можливість підключення та відключення декількох функцій як оброблювачів подій. Якщо перші два підходи спрацюють у будь-якому браузері, то використання методів відрізняється у Internet Explorer та у браузерах, сумісних з W3C. Для Internet Explorer виклик виглядатиме як:

```
<body>
  <div id='mySection'>Секція</div>
  <script>
    document.getElementById('mySection').attachEvent('onclick', function()
{alert('Натискання')})
  </script>
```

```
</body>
```

Для браузерів, сумісних з W3C, виклик буде дещо іншим:

```
<body>
```

```
<div id='mySection'>Секція</div>
```

```
<script>
```

```
document.getElementById('mySection').addEventListener('click', function()  
{alert('Натискання')}, false)
```

```
</script>
```

```
</body>
```

Особливістю останнього прикладу є наявність третього параметра (*true/false*), що вказує послідовність виклику події для батьківських елементів, у яких визначено оброблювач тієї ж події. При *false* обробка почнеться з найнижчого за ієрархією елемента; інакше – з найвищого, а потім буде оброблено решту в ієрархічному порядку. Крім того, слід звернути увагу, що перший параметр (назва події) записано без префіксу ‘*on*’.

Об’єкт Event

Об’єкт події доступний у тілі функції оброблювача і містить додаткову інформацію про характеристики події. Серед властивостей, що описують подію, можна виділити: координати курсору над елементом (*clientX*, *clientY*) або координати екрану (*screenX*, *screenY*); управляючі клавіші клавіатури (*altKey*, *ctrlKey*, *shiftKey*), кнопки миші (*button*), які натиснуто; назву події (*type*) та інші. Доступ до об’єкта події відрізняється у Internet Explorer та у браузерах, сумісних з W3C. Для Internet Explorer об’єкт події має назву *event* і є властивістю глобального об’єкта *Window*, а для інших браузерів він передається як параметр функції оброблювача події. Тому кросбраузерний варіант (виділений напівжирним шрифтом) записується наступним чином:

```
<body>
```

```
<div id='mySection'>Секція</div>
```

```
<script>
```

```
function func(e)
{
    e = e || window.event
    alert(e.clientX+';'+e.clientY)
}
document.getElementById('mySection').onclick=func
</script>
</body>
```

Модифікація елементів XHTML DOM

Як було сказано вище, будь-який Web-документ з огляду на інтерфейс програмування, являє собою ієрархічно побудовану структуру вкладених об'єктів, доступ до яких забезпечується засобами XHTML DOM.

Базовим об'єктом в даній об'єктній моделі є *window*. На першому рівні він включає усі глобальні змінні програми, вбудовані в браузер об'єкти *history*, *screen*, *location* та *navigator*, глобальні сервісні функції, а також об'єкт *document*, який є проекцією тега `<body>`. Всередині *document* автоматично при завантаженні сторінки будується дерево таких об'єктів, як *Form*, *Link*, *Table* тощо в залежності від наповнення XHTML-документа. Об'єкт *document* та інші елементи Web-документа, успадковані від об'єктів *DOMDocument* та *DOMElement* відповідно. Вказані об'єкти містять функціональні можливості щодо модифікації елементів XHTML DOM. Основні операції над елементами було перераховано у теоретичних відомостях. Наведемо деякі приклади роботи із DOM.

Пошук елементів

1. Пошук за назвою тега:

```
<head>
<script>
function f()
```

```

{
  for (i in document.getElementsByTagName('P'))
    document.getElementsByTagName('P')[i].style.backgroundColor="red"
}
</script>
</head>
<body>
  <p>Параграф 1</p>
  <p>Параграф 2</p>
  <p>Параграф 3</p>
  <button onclick="f()">Розфарбувати</button>
</body>

```

В даному прикладі при натисканні кнопки виконується пошук усіх параграфів і для знайдених встановлюється червоний колір фону.

2. Пошук за значенням атрибуту *id*:

```

<head>
<script>
function f()
{
  var p = document.getElementById('p1')
  p.parentNode.removeChild(p)
}
</script>
</head>
<body>
  <p id='p1'>Параграф 1</p>
  <p>Параграф 2</p>
  <button onclick="f()">Вилучити</button>
</body>

```

В даному прикладі при натисканні на кнопку «Вилучити» знаходиться параграф з *id='p1'*, присвоюється змінній *p* і потім вилучається зі списку його батьківського елемента (зникає з екрану). Метод *getElementById* на відміну від інших знаходить завжди один елемент або повертає *null*.

3. Пошук елементів на значенням атрибуту *name*:

```
<head>
<script>
function f()
{
    var p = document.getElementsByName('r1')
    for (i in p)
        p[i].selected = ""
}
</script>
</head>
<body>
<form onsubmit="f();return false;">
    <input type="radio" name="r1" selected="selected" />V1
    <input type="radio" name="r1" />V2
    <input type="radio" name="r1" />V3
    <input type="submit" />
</form>
</body>
```

Даний приклад ілюструє роботу методу *getElementsByName*. При цьому створюється форма з трьома елементами типу «перемикач один з багатьох». При натисканні на кнопку *Submit* спрацьовує відповідний оброблювач: знаходяться всі перемикачі та в циклі для кожного відміняється його виділення.

Додавання елементів у Web-документ

Створення нових елементів відбувається завдяки використанню декількох методів: *createElement* (створити елемент), *appendChild* (додати елемент у кінець списку), *insertBefore* (вставити елемент перед визначеним елементом), *insertAfter* (вставити елемент після визначеного елемента). Розглянемо приклад:

```
<head>
<script>
function f()
{
    var d = document.getElementById('container')
    for (var i=0;i<3;++i){
        var p = document.createElement('p')
        p.innerHTML="Текст"+i;
        d.appendChild(p)
    }
    var a = document.createElement('a')
    a.href = "#"
    a.innerHTML="Гіперпосилання";
    d.insertBefore(a,p)
}
</script>
</head>
<body onload="f()">
<div id="container"/>
</body>
```

В даному прикладі безпосередньо після завантаження сторінки динамічно створюються та додаються 3 параграфи з текстом Текст1, Текст2 та Текст3 відповідно. Після цього створюється новий об'єкт типу гіперпосилання і вставляється перед останнім параграфом. Властивість `innerHTML`

використовується для заповнення вмісту елемента. Слід зауважити, що використання змінної p після циклу є коректним, незважаючи на те, що оголошено її всередині циклу. Це обумовлено тим, що в Javascript найменша локальна область видимості – функція.

Вилучення елементів

Для вилучення елементів з DOM використовується функція `removeChild`:

```
<head>
<script>
function f()
{
    var c = document.getElementById('container')
    c.parentNode.removeChild(c)
}
</script>
</head>
<body onload="f()">
<div id="container">Вміст</div>
</body>
```

В цьому прикладі з документа вилучається елемент *div* безпосередньо при завантаженні сторінки. Слід зауважити, що вилучення відбувається зі списку батьківського елемента, що і проілюстровано у функції $f()$.

Заміна елементів

Заміна елементів відбувається завдяки використанню метода `replaceChild`, наприклад:

```
<head>
<script>
function f()
{
```

```

var im1 = document.getElementById('im1')
var im2 = document.getElementById('im2')
var im_im2_old = im1.parentNode.replaceChild(im1,im2)
im1.parentNode.insertBefore(im_im2_old,im1)
}
</script>
</head>
<body onload="f()">


</body>

```

В даному прикладі у функції *f()* спочатку отримуються вказівники на зображення *im1* та *im2*. Після цього на місце *im2* переноситься *im1* (вказівник на старе значення *im2* повертає функція), а останнім рядком вилучене значення *im2* вставляється перед *im1*.

Обробка форм XHTML

Валідація даних форм

Часто в задачах обробки форм виникає необхідність у перевірці коректності уведених користувачем даних у елементи форм. Зокрема, це відноситься до перевірки коректності введення дат, електронної адреси тощо. Для виконання валідації даних використовують події форми та її елементів, а також, як правило, регулярні вирази. Наведемо приклад перевірки введення рядка в текстове поле за шаблоном «тільки літери у нижньому регістрі та символ підкреслення».

```

<head>
<script>
function f()
{

```

```

    var v = document.getElementById("a").value
    var p=new RegExp("[a-z_]+$","m")
    var res = p.test(v)
    if (!res) alert('Недопустимі символи')
    return res
}
</script>
</head>
<body>
    <form action="http://example.com" onsubmit="return f()">
        Поле [a..z_]<input type="text" id="a" value=""/>
        <input type="submit" value="Перевірити"/>
    </form>
</body>

```

В даному прикладі використовується оброблювач події форми “*onsubmit*”, який перед відправленням форми перевіряє задану умову. Якщо результат перевірки негативний, виводиться повідомлення та відправлення відміняється (*return false*). Функція *f()* виконує перевірку співпадіння уведених даних із шаблоном регулярного виразу.

Створення та вилучення елемента форми

Наступний приклад ілюструє можливості вставлення та вилучення елементів форми:

```

<head>
<script>
function add()
{
    var b = document.createElement("input")
    b.type="text"
    b.id="i"+document.getElementById("frm").childNodes.length

```

```

        document.getElementById("frm").appendChild(b)
    }
    function del()
    {
        var cnt = document.getElementById("frm").elements.length
        if (cnt<3) return
        var last = document.getElementById("frm").elements[cnt-1]
        document.getElementById("frm").removeChild(last)
    }
</script>
</head>
<body>
    <form id="frm">
        <input type="button" onclick="add()" value="Створити"/>
        <input type="button" onclick="del()" value="Вилучити останній"/>
    </form>
</body>

```

В даному прикладі виводяться дві кнопки, перша додає елемент введення і встановлює йому атрибут «текстовий». Друга функція вилучає останній в списку елемент. Якщо елементів залишилось два (кнопки), то вилучення відміняється.

Обробка даних елементів форм

Наступний приклад ілюструє обробку даних елементів форми: для значень двох текстових полів виконується обчислення їх суми. У випадку, коли введені значення не числові, виводиться повідомлення про помилку:

```

<head>
<script>
    function sum()
    {

```

```

var a = document.getElementById("a").value
var b = document.getElementById("b").value
var res = document.getElementById("res")
if (isNaN(a) || isNaN(b)) res.value = "невірні дані"; else res.value =
Number(a)+Number(b)
}
</script>
</head>
<body>
<form id="frm">
  <input type="text" id="a" value=""/><br/>
  <input type="text" id="b" value=""/><br/>
  <input type="text" id="res" readonly="readonly" value="Результат"/><br/>
  <input type="button" onclick="sum()" value="Обчислити суму"/>

</form>
</body>

```

Функція *isNaN()* перевіряє чи є передане значення невірним числом.

Варіанти завдання

Забезпечити виконання завдання згідно із варіантом, сформувані необхідні дані, вбудовані у програму, у вигляді масивів, виконати тестування програми. Завдання виконується бригадами по 2 студенти, номер варіанту визначається викладачем (табл.2.1).

Варіанти завдань

№	Назва завдання	Примітки
1	2	3
1.	Валідація HTML-форми реєстрації користувача	Забезпечити перевірку елементів HTML-форми таких видів: ціле число, дійсне число, дата у форматі DD.MM.YYYY, однаковість даних у полях введення паролю (основне та перевірочне). Крім того, забезпечити перевірку введення обов'язкових полів, помічених (*).
2.	Динамічне меню	Сформувати багаторівневе меню (горизонтальне та вертикальне), пункти меню взяти із статичного масиву. При натисканні на визначений пункт меню має завантажуватися відповідна HTML-сторінка. Меню має бути присутнім на усіх сторінках.
3.	Калькулятор	Реалізувати калькулятор мовою JavaScript. Обов'язкові операції: +, -, *, /, %, корінь квадратний, x^u . Кнопки з цифрами та знаками операцій мають бути доступні у інтерфейсі вікна.
4.	Текстовий редактор	Реалізувати редактор, який би дозволяв виконувати форматування тексту, який уводиться. Обов'язкові елементи форматування: напівжирність, нахил, підкреслювання, колір тексту, регістр символів. Елементи форматування мають бути доступними через відповідні кнопки, форматований текст – у окремому вікні на сторінці.
5.	Слайд-шоу	Реалізувати можливість перегляду зображень за таймером, врахувати можливість налаштування розміру зображень, циклічність перегляду, а також часу між переключеннями.

1	2	3
6.	Календар	Реалізувати можливість вибору дати у елемент форми з спеціального візуального компоненту. Вимоги до компоненту: вибір числа, перехід між місяцями та роками, відміна вибору дати.
7.	Гістограма	Реалізувати можливість уведення довільної кількості чисел у елементи форми та побудови нормалізованої за максимальним числом гістограми значень цих полів. При візуалізації гістограми використовувати лише засоби CSS.
8.	Будильник	Реалізувати програму-будильник. Основні функції: увімкнення-вимкнення, уведення дати та часу. Врахувати можливість включення довільної кількості будильників, а також перегляд «заведених будильників». При досягненні часу спрацьовування, заданого для будильника, видавати повідомлення.
9.	Візуальний елемент «Підказка при уведенні»	Надати можливість виведення підказки при наборі тексту у елемент уведення «текстове поле». Передбачити можливість створення списку слів-підказок.
10.	Генерація змісту тексту	Обробити масив рядків, сформувати з них зміст, використовуючи функції DOM JavaScript. Розбити програмно список на 2 рівні: 1-й – перша літера рядка, 2-й – відповідний рядок. Рівні мають характеризуватися відступами від лівого краю сторінки. По натисканню на рівень має завантажуватися сторінка у розташований праворуч елемент <IFRAME>.

1	2	3
11.	Візуальний елемент «Спливаюча підказка»	Надати можливість виведення спливаючої підказки при наведенні миші над текстом Web-сторінки. Передбачити можливість створення нових підказок та їх підключення до обраного фрагменту тексту.
12.	Візуальний елемент «Дерево»	Розробити програму виведення та навігації по компоненту «дерево» (згортання/розгортання гілок) на основі багатовимірного масиву Javascript.
13.	Редактор cookie-змінних браузера	Розробити програму редагування cookie-змінних, що зберігаються у браузері. Забезпечити редагування полів cookie-змінних, що описані для протоколу HTTP: назва, значення, час застарівання, піддомен, підкаталог.
14.	Візуальний елемент «Таблиця»	Розробити візуальний редактор для створення таблиці HTML. Надати можливість встановлення кількості стовпчиків таблиці, додавання вмісту комірок таблиці, а також створення та вилучення рядків у таблиці.
15.	Елемент форми SELECT	Реалізувати власний компонент вибору одного елемента зі списку. Врахувати можливість вибору з клавіатури та мишкою. Дані взяти з масиву JavaScript.

Контрольні запитання

1. Способи встановлення оброблювача подій у Javascript.
2. Основні типи вузлів в моделі XHTML DOM.
3. Методи пошуку елементів засобами XHTML DOM.
4. Методи модифікації елементів засобами XHTML DOM.
5. Характеристика основних об'єктів браузера Javascript.
6. Особливості типів даних Javascript.

Рекомендована література

1. Гудман, Д. JavaScript и DHTML. Сборник рецептов. Для профессионалов [Текст] / Д. Гудман. — СПб. : Питер, 2004. — 523 с. — ISBN 5-94723-817-9.
2. Муссиано, Ч. HTML и XHTML. Подробное руководство [Текст] / Ч. Муссиано, Б. Кеннеди Б. — СПб. : Символ-Плюс, 2008. — 752 с. — ISBN 978-5-93286-104-2.
3. Edwards, J. The JavaScript Anthology: 101 Essential Tips, Tricks & Hacks [Текст] / J. Edwards, C. Adams. — SitePoint Pty Ltd, 2006. — 592 с. — ISBN 978-0975240267.

ЛАБОРАТОРНА РОБОТА №3

Розробка програм асинхронного обміну даними у Web-середовищі засобами бібліотеки JQuery. Ajax.

Метою лабораторної роботи є вивчення основ асинхронного обміну даними мовою JavaScript за допомогою технології Ajax та бібліотеки JQuery при реалізації практичних задач Web-програмування.

Завдання на лабораторну роботу:

1. Ознайомитись із синтаксисом та базовими можливостями бібліотеки JQuery.
2. Набути практичних навичок маніпулювання вмістом Web-сторінок.
3. Вивчити способи реалізації інтерактивного інтерфейсу користувача за допомогою засобів JQuery Ajax.

Основні теоретичні відомості

Бібліотека JQuery як інструмент розробки інтерактивних Web-додатків

Активний розвиток застосування різноманітних Web-додатків, зокрема, соціальних мереж та засобів електронної комерції в Internet, зумовив необхідність створення інтерактивних інтерфейсів користувача, які за функціональними можливостями наближуються до традиційних віконних додатків. Місце програмного інструментарію для вирішення цієї задачі за останні 5-7 років зайняли різноманітні бібліотеки на основі вбудованої в Web-браузер мови Javascript. Серед таких засобів найбільш поширеною та вдалою виявилась бібліотека JQuery. Вона поєднує в собі лаконічний синтаксис, надзвичайно розвинуті засоби пошуку елементів Web-сторінки на основі CSS та XML-орієнтованої спеціалізованої мови XPath, функції маніпуляції XHTML DOM, анімаційні ефекти, а також ефективний інструментарій асинхронного обміну інформацією з Web-сервером Ajax.

Базовий синтаксис команди JQuery

Оскільки бібліотека JQuery написана мовою Javascript, то застосовувати код можна всюди, де допустимий код Javascript: в оброблювачах подій, викликаючи функції тощо. Спрощено запис команди JQuery виглядає наступним чином:

\$(selector).action1(args1).action2(args2) ...

або

\$.action(args)

де знак *\$* означає скорочену назву об'єкта JQuery. Цей знак можна замінити на виклик: *JQuery(selector)*;

selector визначає один або групу елементів Web-документа, об'єднаних за певним критерієм (назвою тега, ідентифікатору, CSS-класу тощо);

action, action1, action2 – визначають функцію, що треба застосувати для відібраних селектором елементів (встановити стиль CSS, заповнити/стерти вміст елемента, додати/вилучити оброблювач деякої події тощо);

arg, arg1, arg2 – необов'язкові аргументи функцій, що застосовуються до елементів (новий вміст елемента, його атрибуту тощо).

Якщо вказано декілька дій (*action*) до відібраних селекторів, то вони виконуються послідовно, наприклад:

\$("p").html("abcd").css("color", "green")

В цьому прикладі для усіх параграфів буде внесено текст *"abcd"* та встановлено колір тексту в зелений.

Селектори JQuery

Бібліотека JQuery надає широкий спектр можливостей щодо пошуку елементів Web-сторінки з використанням синтаксису CSS та XPath. Наприклад, щоб застосувати деяке правило CSS для усіх елементів, достатньо виконати команду: *\$('*').css('color', 'blue')*. Інші приклади:

`$("#input:checked").length` – повертає кількість елементів форм, які виділені;

`$("#td:empty").text("порожній").css('background', 'red')` – знаходить усі комірки таблиці `<td>` за умови, що вміст їх порожній (псевдоклас *empty*), після цього вносить в ці комірки текст «порожній» та встановлює для них колір фону в червоний.

`$("#d1,#s1,#par24").css('border','1px dashed black')` – знаходить елементи з ідентифікаторами *d1*, *s1*, *par24* та встановлює для них пунктирну границю розміром 1px чорним кольором;

`$("#input[value='321']").val('123')` – повертає усі елементи форм, в які уведено значення «321» і замінює це значення на «123»;

`$("#div[id]").text("My text")` – знаходить усі елементи `<div>`, що містять атрибут *id* та заповнює їх вміст текстом «My text».

Методи уточнення та обробки результатів відбору елементів JQuery

Після відбору елементів засобами селекторів, бібліотека JQuery має можливість додатково відфільтрувати результат:

а) *children()* – отримати усіх нащадків для знайдених елементів:

`$("#d1").children().css('background-color', 'red')` знаходить усіх нащадків елемента з ідентифікатором *d1* та застосовує для них правило CSS;

б) *filter(x)* – відфільтрувати елементи, використовуючи замість «х» функцію, інший селектор або елемент JQuery:

```
$('#p').filter(function(index) {  
    return $(this).text().length > 3;  
}).css('background-color', 'red')
```

В наведеному прикладі знаходяться усі параграфи, до них додатково застосовується функція фільтрації, яка перевіряє довжину тексту всередині. Якщо довжина більша трьох, то для таких параграфів буде застосовано правило CSS;

в) *each()* – виконати ітерацію по усіх знайдених елементах:

```

$('div').each(function(index) {
    alert(index + ': ' + $(this).text());
});

```

В даному прикладі буде знайдено усі елементи `<div>` та для кожного з них застосується функція, яка виведе номер елемента та його вміст.

г) `map()` – застосувати функцію для кожного зі знайдених елементів:

```

$('input').map(function() { return this.id; }).get().join(',');

```

В цьому прикладі буде знайдено усі елементи форм, в кожного з них буде взято ідентифікатор і з них буде сформовано рядок виду: `id1,id2,id3`; де `id1,id2,d3` – ідентифікатори елементів.

Маніпуляція вмістом Web-документа

До групи методів маніпуляції елементами сторінки відносяться: додавання/вилучення властивостей CSS, встановлення/очищення текстового вмісту елемента та його дочірніх елементів тощо.

1) `css(property)`, `css(property,value)` – отримує та встановлює значення властивості CSS відповідно:

```

var a = $('#e1').css('position') // взяти значення position для елемента з id=e1

```

```

$('p').css('position',a) // встановити усім параграфам значення властивості position, як у змінній a

```

2) `html()`, `html(value)` – отримує та встановлює значення HTML-вмісту, включаючи теги та вміст дочірніх елементів, відповідно:

```

alert($('#a1').html()) // вивести вміст елемента з id=a1

```

```

$('#a1').html('<span>123</span>') // встановити вміст елемента з id=a1 значенням <span>123</span> (створити елемент <span> всередині a1)

```

3) `remove()` - вилучити знайдені елементи:

```

$('.c1, #e1').remove(); // вилучити наступні елементи: у яких клас = c1; з id=e1

```

3) `empty()` - очистити знайдені елементи:

`$('#p').empty();` // очистити вміст усіх параграфів

4) *`after()`, `before()`* – вставити вміст після/до знайдених елементів:

`$('#d1').after('000')` // вставити текст 000 після елемента з `id=d1`

`$('#d1').before("Link")` // вставити гіперпосилання перед елементом з `id=d1`

5) *`text()`, `text(value)`* – отримує та встановлює значення текстового вмісту, без тегів, але із текстовим вмістом дочірніх елементів, відповідно:

`alert($('#a1').text())` // вивести вміст елемента з `id=a1`

`$('#a1').text('123')` // встановити вміст елемента з `id=a1` значенням `123` (елемент `` не створюється, вставляється лише текст «`123`»)

Оброблювачі подій

Завдяки оброблювачам подій програміст має змогу застосовувати той чи інший програмний код, реагуючи на дії користувача (події від миші, клавіатури, операцій уведення/виведення). Розглянемо способи встановлення оброблювачів подій:

1) *`bind(eventType [, eventData], handler(eventObject))`* – універсальний метод встановлення оброблювача події до знайдених елементів, де *`eventType`* – тип події, яка оброблюється, *`eventData`* – необов'язковий параметр, який містить дані, що передадуться оброблювачу, *`handler(eventObject)`* – функція-оброблювач події:

`$('#p').bind('click', function() { alert($(this).text()); });` // для усіх `<p>` встановити як оброблювач події натискання на ліву кнопку миші функцію, яка виводить вміст елемента

`$('#p').bind('click',function(e){alert(e.pageX+';'+e.pageY)})` - // для усіх `<p>` встановити як оброблювач події натискання на ліву кнопку миші функцію, яка виводить координати курсору в точці, де відбулось натискання

Зауваження. 'e' – параметр, який передається будь-якому оброблювачу і уточнює подію, містить властивості про натиснуту клавішу, координати курсора, про тип елемента та інші.

2) *click()*, *dblclick()*, *focus()*, *blur()*, *change()*, *load()*, *unload()* – методи для обробки найпоширеніших подій: натискання та подвійного натискання кнопки миші, встановлення/втрати фокусу елемента, зміни його вмісту, завантаження/вивантаження елемента (зображення, сторінки).

```
$(':input').change(function() {  
    alert('вміст змінено');  
});
```

3) *unbind([eventType] [, handler(eventObject)])* – відмінняє використання оброблювача події, де *eventType* – тип події, яка оброблюється, *handler(eventObject)* – функція-оброблювач події:

```
$('div').unbind('click'); // усім <div> відмінити обробку події onClick
```

```
var handler = function() {  
    alert('оброблювач');  
};
```

```
$('#a1').bind('click', handler);
```

```
$('#a1').unbind('click', handler); // handler – вказівник на функцію, що  
використовувались при bind
```

4) *\$(document).ready(handler)* – оброблювач події «завантажено DOM-модель».

Даний тип подій застосовується замість події “onload” елемента <body>:

```
<body onload="alert(1)">... </body>
```

На відміну від “onload” оброблювач “ready” виконується раніше – в момент, коли сформовано дерево DOM об’єктів сторінки. В той же час “onload” викликається пізніше, а саме після завантаження усіх сторонніх об’єктів сторінки: зображень, флеш-компонентів тощо. Приклад оброблювача

```
$(document).ready(function() {  
    alert('Сторінка завантажена')  
});
```

Асинхронний обмін інформацією з Web-сервером. Ajax

Ajax (асинхронний Javascript) – це підхід до створення інтерактивних Web-додатків, згідно з яким обмін даними між Web-браузером та Web-сервером відбувається у фоновому режимі та надає можливості оновлювати лише частину Web-сторінки. Такий підхід зменшує мережний трафік та дозволяє будувати додатки, наближені до «віконних», які є більш звичними для користувача. Головним чином підхід базується на використанні засобів мови Javascript. При цьому з серверного боку жодних змін виконувати не потрібно у порівнянні з традиційним підходом. На клієнті за Ajax відповідає об'єкт Javascript XMLHttpRequest. Передача даних за допомогою цього об'єкта може відбуватись як у синхронному, так і у асинхронному режимі. Складність безпосереднього використання цього об'єкта зумовлена неповною сумісністю реалізації в різних браузерах. Тому, як правило, використовують крос-платформні бібліотеки, які уніфікують доступ до даного об'єкта. Однією з таких бібліотек є JQuery. З огляду на задачі Ajax JQuery пропонує наступні можливості:

1. Завантаження даних з сервера безпосередньо у елемент сторінки (наприклад, у параграф) – метод *\$.load()*.
2. Завантаження та виконання програмного коду мовою Javascript – метод *\$.getScript()*.
3. Завантаження даних із сервера у форматі JSON, що спрощує перетворення даних у об'єкти Javascript – метод *\$.getJSON()*
4. Відправлення даних на сервер - метод *\$.post()*
5. Обробка помилок, що виникли при передачі даних.
6. Серіалізація даних HTML-форми для подальшої передачі на сервер.
\$(<selector>).serialize()

Методичні вказівки

Використання серверних Web-технологій

Для виконання даної лабораторної роботи необхідно додатково встановити Web-сервер. Рекомендовано використовувати Web-сервер Apache, виходячи з його широкого розповсюдження та простоти інсталювання. Для ОС сімейства Windows Web-сервер Apache являє собою повноцінний дистрибутив, встановлення якого потребує мінімальної участі користувача. При встановленні Apache треба переконатись, що TCP-порт комп'ютера з номером 80 вільний. Це можна зробити шляхом виклику із термінального вікна (*cmd*) команди

netstat -a -n -p TCP

В результуючому списку не повинно бути запису з портом 80. Для перевірки коректності встановлення Web-сервера необхідно у адресному рядку браузера (Internet Explorer, Mozilla Firefox або іншому) увести команду *http://localhost*. При цьому у відповідь на запит Web-сервер поверне список файлів його кореневого каталогу або сторінку привітання (в залежності від версії Web-сервера). Обидва результати свідчать про успішне встановлення. Особливістю роботи Web-сервера Apache є його функціонування у вигляді служби Windows, що призводить до автоматичного запуску Web-сервера при завантаженні ОС Windows. Файли сайту такі, як html, php, css, js та інші зберігаються в спеціальному каталозі Web-сервера. За замовчуванням він знаходиться у тому ж місці, де і решта файлів Web-сервера – в каталозі *htdocs*. Вміст саме цього каталогу виводиться за запитом *http://localhost* (рис.3.1).



Рис.3.1. Приклад виведення після успішного встановлення Apache

За необхідності шлях до цього каталогу можна змінити, для цього треба відкоригувати файл налаштувань Web-сервера *httpd.conf*, який знаходиться в підкаталозі *conf* Web-сервера. За кореневий каталог відповідає параметр *DocumentRoot*. Після корекції даних конфігурації Web-сервер необхідно перезавантажити через панель управління Windows (наприклад, Адміністрування-Служби-Apache2.2). Файл конфігурації, крім цього, дозволяє налаштувати віртуальні хости (комп'ютери), обмежити доступ до файлів та каталогів сайту, налаштувати переадресацію, а також підключити модулі для обробки певного типу запитів.

Для виконання роботи достатньо перенести усі файли (*html*, *js*, *css*, *txt*) у каталог *htdocs*.

Застосування Ајах для асинхронного завантаження XML-документа

В задачах Web-програмування часто виникає необхідність обробки XML-документів. Прикладом може служити динамічне багаторівневе меню, ієрархічний каталог деяких об'єктів, зокрема, товарів за категоріями, фрагмент файлової системи тощо. Перевагою бібліотеки JQuery є наявність в ній вбудованих засобів роботи з XML в той же спосіб, що і зі звичайною Web-сторінкою: є можливість відбирати елементи XML-документа, використовуючи синтаксис CSS та XPath, застосовувати до них операції отримання/встановлення вмісту, атрибутів тощо.

Нехай на сервері розташовано XML-документ (data.xml) наступного вигляду:

```
<people>
  <man id="m1">
    <name id="n1">John</name>
  </man>
  <man id="m2"/>
  <man id="m3"/>
</people>
```

Необхідно асинхронно завантажити документ, вивести ідентифікатори вузлів та вміст елементів *<name>*.

Для реалізації даної задачі будемо використовувати найбільш універсальну функцію JQuery з назвою *.Ajax()*. Вона дозволяє встановити метод HTTP за яким треба виконати запит на сервер, тип ресурсу, який запитується, описати функцію успішного завантаження та встановити інші параметри:

```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script>
$(document).ready(function(){
$.ajax({
  type: "GET",
  url: "data.xml",
  dataType: "xml",
  success: function(xml) {
    $(xml).find('man').each(function(){
      document.writeln($(this).attr("id"))
      $(this).find('name').each(function(){
document.writeln($(this).text())})
    })
  }
})
}
```

```

        }
    })
})
</script>
</head>
<body>
</body>
</html>

```

В цьому прикладі описано оброблювач події `ready()`, який викликається автоматично при завантаженні сторінки. При цьому виконується запит функцією `ajax`. При успішному виконанні запиту (властивість `success`) в оброблювач буде передано параметр `xml`, який і буде відображенням файлу `data.xml`. Всередині цієї функції виконується прохід по елементам 1-го рівня (`$(xml).find('man').each(...)`) із виведенням значення атрибута `id` кожного з елементів. Крім того, для кожного елемента 1-го рівня виконується пошук елементів з атрибутом `name` і виведення значення його `id`. В результаті у вікні браузера буде виведено: `m1 John m2 m3`.

Застосування Ajax для асинхронного завантаження об'єкта Javascript

В Javascript для зручності обробки використовується спеціальний формат подання об'єктів цієї мови JSON (об'єктний запис Javascript). Його особливістю є зберігання об'єктів у вигляді пар: ключ/значення. Ключ – це рядок, який автоматично перетвориться у назву властивості об'єкта; значення – рядок, об'єкт, масив Javascript. Очевидною перевагою JSON-формату у порівнянні з XML є компактне подання даних. Для прикладу розглянемо задачу завантаження персональних даних про особу (ім'я та його вік) у поля форми XHTML. Дані зберігатимемо на сервері у файлі `user.json`. Вміст цього файлу буде наступним:

```

{
  "data": [

```

```

    {
      "name": "user1",
      "age": "20"
    },
    {
      "name": "user2",
      "age": "19"
    }
  ]
}

```

Фігурні дужки в файлі *user.json* означають запис об'єкта. Всередині – під ключем “*data*” зберігаються дані про осіб, при чому значення подано у вигляді списку (масиву), в якому кожний елемент – це об'єкт з двома властивостями: *name* (ім'я), *age* (вік). Реалізує задачу наступний код HTML/Javascript:

```

<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script>
function f(n){
$.getJSON('user.json', {}, function(json) {
    $("#username").val( json["data"][n].name )
    $("#userage").val( json["data"][n].age )
  })
}
</script>
</head>
<body>
<form>
  <input type="text" value="" id="username" /> <br/>
  <input type="text" value="" id="userage" /> <br/>

```

```

</form>

<button value = "User1" onclick="f(0)">User1</button>

<button value = "User2" onclick="f(1)">User2</button>

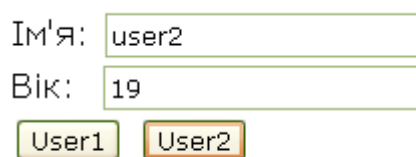
</body>

</html>

```

В даному прикладі буде виведено два елемента форми та дві кнопки. При натисканні першої кнопки заповнюються дані про першу особу, при натисканні другої – про іншу.

Виведення буде наступним (рис.3.2).



Ім'я:

Вік:

Рис.3.2. Виведення програми-прикладу JSON

При натисканні на кнопку *User1* або *User2* викликається функція *f()*, де виконується запит до сервера, завантажуються дані з файлу *user.json* та формується об'єкт мови Javascript, який передається функції-оброблювачу. На основі вмісту цього об'єкту встановлюються відповідні значення полів *username* та *userage*. Оскільки в JSON-файлі використовувався масив, тому можемо звертатись до даних по індексу: *json["data"][n].name*.

Функції *\$.get()*, *\$.post()* та *\$(selector).load()*

Дані функції є скороченнями функції *ajax()* і призначені відповідно до завантаження (відправлення) даних відповідно методами GET або POST, а також методом GET безпосередньо у елемент.

Наприклад, задано три елемента *div*, два елементи уведення форми та кнопка відправлення даних на сервер (рис.3.3):

click d1
Send the form data...
click d3

Рис.3.3. Початкова форма введення прикладу *\$.get*, *\$.post()*, *\$(sel).load()*

При натисканні на текст «*click d1*» буде використано метод *\$.get()*, при натисканні на *d3* – *load()*, а при натисканні на кнопку Send будуть відправлені на сервер дані з форми методом *\$.post()*, сервер їх поверне в тому ж вигляді, а оброблювач *\$.post()* розмістить результат всередину *d2* (рис.3.4).

Вміст #d1
Form data 1;Form data 2
Вміст #d3

Рис.3.4. Результат виконання методів *\$.get*, *\$.post()*, *\$(sel).load()*

Вихідний текст *html* та *javascript* ілюструє наступний фрагмент:

```

<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script>
function f()
{
$.post('form.php', $('#frm').serialize(), function(data) { $('#d2').html(data) })
return false;
}
</script>
</head>
<body>
<div id="d1" onclick="$.get('1.txt', function(data) { $('#d1').html(data)
})">click d1</div>
<div id="d2">Send the form data...</div>
<div id="d3" onclick="$('#d3').load('3.txt')">click d3</div>

```

```

<form id="frm" onsubmit="return f()">
  <input type="text" id="i1" name="i1" />
  <input type="text" id="i2" name="i2" />
  <input type="submit" value="Send" />
</form>
</body>
</html>

```

Особливістю даного фрагмента є передача даних з форми на сервер без перезавантаження основної сторінки. В прикладі для цього використано запис вигляду: `$.post('form.php', $('#frm').serialize(), function(data) { $('#d2').html(data) })`. Тут викликано файл *form.php* на сервері (його вміст: `<? echo $_POST["i1"].".".$_POST["i2"]; ?>`, що означає повернення отриманих даних клієнту). Як другий параметр передано результат функції *serialize()*, застосованої до форми *frm*. *serialize()* – функція, яка збирає усі значення елементів форми у єдиний рядок, розділений символом `&` – в тому ж форматі, як і при відправленні методом GET на сервер. Третій параметр методу *post* – функція обробки результату, отриманого з сервера. В даному випадку остання заповнює вміст елемента `<div id="d2">`.

Варіанти завдання

Завдання виконуються згідно з таблицею варіантів бригадами не більш, ніж по 2 студенти в кожній. Код JavaScript (jQuery), HTML та CSS має зберігатися у окремих файлах. Джерелом даних є XML-файли, які зберігаються на Web-сервері.

Особливості оформлення протоколу. Протокол має включати: титульний аркуш, текст завдання, структуру файлу XML у вигляді XML DTD, фрагмент програмного коду мовою Javascript, а також 2-3 копії екранних форм.

Оцінювання роботи. При оцінюванні роботи враховується: рівень відповідності вимогам до завдання, оформлення роботи, якість програмного коду та структури XML-файлу з даними.

Додаткове програмне забезпечення. Для отримання даних, які зберігаються у файлах XML на сервері, слід встановити Web-сервер Apache 2.2. Для зберігання даних, отриманих від клієнта з HTML-форм та збереження їх у файлі, використовувати мову PHP 5.

Варіант завдання визначається викладачем (див. табл. 3.1).

Таблиця 3.1

Варіанти завдань

№	Назва завдання	Вимоги щодо виконання роботи
1	2	3
1.	Інтерактивна HTML-форма	Реалізувати можливість інтерактивної перевірки коректності заповнення HTML-форми реєстрації користувача. Включити такі елементи: 1) довжина та складність пароля; 2) перевірка наявності імені користувача в системі; 3) інтерактивна підказка країни та міста мешкання. Поля та їх типи для перевірки та підказки про помилки взяти з XML-файлу. Усі перевірки виконувати (і сигналізувати в разі помилки) інтерактивно. Результати перевірки заносити в інший XML-файл.
2.	Аjax-таблиця	Реалізувати можливість редагування та перегляду таблиці HTML із полями: ціле, символьний рядок, елемент зі списку. Включити можливість вставки, вилучення та редагування комірок таблиці. Синхронізувати кожну дію з інформацією у XML-файлі.
3.	Аjax-дерево	Реалізувати інтерактивне завантаження гілок дерева (наприклад, каталогу документів) з бази даних, при виборі визначеної гілки відобразити текст Web-сторінки. Структуру дерева, його вміст та відповідні гілкам HTML-сторінки зберігати у XML-файлі. Навігація по дереву і відображення HTML-сторінки відбувається без перезавантаження основної сторінки.
4.	Аjax-редактор текстових файлів	Надати можливість створення та редагування умовних текстових файлів з сервера. Передбачити виведення списку вже існуючих файлів. Файли зберігати на сервері всередині XML-документа. Реалізувати функції перегляду, запису та вилучення файлів. Операції пересилання даних виконувати асинхронно.

1	2	3
5.	Аjax-електронна пошта	Надати можливість обміну повідомленнями між двома користувачами. Передбачити такі функції, як перевірка поштової скриньки, створення та відправлення нового листа, перегляд та вилучення існуючого листа. Листи зберігати на сервері в XML-файлі. Операції пересилання даних виконувати асинхронно. Реалізувати функцію входу користувача за паролем.
6.	Аjax-щоденник	Реалізувати можливість ведення інтерактивного щоденника із можливістю створення нових тем, відповідних повідомлень та врахування дати їх створення. Крім того, реалізувати можливість перегляду повідомлень та їх вилучення. Дані зберігати у вигляді XML-файлі на сервері.
7.	Аjax-фотогалерея	Реалізувати можливість створення фотогалереї із функціями додавання та вилучення зображення, а також перегляду наявних фотографій як окремо, так і у вигляді слайд-шоу. Файли зображень завантажувати на сервер, інформацію про список файлів зберігати у XML-файлі на сервері.
8.	Гістограма	Реалізувати можливість побудови гістограми засобами CSS. Кожний стовпчик гістограми будувати на основі генерації псевдовипадкового числа (або при уведенні користувачем), який додається у файл на сервері у форматі XML у асинхронному режимі. За сигналом від таймера оновлювати зображення гістограми.
9.	Аjax-авторизація користувачів	Реалізувати можливість створення, вилучення та блокування облікових записів користувачів (ім'я, пароль) і віднесення їх до однієї з трьох груп. У відповідності до групи при авторизації користувача видавати йому одну з трьох HTML-сторінок. Усі операції виконувати без перезавантаження сторінки. Дані про користувачів зберігати у XML-файлі на сервері.
10.	Аjax-словник	Реалізувати можливість інтерактивного створення, наповнення та вилучення слів у словнику, який зберігається на сервері у вигляді XML-файлу. Окремий запис словника являє собою пару слів «слово українською мовою» - «слово англійською мовою». Забезпечити можливість перекладу слів в обидва напрями без перезавантаження сторінки.

1	2	3
11.	Аjax-персоналізатор сторінки	Реалізувати можливість персоналізації сторінки за запитом користувача. Передбачити можливість зміни фонового зображення сторінки, шрифту текстових фрагментів та заголовків сторінки. Крім того, дозволити користувачам створювати нові стилі, на основі передбачених властивостей. Дані зберігати в XML-файлі на сервері. Операції пересилання даних виконувати асинхронно.
12.	Аjax-редактор секцій сторінки	Реалізувати можливість переміщення основних блоків сторінки за запитом користувача. Основні блоки: меню, основний вміст, нижній колонтитул. Створити декілька схем виведення блоків і надати можливість користувачеві обирати одну из них. Крім того, дозволити користувачеві запам'ятовувати свій вибір для подальшого відновлення. Дані зберігати в XML-файлі на сервері. Операції пересилання даних виконувати асинхронно.
13.	Аjax-планувальник справ	Реалізувати можливість створення списку нагадувань щодо справ на день. Передбачити функцію візуального сповіщення про настання запланованої події. Кожна подія включає наступні поля: назва, текст, запланований час. Дані зберігати у вигляді XML-файлу на сервері, передачу даних виконувати асинхронно.
14.	Аjax-меню	Реалізувати динамічне формування і завантаження меню та відповідної Web-сторінки без перезавантаження основної сторінки. Структуру меню, включаючи вкладені рівні необхідно завантажувати з XML-файлу.
15.	Аjax-закладки	Сформувавши 5 закладок на основній Web-сторінці, кожна з яких відповідає окремій сторінці. При переході на закладку необхідно завантажити її вміст без перезавантаження основної. Назви, кількість та вміст закладок брати з XML-файлу на сервері.

Контрольні запитання

1. Відмінність синхронного та асинхронного режиму передачі даних Аjax-технології за допомогою бібліотеки JQuery.
2. Характеристика можливостей бібліотеки JQuery щодо модифікації HTML-сторінки.
3. Характеристика можливостей бібліотеки JQuery щодо модифікації CSS властивостей HTML-сторінки.
4. Специфіка роботи бібліотеки JQuery з подіями Web-сторінки.

Рекомендована література

1. Крейн, Д. Аjax в действии [Текст] / Д. Крейн, Э. Паскарелло, Д. Джеймс. — М. : Вильямс, 2008. — 640 с. — ISBN 978-5-8459-1034-9.
2. Бибо, Б. jQuery. Подробное руководство по продвинутому JavaScript [Текст] / Б. Бибо, И. Кац. — М. : Символ-Плюс, 2011. — 624 с. — ISBN 978-5-93286-201-8.
3. Закас, Н. Аjax для профессионалов [Текст] / Н. Закас, Д. Мак-Пик, Д. Фоссет. — М. : Символ-Плюс, 2008. — 488 с. — ISBN 978-5-93286-081-6.