МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Методичні рекомендації
до самостійної роботи
з навчальної дисципліни

"ОСНОВИ ПРОЕКТУВАННЯ WEB-ВИДАНЬ"

для студентів напряму підготовки
6.051501 "Видавничо-поліграфічна справа"
всіх форм навчання

Харків. ХНЕУ ім. С. Кузнеця, 2015

Затверджено на засіданні кафедри комп'ютерних систем і технологій.

Протокол № 7 від 03.02.2015 р.

Самостійне електронне текстове мережне видання

Укладач Молчанов В. П.

М 54 Методичні рекомендації до самостійної роботи з навчальної дисципліни "Основи проектування WEB-видань" для студентів напряму підготовки 6.051501 "Видавничо-поліграфічна справа" всіх форм навчання : [Електронне видання] / уклад. В. П. Молчанов. – Х. : ХНЕУ ім. С. Кузнеця, 2015. – 56 с. (Укр. мов.)

Наведено методи, засоби та завдання для організації самостійного вивчення та поглиблення отриманих у рамках лекційного курсу і аудиторних лабораторних робіт компетентностей, знань, вмінь та навичок. Подано перелік необхідної для виконання завдань літератури.

Рекомендовано для студентів напряму підготовки 6.051501 "Видавничо-поліграфічна справа" всіх форм навчання.

Вступ

Навчальна дисципліна "Основи проектування WEB-видань" є вибірковою і вивчається студентами напряму підготовки "Видавничо-поліграфічна справа" всіх форм навчання протягом четвертого семестру. Навчальна дисципліна потребує від студентів інтенсивної самостійної роботи зі спеціальною літературою та програмним забезпеченням у час, вільний від обов'язкових навчальних занять. До методичних рекомендацій включені теми, які не розглядалися раніше в методичних рекомендаціях до лабораторних робіт [2].

Метою самостійної роботи є поглиблення знань, які були отримані на лекційних заняттях, а також підтвердження і реалізація навичок, що були сформовані на лабораторних заняттях. Важливим завданням самостійної роботи є формування компетентностей, що дозволяють студенту реалізовувати на практиці отримані знання.

Студенти повинні розширити свої знання про технологічні засоби створення WEB-ресурсів, вивчити матеріали, наявні в мережі Інтернет за досліджуваним питанням, і виконати запропоновані завдання.

Основні види самостійної роботи, які запропоновані студентам із навчальної дисципліни:

вивчення лекційного матеріалу;

робота з вивчення рекомендованої літератури;

вивчення основних термінів та понять за темами дисципліни;

підготовка до лабораторних занять;

перевірка особистих знань за запитаннями для самостійного контролю та виконання контрольних завдань;

робота над індивідуальним завданням.

Для закріплення і перевірки набутих компетентностей під час самостійної роботи до кожної роботи пропонуються довідкові матеріали, практичні завдання і запитання для самодіагностики.

Змістовий модуль 1. HTML і його використання

Тема 1. Проектування WEB-сайта

Самостійна робота № 1. Створення концепції WEB-сайта

Мета роботи: отримання знань та навичок у плануванні та організації робіт зі створення сайта.

У результаті виконання самостійної роботи у студента формуються такі **компетентності:** здатність самостійно визначати мету створення WEB-сайта, вимоги до сайта, план створення сайта, вміння оформляти документально власне бачення майбутньої роботи.

Результатом виконання самостійної роботи є звіт із виконання завдання, який містить створені документи.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно після виконання першої лабораторної роботи "Проектування WEB-сайта".

1. Знайти визначення термінів концепція, бриф, технічне завдання. Знайти зразки відповідних документів.
2. Розробити бриф для своєї публікації.
3. Сформулювати цілі і завдання своєї публікації.
4. Сформулювати специфічні риси майбутніх відвідувачів, які будуть ураховуватися під час проектування.
5. Обґрунтувати структуру майбутнього сайта, сформулювати фактори, що впливають на вибір його структури.
6. Підготувати модульну сітку, ескізи сторінок (схеми розміщення елементів).
7. Розробити концепцію для своєї публікації.

Контрольні запитання для самодіагностики

1. Сформулюйте напрями розширення сфери використання мережі Інтернет.
2. Які технології використовуються для додання сторінкам динамічних властивостей?
3. Що таке RSS?
4. Знання яких програмних засобів і для чого вони можуть знадобитися творцеві WEB-документів?

5. Опишіть процес взаємодії сервера і браузера.

6. Що таке юзабіліті?

7. Що таке бриф?

8. Опишіть можливий вид подання концепції сайта.

9. Сформулюйте зміст етапів з реалізації проекту створення WEB-сайта.

Методичні рекомендації

Створення сайта – це не разовий акт, а достатньо складний багато-етапний проект. Практика, що склалася, дозволяє розглядати його як послідовність таких етапів:

визначення цілей і завдань проекту;

позиціонування проекту, тобто визначення його ролі і аудиторії;

розробка концепції сайта;

розробка технічного завдання (ТЗ) на сайт, підсумкового кошторису і календарного плану робіт;

розробка дизайн-концепції сайта;

розробка макета (ескіза) головної сторінки;

розробка макетів (ескізів) внутрішніх сторінок;

HTML-верстка сайта;

розробка елементів оформлення (логотипи, флеш, шрифти і т. д.);

збірка сайта і розробка додаткового функціоналу (скрипти, "движки" і тому подібне);

контент-наповнення сайта, наповнення БД ;

запуск пілотної версії (бета-версії) сайта, тестування, усунення помилок;

перенесення сайта на хостинг, тестування, відкриття сайта.

Цілі і завдання проекту формулюються в ході роботи із замовником, а це етап неформальний і дуже відповідальний. Результат має бути конкретним, але залишати творчу свободу розробникові.

У ході позиціонування проводиться аналіз діяльності замовника (що він пропонує), визначення аудиторії (на кого розрахований), шукаються і аналізуються аналоги (проводиться їх аналітика). За наявності ресурсів (коштів) для визначення можливої аудиторії (кола користувачів) можуть проводиться соціологічні дослідження.

На цьому етапі надзвичайно важливо врахувати думку замовника. Робиться це в ході співбесіди або проведення опитування за заздалегідь

складеним питанням. Останнім часом поширення набуло використання брифу – спеціально розробленої анкети, що заповнюється замовником. Таким чином, заповнений і підписаний бриф містить опис сайта, яким його бачить замовник, і використовується для розробки концепції сайта.

Концепція створення сайта – це документ, що містить опис майбутнього сайта таким, яким його бачить розробник. У ньому відображаються такі питання.

1. Мета створення. Повинен бути сформульований кінцевий результат, який очікується від успішно завершеного проекту. Можливі формулювання можуть виглядати так:

забезпечити доступ користувачам мережі Інтернет до основних електронних інформаційних ресурсів ...;

зробити діяльність .... більш відкритою ...;

надати можливість для зв'язку ...;

створити єдине інформаційне середовище для ...;

сформувати імідж.

2. Завдання, що вирішуються. Це окремі кроки, що ведуть до досягнення кінцевого результату, як правило, пов'язані з конкретною функцією, що надається відвідувачам сайта. Приклади часто вирішуваних завдань:

реєстрація та розмежування прав користувачів;

надання конкретних інструментів для виконання дій;

збір побажань, відгуків, замовлень тощо.

3. Тип створюваного ресурсу: персональний, інформаційно-довідковий, рекламний, розважальний, освітній, навчальний тощо.

4. Передбачувані відвідувачі – це характеристика особливостей відвідувачів, яка буде врахована під час розробки.

5. Тип доступу – загальний, розмежування на зареєстрованих та незареєстрованих відвідувачів, авторизований, корпоративний і т. д.

6. Функціональність. У ході висвітлення цього питання формулюються функції, які потребуватимуть розробки програмних елементів (клієнтських або серверних), наприклад:

надання доступу до інформації з конкретних баз даних;

публікація новин та оголошень;

ведення блогів;

створення електронних поштових повідомлень за наданими адресами;

зворотний зв'язок для проведення опитувань і збору відгуків;
збір замовлень;
реєстрація відвідувачів із метою збору статистичних даних;
форуми для обговорень;
різні сервіси (переклад текстів, розрахунки тощо).

Література: основна [2; 3]; додаткова [5; 9].

Тема 2. Розмітка тексту з використанням HTML

Самостійна робота № 2. Нові можливості мови HTML5

Мета роботи: отримання знань з нових засобів створення WEB-сторінок, а також навичок використання нових можливостей мови HTML5.

У результаті виконання самостійної роботи у студента формуються **компетентності:** здатність самостійно вивчати нові технології, створювати сторінки для мережі Інтернет з використанням мови розмітки HTML5.

Результатом виконання самостійної роботи є звіт із виконання завдання, та створені відповідно до завдання документи.

Завдання для самостійної роботи

1. Вивчити довідкові матеріали до самостійної роботи і вказану літературу.
2. Створити сторінку з використанням семантичних тегів розмітки HTML5.
3. Створити сторінку і протестувати відтворення в різних браузерах мультимедійних файлів різних типів.
4. Вивчити питання використання засобів створення динамічних зображень, створити сторінку з динамічною графікою за власним задумом.
5. Вивчити основний синтаксис SVG, і створити файл із малюнком за власним задумом та сторінку для його відображення.
6. Вивчити нові можливості, надані HTML5 для створення форм, створити сторінку з формою для вирішення гіпотетичного завдання за власним задумом.

Контрольні запитання для самодіагностики

1. Поясніть термін "семантична розмітка"?

2. Що дає HTML5 для дизайну?

3. У чому відмінність у вбудовуванні мультимедійних файлів різними тегами?

4. Що таке динамічна графіка?

5. Що таке зображення, створене на холостому canvas? Чи можна застосувати до нього таблицю стилів?

6. Дайте порівняльну характеристику графіки SVG і canvas?

7. Що нового надає HTML5 для створення форм?

Методичні рекомендації

Мова розмітки HTML5 є подальшим розвитком мов розмітки для мережі Інтернет, в стандарт якої закладено багато нових можливостей. Поки що різні браузери підтримують ці нововведення різною мірою. У зв'язку з цим доцільно розглянути засоби, що впливають на вигляд і зручність розмітки, залишивши осторонь такі, поза сумнівом, важливі і цікаві, як геолокацію, створення локальних сховищ, технології Drag'n'Drop і webworker.

Порівняно з попередньою версією (HTML4) додано дуже багато нових тегів і атрибутів, насамперед доцільно розглянути такі:

footer, header, nav, article – семантичні теги, що дозволяють зробити WEB-сторінки зрозумілішими для пошукових систем, браузерів та інших програм і пристроїв, що аналізують WEB-сторінки;

video і audio – теги, які покращують відтворення відео і аудіо;

тег canvas – для динамічного створення графіки на WEB-сторінках;

можливість відтворення графіки у форматі SVG;

нові значення type для <input> і безлічі нових атрибутів.

Елемент !DOCTYPE, який призначений для вказівки типу поточного документа і завжди розташовується в першому рядку, для документів, розмічених як HTML5, має вигляд:

<!DOCTYPE html>

Група семантичних тегів дозволяє структурувати контент, що розміщується на сторінці. Всі ці теги є аналогами тега , тобто є блоковими і не несуть спеціального форматування (для задання конкретного розташування і форматування повинні використовуватися специфікації CSS). Їх призначення зрозуміле з їх імен:

header – містить заголовок, передбачається розміщення у верхній частині сторінки;

footer – нижня частина сторінки;

nav – містить елементи навігації;

section – використовується для розміщення блоків тексту, наприклад, у декілька колонок;

article – можна використовувати для структуризації контенту за призначенням (наприклад, новини, статті, записи блога і тому подібне).

З урахуванням зазначеного, сторінка мовою HTML5 може мати таку структуру (pr1.htm):

```
<!DOCTYPE html>
<html>
<head>
<title>Приклад сторінки на HTML5</title>
<link rel="stylesheet" href="HTML5_CSS.css" type="text/css">
</head>
<body>
<header>
<h1>Цей тег h1 усередині header</h1>
</header>
<nav>
<a href="http://...">посилання 1 усередині nav</a>
<a href="http://...">посилання 2 усередині nav</a>
</nav>
<section class="n1">
<h1>Перша колонка</h1>
<article>
<h1>Цей тег h1 усередині section  усередині article</h1>
<p>Цей тег p усередині section  усередині article</p>
</article>
<article>
<h1>Цей тег h1 усередині section  усередині article</h1>
<p>Цей тег p усередині section  усередині article</p>
</article>
</section>
<section class="n2">
<h1>Друга колонка</h1>
</section>
<footer>
<p>p усередині footer</p>
</footer>
</body>
</html>
```

Якщо відкрити її у такому браузері, який навіть підтримує HTML5, то побачимо просто послідовність рядків тексту, ніби використовувалися звичайні теги div. Не потрібно дивуватися, оскільки для надання необхідного зовнішнього вигляду потрібні таблиці CSS. Але і без них структура документа під час аналізу його тексту стає зрозумілішою не тільки людині, але і пошуковим системам.

Щоб подивитися, як виглядає сторінка, яка відформатована з цими тегами, слід розмістити в контейнері заголовка (<head>... </head>) тег <link rel="stylesheet" href="HTML5_CSS.css" type="text/css">, а в папці зі сторінкою – файл HTML5_CSS.css, що містить параметри форматування (html5_css.css):

```
HTML, body {width: 90%;
margin:5%;
height:100%;
    }
header {background-color:rgb(188,192,192);
text-align:center;
min-height:10%;
margin-bottom:2%;}
nav    {min-height: 5%;
background: rgb(188,192,192);
text-align:center;
font-size:14pt;
margin-bottom:2%;}
section.n1 {float:left;
width: 80%;
background-color:rgb(188,192,192);
min-height:40%;
margin-bottom:2%;}
article {background-color:rgb(96,102,122);
margin:20px;}
section.n2 {float:right;
width: 18%;
min-height:40%;
background-color:rgb(188,192,192);}
footer  {clear:both;
background-color:rgb(188,192,192);
height:10%;
margin-top:2%;}
```

Інформацію про інші теги з цієї групи (aside, dialog, figure і т. д.) можна знайти в довіднику [9].

Для розміщення мультимедійного контенту у HTML4 використовувались теги embed або object. При цьому для його відтворення були потрібні плагіни, які окремо завантажуються для розширення можливостей браузерів. Наприклад, Flashplayer, VLCplayer, QuickTimeplayer.

HTML5 припускає початкове включення такого модуля в браузер з оптимізацією його роботи. А для вбудовування введено два теги video та audio. Передбачається, що в цьому випадку швидкість роботи значно вища, оскільки всі завдання виконуються на рівні браузера і не використовуються зовнішні елементи – плагіни.

Слід зазначити, що підтримка аудіо і відео в HTML5 дозволяє відтворювати не всі формати, немає перемикання в повноекранний режим.

Далі наведений текст сторінки, яка розмічена HTML5, з убудованими відео і аудіо, яка демонструє використання тегів і дозволяє протестувати роботу браузера (pr2.htm).

```
<!DOCTYPE html>
<html>
<head>
<title>Teri video, audio</title>
</head>
<body>
<video width="320" height="240" controls="controls">
<source src="f1.wmv">
</video>
<video src="f2l.ogv" controls="controls">
Ваш браузер не підтримує теги video! Обновіть версію браузера!
</video>
<audio src="f3.mp3" controls="controls">
Ваш браузер не підтримує теги audio! Обновіть версію браузера!
</audio>
<br/>
<video width="320" height="240" controls="controls">
<source src="f4.flv">
</video>
<video src="f5.mp4" controls="controls">
Ваш браузер не підтримує теги video! Обновіть версію браузера!
</video>
</body>
</html>
```

Треба звернути увагу, що атрибут controls указує на необхідність відображення панелі управління відтворенням. Указані як джерела файли різних форматів повинні розміщуватися в тій же теці, що і сама сторінка.

Для динамічного створення зображень використовується елемент canvas. Він є областю, в якій за допомогою JavaScript можна малювати геометричні об'єкти (лінії, дуги і тому подібне), виводити зображення, трансформувати їх і міняти властивості. При цьому зображення є растровим. Таким чином, можна створювати малюнки, анімацію, ігри і так далі.

Тег canvas парний. Текст, що міститься усередині, відображається тоді, коли браузер тег не підтримує. Основні атрибути: id, height і width (визначають висоту і ширину відповідно). За замовчуванням розмір полотна 150x300 пікселів. Елемент можна форматувати за допомогою таблиць стилів, наприклад, задати рамку.

<canvas id='exampl' height =200 width=300 style="border: 1px solid;">

Обновити браузер

</canvas>

Для можливості доступу до контексту малювання (вмісту "полотна") необхідно мати доступ до конкретного елемента (їх може бути декілька). Робиться це з використанням значення атрибута id. Наприклад:

canvas = document.getElementById('exampl');

Потім встановлюється режим малювання 2d або 3d (поки підтримується тільки 2d), і запам'ятовується посилання, яке буде використане для доступу до вмісту елемента, наприклад:

can = canvas.getContext('2d');

Таких посилань (контекстів) може бути декілька. Часто це роблять з перевіркою підтримки відповідного режиму:

var canvas = document.getElementById('exampl');

if (canvas.getContext){varcan = canvas.getContext('2d');

…

Саме малювання (створення зображення) здійснюється шляхом виклику функцій API. Для кожної фігури передбачена своя функція (табл. 1). Повний список функцій істотно більший, їх можна знайти в довіднику [10].

Усі функції викликаються як методи контексту (can.func();). Аргументами функцій, що викликаються, є координати й інші параметри зображень (розміри і тому подібне). Початок системи координат поміщений у лівий верхній кут області (позитивний напрям осі Y– вниз, X – управо).

Таблиця 1

Графічні примітиви

Фігура	Функція
Прямокутники	
прямокутник	strokeRect(x, y, ширина, висота)
прямокутник із заливкою	fillRect(x, y, ширина, висота)
очищення області	clearRect(x, y, ширина, висота)
Лінії і дуги	
зміна поточної позиції	moveTo(x, y)
лінія з поточної позиції до крапки	lineTo(x, y)
дуга	arc(x, y, rad, startAng, endAng, anticlockwise)
крива Безьє 2-го порядку	quadraticCurveTo(P1x, P1y, P2x, P2y)
крива Безьє 3-го порядку	bezierCurveTo(P1x, P1y, P2x, P2y, P3x, P3y)

У ході використання наведених примітивів прямокутники виводяться відразу після виконання функцій. Колір заливки і ліній визначаються поточними значеннями, що зберігаються як властивості контенту. Зміна значення кольору ліній, заливки і товщини ліній:

```
can.fillStyle = '#00f';
can.strokeStyle = '#f00';
can.lineWidth = 4;
```

Створення довільних фігур із ліній, кривих і дуг проводиться в два етапи. На початку послідовністю викликів функцій задається контур майбутньої фігури. Потім проводиться промальовування контуру або його заливка.

Задавання контуру починається з виклику функції beginPath(), а потім викликаються функції всіх примітивів, які створюють фігуру. У разі необхідності з'єднати початкову і кінцеву точки (для отримання замкнутого контуру) викликається closePath ().

Для промальовування фігури, що вийшла, викликається stroke () або fill(). Перша промальовує контур поточним кольором і товщиною, друга заливає фігуру поточним кольором.

Для виведення тексту використовується функція fillText(st,x,y), рядок st буде виведений в точку x,y поточним кольором. Поточні характеристики шрифту задаються як властивість контексту за правилами CSS.

Окремо задається вирівнювання (властивість textAlign) і зсув щодо базової лінії (властивість textBaseline). Наприклад:

```
can.font='bold 2em "sans-serif"';
can.textAlign="center";
can.textBaseline="top";
can.fillText('Успіхи!',110,180);
```

Приклад використання наведених функцій (pr4.htm):

```
<!DOCTYPE html>
<html>
<head>
<title>Canvas</title>
</head>
<body>
<canvas id="canva" width="220" height="220" style="border: 1px solid black;">
Ваш браузер не підтримує цю можливість
</canvas>
<script>
var canvas = document.getElementById('canva');
if (canvas.getContext){var can = canvas.getContext('2d');
can.lineWidth = 2;          //товщина лінії
can.strokeRect(40,40, 140,140);       //малювання прямокутника
can.beginPath();            //початок створення контуру
can.moveTo(110, 50);        //задавання поточного положення
can.lineTo(170, 170)        //лінія
can.lineTo(50, 170);        //лінія
can.closePath();            //завершення контуру
can.fillStyle ='#00f';      //колір заливки синій
can.strokeStyle ='#f00';    //колір ліній червоний
can.lineWidth = 8;          //товщина ліній
can.stroke();               //малювання контуру
can.fill();                 //заливка фігури
can.font='bold 2em "sans-serif"';
can.textAlign="center";
can.textBaseline="top";
can.fillText('Успіхи!',110,180);  //виведення тексту
}
</script>
</body>
</html>
```

Вид зображення у вікні браузера наведений на рис. 1.

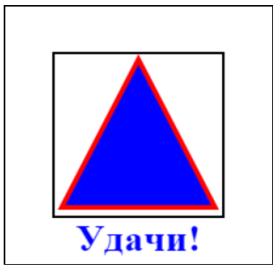

Рис. 1. **Зображення у вікні браузера**

Окрім малювання фігур і тексту на полотно можуть виводитися зображення, захоплені з різних джерел, наприклад, зображення з тегів img, video, інших елементів canvas.

Захоплення і малювання відбуваються в методі drawImage, який працює з різними джерелами.

У тому випадку, коли джерелом є зображення у файлі, спочатку створюється графічний об'єкт, який зв'язується з файлом, що містить зображення.

varimg = newImage(); // Створення нового об'єкта зображення,

img.src = 'image.png'; // пов'язання з файлом.

Для того, щоб до початку малювання зображення з файла встигло завантажитися, виклик функції drawImage розміщується в обробнику події, пов'язаної із закінченням завантаження зображення.

img.onload = function() { ctx.drawImage(img, 0, 0);}

Функція drawImage виконує три варіанти дій залежно від варіанта задавання аргументів.

По-перше, drawImage(image, dx, dy) вставляє зображення в його дійсному розмірі, поміщаючи лівий верхній кут у крапку з координатами dx, dy.

По-друге, drawImage(image, dx, dy, dw, dh) вставляє зображення в прямокутник, з кутом у точці dx, dy, висотою dh і шириною dw, масштабуючи його за кожною віссю.

По-третє, drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh) вирізає з початкового зображення ділянку, починаючи з точки sx, sy розміром sw, sh, і вставляє її з масштабуванням у прямокутник з розмірами dw, dh, поміщаючи кут у точку dx, dy.

Наприклад, додавши в кінець скрипта попереднього прикладу наведений далі код (pr5.htm), можна побачити у вікні таке зображення (рис. 2).

```
varim = new Image();
im.src = "im1.jpg";
im.onload = function() { can.drawImage(im, 600, 500,100,100,60,60,100,100); };
```

Рис. 2. **Зображення у вікні браузера**

Під час виведенні елементів на полотно вони можуть піддаватися змінам (перетворенням). Для цього використовуються методи translate, scale і rotate.

Метод translate(dx, dy) переміщає об'єкти (фігури, зображення), що виводяться на полотні, в іншу точку координатної сітки. Значення dx,dy позначають, відповідно, кількість пікселів, на яку буде переміщено зображення у напрямі X і Y.

Метод scale(kx,ky) змінює розміри об'єктів, що виводяться. Аргумент kx задає коефіцієнт масштабування по горизонталі, а аргумент ky задає коефіцієнт масштабування по вертикалі.

Метод rotate(angle) повертає об'єкт, що виводиться, на заданий кут в радіанах.

Після виклику методу перетворення, яке він виконує, стає властивістю контексту, і всі об'єкти, що виводяться після цього, будуть схильні до перетворення. Для продовження нормального виведення об'єктів необхідно відновити попередній стан контексту. Збереження і відновлення стану здійснюється за допомогою функцій save і restore, відповідно. У загальному випадку може бути рекомендована така послідовність дій.

```
varcanvas = document.getElementById('canva');
var context = canvas.getContext('2d');
context.save();      //збереження контексту
context.translate(70, 140);       //задавання перетворення
context.beginPath();      //виведення об'єктів з перетворенням
```

```
context.moveTo(0, 0);
context.lineTo(70 -70);
context.stroke();
context.restore();  //відновлення контексту
…      //виведення об'єктів без перетворення
```

Під час перетворення об'єктів методом rotate(ang) відбувається їх обертання навколо початку координат (за замовчуванням крапка 0,0), кут (ang) задається в радіанах. Для здійснення обертання навколо іншого центру (наприклад, власного центру симетрії об'єкта) необхідно змістити початок координат за допомогою методу translate(x,y) в потрібну крапку. Приклад (pr6.htm) і зображення у вікні браузера (рис. 3) ілюструють ці дії.

```
<!DOCTYPE html>
<html>
<head>
<title>Canvas</title>
</head>
<body>
<canvas id="canva" width="220" height="220" style="border: 1px solid black;">
Ваш браузер не підтримує цю можливість
</canvas>
<script>
var canvas = document.getElementById('canva');
if (canvas.getContext){var can = canvas.getContext('2d');
can.fillStyle ='#00f';
can.strokeStyle ='#f00';
can.lineWidth = 2;
can.strokeRect(40,40, 140,140);
can.font='bold 2em "sans-serif"';
can.textAlign="center";
can.textBaseline="top";
can.save();
can.translate(110,110);
can.rotate(1.0);
can.fillText('Удачи!',0,0);
can.restore();
can.fillText('Удачи!',110,180);
}
</script>
</body>
</html>
```

Рис. 3. **Зображення у вікні браузера**

Слід зазначити, що для спрощення у всіх наведених прикладах використані скрипти, що виконуються браузером прямо в тексті сторінки. Проте найчастіше клієнтські скрипти робляться обробниками різних подій, наприклад, клацання лівої клавіші миші на якому-небудь елементі (onClick) і тому подібне.

Важливим кроком у розвитку технології мов розмітки було включення в стандарт HTML5 можливості з відображення векторної графіки. Як формат був вибраний SVG (від англ. ScalableVectorGraphics) – мова для опису двовимірної графіки у форматі XML. Зображення може складатися з ліній, зображень і тексту. Основною перевагою формату є здатність змінювати розміри без втрати якості зображення. Крім того, для SVG є достатньо розвинені засоби анімації.

Код SVG може відображатися в браузері з файла з використанням тегів IMG, OBJECT, EMBED, IFRAME або вбудовуватися безпосередньо в текст сторінки за допомогою тега SVG.

Оскільки зображення зберігається у вигляді розміченого тексту, створювати його можна в звичайному текстовому редакторові, хоча можна використовувати редактори векторної графіки або спеціальні редактори SVG.

Під час розміщення документа у файлі він повинен починатися з визначення xml і !DOCTYPE, потім йде тег <svg> з атрибутами, що визначають простори імен. Файл повинен мати розширення svg, в цілому вміст файла виглядає так:

```
<?xml version="1.0"?>
<!DOCTYPEsvg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svgxmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
. ..      //тут розміщуються теги, які задають зображення
</svg>
```

У процесі розміщенні зображення в тексті сторінці визначення xml і !DOCTYPE не потрібні.

Тег <svg>. Елемент <svg> є основним контейнером, що містить всі інші елементи. На додаток до атрибутів, що визначають простір імен, можуть використовуватись width, height, preserveAspectRatio і viewBox – вони визначають область перегляду і полотно для створюваної графіки. Область перегляду є видимою частиною полотна SVG. Розмір області перегляду встановлюється атрибутами height і width.

Атрибут viewBox задає область, до якої повинен масштабуватися набір графічних елементів, що потрапляють в область перегляду. Його значеннями є чотири числа (координати кута, ширина і висота), найчастіше вони збігаються зі значеннями меж області перегляду. За відсутності цього атрибуту зображення не масштабуватиметься.

<svgwidth="115px" height="190px" viewBox="0 0 65 140">

<!--опис зображення -->

</svg>

Зображення може бути схильне до трансформацій (обертання, масштабування, переміщення і нахил). Трансформації можуть застосовуватися до окремих елементів або до групи елементів. Відповідні функції для перетворення задаються як значення атрибуту transform у керованому елементі:

transform="translate(tx,ty) rotate(кут)";

translate задає зрушення елемента;

rotate задає поворот фігури навколо її початкової точки (у градусах), також є можливість додати значення x і y для зсуву центру обертання;

transform=rotate(кут [cx,cy]);

scale дозволяє змінювати розміри елемента відповідно до значень горизонтального і вертикального масштабу, при цьому змінюється масштаб координат, а не розміри об'єкта;

transform="scale(kx[ky])".

Значення ky є необов'язковим, і якщо воно не вказане, то передбачається, що воно рівне kx, що гарантує пропорційну зміну масштабу зображення.

Тег<g> є контейнером для угрупування пов'язаних графічних елементів. Це особливо зручно у ході анімації елементів або їх трансформації, оскільки можна анімувати або трансформувати всю групу. Будь-який елемент, який не міститься всередині g, розглядається як група з нього одного.

Те г<use> дозволяє повторно використовувати елементи в будь-якому місці документа. Тег може містити такі атрибути, як x,y,width і height, які визначають подробиці положення елемента в системі координат. Атрибут xlink:href дозволяє звернутися до елемента, щоб використовувати його повторно. Наприклад, якщо є елемент g, що містить деяке зображення, <g id="apple">...</g>, то це зображення можна розмістити повторно, <usex="50" y="50" xlink:href="#apple" />.

Порядок накладення під час зображення елементів повністю залежить від їх розміщення усередині фрагмента документа. Елемент, опис якого поміщений пізніше, розташовується вище. Атрибут z-index не діє.

SVG містить наступний набір основних фігур: прямокутник, круг, еліпс, прямі лінії, ламані лінії і багатокутники. Їх параметри задаються за допомогою атрибутів.

Тег <rect> визначає прямокутник.

<svg>

<rectx="10" y="20" width="200" height="100" fill="#BBC43A" />

</svg>

Атрибути width і height встановлюють розмір прямокутника, fill – внутрішній колір фігури, x,y – координати.

Тег <circle> задає круг.

<svg>

<circle cx="75" cy="75" r="75" fill="#ED6E46" />

</svg>

Тег <line> визначає пряму лінію з початковою і кінцевою точкою.

<svg>

<line x1="5" y1="5" x2="100" y2="100" stroke="#765373" stroke-width="8"/>

</svg>

Значення атрибутів stroke і stroke-width задають колір і товщину.

Тег <polyline> визначає набір сполучених відрізків прямої лінії.

<svg>

<polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" stroke="#BBC42A" stroke-width="6" />

</svg>

Тег <polygon> визначає замкнуту фігуру, що складається зі зв'язаних прямих ліній.

<svg>

<polygon points="50,5 100,5 125,30 125,80 100,105 50,105 25,80 25,30" fill="#ED6E46" />

</svg>

Тег <path> є контуром фігури. Ця фігура може бути залита кольором, обведена, використана як напрямна для тексту або як контур обрізання. Контур будується з різних ліній (прямих, дуг еліпсів, кривих Безьє). Дані про лінії і їх координати містяться в значенні атрибута d. Деталі правил завдання контуру можна подивитися в довіднику [11].

Тег <text> визначає графіку, що складається з тексту. Основними атрибутами тегу є x, y, fill, font-size, font-family, призначення і сенс яких очевидний.

<svg width="620" height="100">

<text x="30" y="90" fill="#ED6E46" font-size="100" font-family="'Leckerli One', cursive">Watermelon</text>

</svg>

Текст може розташовуватися не тільки по прямій лінії, але й описувати деяку траєкторію відповідно до тієї, що спрямовує текст, заданою елементом path.

```
<svg>
<defs>
<path id="testPath" d="<....>"/>
</defs>
<text>
<textPathxlink:href="#testPath">Place text here</textPath>
</text>
</svg>
```

У даному прикладі елемент path задає траєкторію. Тег defs використаний, щоб крива була невидимою. Усередині тега text поміщений спеціальний елемент textPath, пов'язаний посиланням з елементом path (xlink:href="#testPath"). У результаті текст буде розміщений уподовж кривої.

Приклад, що демонструє описані можливості (pr7.svg):

```
<?xmlversion="1.0"?>
<!DOCTYPEsvg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="1200" height="400" viewBox="0 0 1200 400"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
```

```
version="1.1">
<rect x="0" y="0" width="100" height="200" fill="green"
transform="translate(300,100) rotate(10,400,100) "/>
<g transform="translate(200,220) ">
<defs>
<path id="testPath" d="M3.858,58.607 c16.784-5.985,33.921-10.518,51.695-
12.99c50.522-7.028,101.982,0.51,151.892,8.283c17.83,2.777,35.632,5.711,53.437,8.628
c51.69,8.469,103.241,11.438,155.3,3.794c53.714-7.887,106.383-20.968,159.374-
32.228c11.166-2.373,27.644-7.155,39.231-4.449" />
</defs>
<text fill="#000000" font-size="30" font-family="sans-serif">
<textPathxlink:href="#testPath">rectangle rotated 10 degrees</textPath>
</text>
</g>
</svg>
```

Сторінка для виведення малюнка (файл pr7.htm).

```
<!DOCTYPE html>
<html>
<head>
<title>SVG</title>
</head>
<body>
<imgsrc=pr7.svg />
</body>
</html>
```

Вид зображення у вікні браузера наведений на рис. 4.

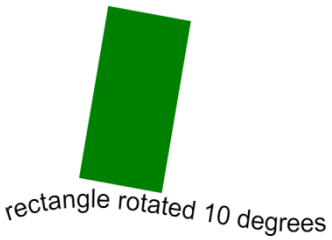

Рис. 4. **Вид малюнка у вікні браузера**

Форма – це елемент HTML-документа, що забезпечує відправку даних серверу. Форми використовуються для опитування відвідувачів, за-

мовлення товарів та інших дій. Основними елементами, що розміщуються на формі, є різні текстові поля, перемикачі і кнопки. Більшість із них створюється за допомогою тега <**INPUT>** з різними атрибутами [2; 3]. У HTML5 істотно розширений перелік таких елементів. У табл. 2 наведені атрибути тега <**INPUT>** їх значення, яких не було в попередній версії.

Таблиця 2

Атрибути тега <INPUT>

Ім'я атрибута	Значення	Опис
1	2	3
type	color,date, datetime, datetime-local, email, image, month, number, range, search, tel, time, url, week	Указує тип елемента введення
autocomplete	autocomplete	Активує можливість автозаповнення поля
autofocus	autofocus	Указує, що даний елемент має бути активним відразу після завантаження сторінки
form	ідентифікатор форми	Указує одну або декілька форм, до яких належить даний елемент
formaction	URL	Указує адресу, на яку повинні відсилатися дані форми (тільки для type="submit" і type="image")
formenctype	application/x-www-form-urlencoded multipart/form-data text/plain	Указує, як повинні кодуватися дані форми перед відправкою (тільки для type="submit" і type="image")
formmethod	GET POST	Указує спосіб відправки даних форми (тільки для type="submit" і type="image")
formnovalidate	formnovalidate	Указує, що дані форми не повинні перевірятися перед відправкою

Закінчення табл. 2

1	2	3
formtarget	_blank _self _parent _top имяфрейма	Указує, де повинна відображатися відповідь сервера після відправки форми (тільки для type="submit" і type="image")
height	пікселі	Встановлює висоту елемента (тільки для type="image")
list	ідентифікатор елемента datalist	Дозволяє прив'язати елемент datalist до елемента <input>
max	число дата	Указує максимальне число або дату, яку можна ввести в поле введення
min	числодата	Указує мінімальне число або дату, яку можна ввести в поле введення
multiple	multiple	Указує, що в дане поле можна ввести декілька значень
pattern	регулярний вираз	Задає регулярний вираз, з яким має бути звірене значення елемента <input> перед відправкою
placeholder	текст	Встановлює текст-підказку, який відображатиметься в елементі <input> поки він неактивний
required	required	Указує, що дане поле обов'язково має бути заповнене перед відправленням форми
step	число	Встановлює ширину кроку для елемента <input>
width	пікселі	Встановлює ширину елемента (тільки для type="image")

Завдяки цим нововведенням форми стали привабливішими, крім того, істотно спрощується перевірка правильності введення (валидация) даних на стороні клієнта. Зважаючи на велику кількість різних атрибутів, рекомендується вибрати необхідні для свого прикладу, і скористатися довідником [10] для їх правильного використання.

Література: основна [2; 3]; ресурси мережі Інтернет [7; 9 – 11].

Тема 3. Використання стильових специфікацій

Самостійна робота № 3. Нові можливості версії CSS3

Мета роботи: вивчення нових можливостей технології таблиць стилів, набуття навичок їх застосування.

У результаті виконання самостійної роботи у студента формуються такі **компетентності:** здатність самостійно вивчати нові технології, створювати сторінки для мережі Інтернет з використанням засобів CSS3.

Результатом виконання самостійної роботи є звіт з виконання завдання та створені під час виконання кожного пункту завдання сторінки.

Завдання для самостійної роботи

1. Вивчити довідкові матеріали до самостійної роботи.
2. Виконати всі приклади, наведені у довідкових матеріалах.
3. Створити сторінку, що містить блоки із закругленими кутами, тінню і нестандартним шрифтом.
4. Створити сторінку для перевірки можливостей управління фоном (кілька зображень, градієнт тощо).
5. Створити власну анімацію для сторінки, використовуючи властивості CSS3.

Контрольні запитання для самодіагностики

1. Що таке вендорні префікси?
2. Запишіть селектор для вибору елементів, що містять атрибут src.
3. Запишіть і поясніть варіанти задавання значень властивості border-radius.
4. Чи можна створити текст із тінню? Як це зробити?
5. Якою властивістю визначається прозорість елемента? Запишіть приклад.
6. Які нові можливості для створення фону елементів містить специфікація CSS3?
7. Чим відрізняється анімація і трансформації в CSS3?

Методичні рекомендації

Новий стандарт CSS3 передбачає істотне розширення можливостей для форматування елементів WEB-сторінок. Серед цих нововведень доцільно звернути увагу на такі:

нові селектори;
закруглені рамки у блоків;
створення тіні до елементів;
можливість задати свій шрифт;
можливість задати прозорий колір;
лінійні і сферичні градієнти;
нові можливості по роботі з фоном;
трансформації елементів;
анімація.

Проте, в даний час браузери підтримують ці можливості різною мірою, крім того склалося так, що в різних браузерах синтаксис правил дещо відрізняється. Тому було запропоновано використовувати перед записом правил спеціальний покажчик (вендорний префікс), що визначає браузер, для якого це правило призначене. Таким чином, для кожного браузера можна використовувати свій синтаксис. Вендорні префікси для популярних браузерів подані в табл. 3.

Таблиця 3

Вендорні префікси

Префікс	Виробник	Браузер	Браузерний движок
-o-, -op- -xv-	OperaSoftware	Opera	Presto
-moz-	проект Mozilla	Firefox, SeaMonkey, Camino та ін.	Gecko
-ms-	Microsoft	InternetExplorer 8	Trident
-khtml-	проект KDE	Safari до версії 3, Konqueror та ін.	KHTML, слугує основою для WebKit
-webkit-	Apple	Safari 3+, Google Chrome та ін.	WebKit

Наприклад, задавання властивості прозорості елемента, у разі чого буде забезпечена кросплатформність, може виглядати так:

```
-ms-filter:"progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";  /* IE 8*/
-moz-opacity:0.5;        /* Mozilla 1.6 */
-khtml-opacity:0.5;      /* Konqueror 3.1, Safari 1.1 */
```

Слід зазначити, що у міру повнішого обліку стандарту в нових версіях браузерів, необхідність у використанні префіксів відпаде.

У ході вивчення форматування із застосуванням таблиць стилів найчастіше розглядається обмежений набір селекторів: класові, id, контекстні, декілька псевдокласів. Зазвичай їх використання цілком достатньо. Проте слід мати на увазі, що селектори – це могутній апарат, що дозволяє створювати складні конструкції, що визначають вибір елементів. У табл. 4 наведені селектори, які не входять в мінімальний набір і які можна використовувати для форматування за допомогою таблиць стилів.

Таблиця 4

Розширені селектори CSS

Селектор	Приклад	Опис
1	2	3
:not(x)	*:not(div)*	Вибираються всі елементи, окрім елементів div
[атрибут]	p[id]	Вибираються всі абзаци, які мають атрибут id
::selection	*::selection*	Виділений користувачем текст
[атрибут=значение]	p[id="el1"]	Вибираються всі абзаци, які мають атрибут id зі значенням el1
[атрибут~=значение]	a[href~="wisdomweb]"	Вибираються всі посилання з атрибутом href, що містить у значенні підрядок "wisdomweb", відокремлений пропусками від решти вмісту
[атрибут^=значение]	*[src^="http://"]*	Вибираються всі елементи, що мають атрибут src зі значенням, що починається на "http://"
[атрибут$=значение]	*[src$=".gif"]*	Вибираються всі елементи, що мають атрибут src зі значенням, що закінчується на ".gif"
[атрибут=*значение]	*[src*="picture"]*	Вибираються всі елементи, що мають атрибут src зі значенням що містить підрядок "picture"
:first-child	p:first-child	Вибираються всі абзаци, які є першими в батьківському елементі
:last-child	*div:last-child*	Вибираються всі елементи div, що є останніми елементами-нащадками в батьківському
эл1> эл2	div> p	Вибираються всі абзаци, що є нащадками елемента div
эл1 + эл2	div + p	Вибираються всі абзаци, наступні після елемента div
элемент1~элемент2	*div~p*	Вибираються всі елементи div, що знаходяться перед елементом p
:before	p:before	Вставляється довільний вміст перед елементом p

Закінчення табл. 4

1	2	3
:after	p:after	Вставляється довільний вміст після елемента p
:focus	input:focus	Вибираються всі активні елементи введення на сторінці
:enabled	*:enabled*	Вибираються всі працездатні елементи введення
:disabled	*:disabled*	Вибираються всі непрацездатні елементи введення
:first-of-type	*div:first-of-type*	Вибираються всі елементи div, що є першими в батьківському
:last-of-type	*div:last-of-type*	Вибираються всі елементи div що є останніми в батьківському
:only-of-type	*div:only-of-type*	Вибираються всі елементи div, які є унікальними батьківському
:nth-child(x)	*div:nth-child(3)*	Вибираються всі елементи div що є третіми по рахунку в батьківському
:nth-last-child(x)	*div:nth-last-child(2)*	Вибираються всі елементи div що є другими елементами нащадками в батьківському з кінця
:root	*:root*	Вибирається кореневий елемент документа
:empty	*p:empty*	Вибираються порожні абзаци

Ці селектори роблять механізм вибору елементів незвичайно гнучким.

Часто в оформленні елементів сайта використовуються закруглені кути у різних елементів. Для їх створення раніше застосовувалися фонові малюнки. CSS 3 містить властивість border-radius, яка дозволяє закругляти кути будь-якого блоку. Параметрами є радіуси закруглення, може використовуватися від одного до чотирьох значень, розділених пропусками.

1 – радіус указується для всіх чотирьох кутів.

2 – перше значення задає радіус верхнього лівого і нижнього правого кута, друге значення – верхнього правого і нижнього лівого кута.

3 – перше значення задає радіус для верхнього лівого кута, друге – одночасно для верхнього правого і нижнього лівого, а третє – для нижнього правого кута.

4 – по черзі встановлює радіус для верхнього лівого, верхнього правого, нижнього правого і нижнього лівого кута.

Як значення указуються числа в будь-якому допустимому для CSS форматі. У разі застосування відсотків, відлік ведеться щодо ширини блоку.

Наприклад: border-radius: 50px 55px60px65px.

Окрім закруглених кутів, блоки можуть мати тінь, для її задавання використовується властивість box-shadow. Тінь також може бути у тексті text-shadow. У властивості може бути 4 параметри, які розділяються пропусками: зрушення тіні по осі X, зрушення тіні по осі Y, розмиття тіні, колір тіні. Допускається використовувати декілька тіней, указуючи їх параметри через кому, у разі накладення тіней перша тінь у списку буде вища, остання нижче. Якщо для елемента задається радіус закруглення через властивість border-radius, то тінь також вийде із закругленими куточками. Додавання тіні збільшує ширину елемента, тому можлива поява горизонтальної смуги прокрутки в браузері.

Наприклад,

-webkit-box-shadow: inset 0px 3px 20px 3px #f00, 3px 10px black;

Під час відображення тексту браузер використовує встановлені в даній операційній системі шрифти. У разі необхідності використання нестандартних шрифтів для сайта, вони мають бути заздалегідь завантажені на комп'ютер користувача. CSS 3 надає таку можливість. Правило @font-face слугує для завантаження шрифтів, необхідних для сайта. Усередині конструкції @font-face поміщаються декларації, що визначають ім'я шрифту, файл шрифту та ін. Зрозуміло, вибраний шрифт має бути розміщеній і доступний на сервері.

Наприклад:

```
@font-face {
font-family: Pompadur;         /* Імяшріфта */
src: url(fonts/pompadur.ttf);  /* Путькфайлусошріфтом */
}
```

Далі наведений приклад сторінки, на якій використовуються закруглені кути, тіні і нестандартний шрифт (pr8.htm).

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>@font-face</title>
<style>
.t {
background: #f0f0f0;       /* колір фону */
border: 1px solid black;  /* Параметри рамки */
padding: 15px;       /* Поля навколо тексту */
margin-bottom: 10px;      /* Отступ знизу */
box-shadow: inset  0px 3px 20px 3px  #f00, 3px 10px ; /* Тінь */
      }
 @font-face {
```

```
font-family: my_font;       /* Ім'я шрифту */
src: url(my.ttf);      /* Шлях до файла зі шрифтом */
}
   P {font-family: my_font;}
   #d1 {border-radius: 50px 0 0 50px;}
   #d2 {border-radius: 20em 10em 5em;}
   #d3 {border-radius: 40px 10px}
   #d4 {border-radius: 8px;}
</style>
</head>
<body>
<div id="d1" class="t" >
<p>border-radius: 50px 0 0 50px;</p>
</div>
<div id="d2" class="t" >
<p>border-radius: 20em 10em 5em;</p>
</div>
<div id="d3" class="t" >
<p>border-radius: 40px 10px;</p>
</div>
<div id="d4" class="t">
<p>border-radius: 8px;</p>
</div>
</body>
</html>
```

Прозорі елементи використовуються на сторінках для створення різних ефектів. Стандарт CSS3 істотно спростив це завдання. Будь-якому елементу на WEB-сторінці можна встановити прозорість за допомогою властивості opacity з параметром від 0 (невидимий) до 1 (початкова прозорість, встановлений колір). Наприклад, opacity: 0.5 зробить елемент напівпрозорим. При цьому прозорість міняється і для всіх дочірніх елементів.

CSS3 пропонує ряд нових можливостей для управління фоном об'єктів. З'явилася можливість одночасно використовувати відразу декілька фонових зображень для одного і того ж елемента. Робиться це за допомогою background-image з тією лише відмінністю, що тепер можна вказувати не одну, а відразу декілька адрес до різних картинок перераховуючи їх через кому:

background-image: url(fon1.jpg), url(fon2.jpg), url(fon3.jpg).

Те зображення, яке йде в списку першим, буде знаходитися поверх інших, останні ж підкладатимуться під нього в тому порядку, в якому вони перераховані. Під час використання як фону декількох зображень, до них так само можна застосовувати додаткові властивості: background-attachment, background-repeat, background-position і так далі. Значення

для цих властивостей, у випадку якщо картинок декілька, так само вказується через кому, перше значення призначене для першого фону друге для другого і так далі.

За допомогою нової властивості background-size можна вказувати розмір зображення, яке служить фоном. Значеннями властивості можуть бути:

ширина і висота фонової картинки у px або % (зображення буде підігнано під цей розмір);

cover – фонове зображення буде розтягнуто на весь блок;

contain – фонове зображення заповнить блок зі збереженням пропорцій (до тих пір, поки не упреться в краї блоку);

auto – початковий розмір зображення (за замовчуванням).

Наприклад:

```
background-size: 200px 50px;
background-size: 20% 80%;
background-size: contain;
background-size: cover;
background-size: auto 50px;
background-size: auto auto;
```

За допомогою нової властивості background-origin можна визначати область позиціонування фону в тому або іншому блоці. Можливі значення такі:

padding-box – фон позиціонується від країв блоку;

border-box – фон позиціонується від меж блоку;

content-box – фон позиціонується від полів блоку.

Властивість background-origin не застосовується, якщо значення background-attachment задане як fixed.

Крім того, CSS3 дозволяє створювати як фон градієнти безпосередньо на сторінці без необхідності створювати їх зображення в текстовому редакторові. Градієнт задається за допомогою методу linear-gradient (лінійний градієнт) або radial-gradient (сферичний градієнт) властивості background-image. Аргументами є характеристики кольору і координати, які задають напрям і щільність. Далі наведені приклади правил, які задають для елементів різні градієнти.

```
.c1 {filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#cccccc',
endColorstr='#000000'); /* для IE */
background-image: -webkit-gradient(linear, left top, left bottom, from(#ccc),
to(#000)); /* для webkit */
```

```
background-image: -moz-linear-gradient(top,  #ccc,  #000); /* для firefox 3.6+ */}
.c2 {background-image: -webkit-gradient(linear, 30% 30%, 70% 70%, from(#ccc),
to(#000)); /* для webkit */}
.c3 {background-image: -moz-radial-gradient(center, ellipse cover, #feffff 0%,
#d2ebf9 100%);
background-image: -webkit-radial-gradient(center, ellipse cover, #feffff 0%,#d2ebf9
100%);
background-image: -o-radial-gradient(center, ellipse cover, #feffff 0%,#d2ebf9
100%);
background-image: -ms-radial-gradient(center, ellipse cover, #feffff 0%,#d2ebf9
100%);
background-image: radial-gradient(center, ellipse cover, #feffff 0%,#d2ebf9 100%);
}
```

У CSS3 є можливість зміни зовнішнього вигляду елементів шляхом їх трансформації на зразок того, як це робилося із зображенням на полотні canvas. Передбачені такі види трансформуючих перетворень: зрушення, масштабування, поворот, нахил, а також їх комбінації. Для цього передбачена властивість transform, значенням якої є функція перетворення.

Функція translate(tx,ty) зрушує елемент на задане значення по горизонталі і вертикалі.

Функція scale(sx,sy) змінює масштаб елемента по горизонталі і вертикалі. Значення більше 1 збільшує масштаб елемента, менше 1 – зменшує масштаб.

Функція rotate(ang) задає поворот елемента на заданий кут щодо точки обертання. Точка обертання задається властивістю transform-origin (transform-origin: x y;). Кут задається в градусах або радіанах (наприклад, 90deg або 1.57rad).

Функції skewx(ang), skewY(ang) спотворюють форму елемента шляхом нахилу на вказаний кут щодо однієї з осей. Кут нахилу задається в таких же одиницях вимірювання, як і для трансформації повороту.

Для послідовного виконання декількох трансформацій необхідно вказати їх в одному правилі, розділивши пробілами.

Виконання трансформації пов'язується з деякою подією за допомогою скрипта або псевдокласу (наприклад, :hover). У разі настання події зміни відбуваються миттєво, проте їх можна перетворити на анімацію, якщо задати властивість transition. Далі наведений приклад сторінки (pr10.htm), що демонструє трансформації.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```
<title>CSS3 Transform</title>
<style>
h1  {margin: 50px;text-align: center;color: red;border-bottom: 1px solid red;}
p   {font-size: 1.1em;color: #ffffff;}
p.primer {color: #e30000;font-size:24pt}
div {position:absolute;left:400px;width:300px;height:200px;background:#999999;}
div.primer {
transition:all 1s ease-in-out;
    -webkit-transition:all 1s ease-in-out;
     -moz-transition:all 1s ease-in-out;
          -o-transition:all 1s ease-in-out;
}
div.primer:hover {
transform: translate(50px,50px) scale(1.5,1.5) rotate(45deg);
   -moz-transform: translate(50px,50px) scale(1.5,1.5) rotate(45deg);
-webkit-transform: translate(50px,50px) scale(1.5,1.5) rotate(45deg);
   -o-transform: translate(50px,50px) scale(1.5,1.5) rotate(45deg);
}
</style>
</head>
<body>
<h1>Демонстрація можливостей властивості CSS3 Transform</h1>
        <div class="primer" align="center">
                <p class="primer">Приклад</p>
                <p>translate, scale, rotate + transition</p>
        </div>

</body>
</html>
```

CSS3 дозволяють створювати і складніші види анімації за допомогою ключових кадрів, які містять інформацію про послідовність значень змінних параметрів. Правило @keyframes дозволяє вказати значення, які мають бути у властивості CSS в різні моменти анімації.

Наприклад:

```
@keyframesfadeOutInOut {
0% {
opacity: 1;
  }
25% {
opacity: 0;
  }
75% {
opacity: 1;
  }
100% {
opacity: 0;
  } }
```

Для управління анімацією (тривалість, напрям, затримки і тому подібне) слугує узагальнена властивість animation:

animation-name вказує одне або декілька імен анімації для ідентифікації правило CSS @keyframes;

animation-delay вказує затримку анімації (час з початку циклу) до відображення анімації в секундах (наприклад, animation-delay: 2s;);

animation-direction вказує напрям відтворення циклу анімації, "normal", "reverse", "alternate" і "alternate-reverse";

animation-duration вказує тривалість одного циклу анімації в секундах (наприклад, animation-duration: 2s;);

є і інші властивості.

Сторінка pr11.htm демонструє використання анімації для переміщення елемента, який в кінці розчиняється:

```
<!DOCTYPE html>
<html>
<head>
<title>анимация</title>
<style type="text/css">
div {position: relative; top: 50px;left: 0px;width:200px;background-color:red;
        -webkit-animation-name: 'movement';
        -webkit-animation-duration: 10s;}
  @-webkit-keyframes 'movement' {
        0% {
      top: 50px;
      left: 0px;
opacity: 1;
        }
        50% {
      top: 150px;
      left: 100px;
opacity: 1;
        }
        100% {
      top: 50px;
      left: 300px;
opacity: 0;
        }
      }

</style>
</head>
<body>
<div>блок повинен рухатися, а в кінці зникнути</div>
</body>
</html>
```

Література: основна [2, 3]; додаткова [6].

Змістовий модуль 2. Програмування для WEB

Тема 4. Сценарії, що виконуються на клієнтській стороні

Самостійна робота № 4. Використання бібліотек

Мета роботи: ознайомитися з можливостями бібліотеки jQuery і основами її застосування на WEB-сторінках.

У результаті виконання самостійної роботи у студента формуються такі **компетентності:** здатність до вибору засобів та розробки WEB-ресурсів із використанням бібліотек і фреймворків.

Результатом виконання самостійної роботи є звіт із виконання завдання та створені під час виконання кожного пункту завдання документи.

Завдання для самостійної роботи

1. Вивчити довідкові матеріали до самостійної роботи.
2. Виконати всі приклади, наведені у довідкових матеріалах.
3. Створити сторінку з використанням селекторів і фільтрів jQuery для зміни властивостей елементів.
4. Створити сторінку з використанням функцій jQuery для зміни елементів на сторінці зі скриптів.
5. Створити сторінку і включити для неї обробники подій різними способами, в обробниках задати різні анімаційні ефекти (не менше чотирьох).
6. Знайти в Інтернеті фотогалерею на основі jQuery, адаптувати та розмістити на своїй сторінці.

Контрольні запитання для самодіагностики

1. Дайте визначення поняттям "бібліотека", "фреймворк".
2. Назвіть відомі вам бібліотеки.
3. Що дає використання jQuery розробнику?
4. Назвіть групи функцій, реалізовані в ядрі jQuery.
5. Опишіть механізми селекції елементів за допомогою jQuery, поясніть прикладами.
6. Яку обробку елементів і даних забезпечують функції jQuery?
7. Опишіть ефекти, які можна створювати засобами jQuery.

Методичні рекомендації

Для підвищення ефективності розробки клієнтських скриптів створено достатньо багато різних бібліотек і фреймворків.

Бібліотеки у програмуванні є набором об'єктів і функцій, які можуть використовуватися багато разів. Включення їх у код скрипта забезпечує вищий рівень абстракції під час розробки, зменшується об'єм написаного коду, розширюється кросплатформеність і так далі.

Фреймворк – це каркас програмної системи (або підсистеми). Він може включати бібліотеки коду, допоміжні програми, мову сценаріїв і інші засоби, що полегшують розробку й об'єднання різних компонентів великого програмного проекту. Зазвичай, об'єднання відбувається за рахунок використання єдиного API.

Однією з найпопулярніших бібліотек, що розширюють можливості зі створення клієнтських скриптів, є jQuery. Оскільки чіткої межі між бібліотеками і фреймворками немає, відносно jQuery вживаються обидва терміни.

Ядро бібліотеки (основний файл) має невеликий розмір (до 200 Кб). Бібліотека має безліч розширень, орієнтованих на різні додатки галереї, меню і тому подібне. Вони підключаються в міру необхідності.

Для підключення jQuery досить включити в текст сторінки тег <script type="text/javascript" src=".."></script>, і як значення атрибута src вказати шлях до файла з кодом ядра. Файл (наприклад, jquery.js) може знаходитися на сервері або безпосередньо в теці сайта. У другому разі переважно під час відладки і виконання скриптів на комп'ютері без підключення до мережі Інтернет.

Для звернення до функцій jQuery використовується конструкція $(arg) або jQuery(arg). Тут arg – список аргументів. Аргументом може бути HTML-код, селектор для ідентифікації елементів, ім'я функції і тому подібне. В більшості випадків буде повернено посилання на колекцію об'єктів або єдиний об'єкт. Наприклад,

```
varmy= $("<div><p>Текст!</p></div>");
my = jQuery("<div><p>Текст!</p></div>");
varmy=$(document.body).
```

Функції для виконання дій з об'єктами викликаються як методи об'єктів. Наприклад $("p.hig").css("background-color", "#000"); тут $("p.hig") поверне посилання на абзаци, у яких атрибут class="hig", а потім буде викликаний метод css, який встановить для них значення властивості background-color.

До складу ядра включені такі групи функцій:

робота з селекторами;

робота з атрибутами;

робота з CSS-властивостями елементів;

маніпуляції з елементами сторінки (додавання, видалення, зміна і тому подібне);

робота з подіями;

обхід дерева DOM;

візуальні ефекти;

взаємодія з сервером (ajax).

Селектори в jQuery базуються на CSS-селекторах, а також підтримують XPath. Вони дозволяють знайти (задати посилання на) один або декілька елементів сторінки. Будь-який CSS-селектор може бути використаний як аргумент функції $() або jQuery(): $("div span"). Будуть вибрані всі елементи span, які знаходяться всередині div.

Для більшої гнучкості можна використовувати фільтри – функції, які задають додаткові ознаки для відбору. Фільтри схожі на псевдокласи в CSS і вказуються через двокрапку після селектора. Наприклад, $('div:first') – буде вибраний перший div у дереві DOM, $('div:eq(N)') – буде вибраний div номер N у дереві DOM, і тому подібне. Список найбільш часто використовуваних фільтрів наведений в табл. 5.

Таблиця 5

Фільтри

Функція	Дія
":focus"	елемент, що знаходиться у фокусі
":first"	перший знайдений елемент
":last"	останній знайдений елемент
":header"	елементи, що є заголовками (з тегами h1, h2 і т. д.)
":animated"	елементи, які в даний момент задіяні в анімації
":hidden"	невидимі елементи сторінки
":visible"	видимі елементи сторінки
":contains(text)"	елементи, що містять заданий текст
":empty"	елементи без вмісту (без тексту і інших елементів)
":parent"	непорожні елементи
":button"	елементи з тегом button або типом button
":radio"	елементи, що є перемикачами
":checkbox"	елементи, що є прапорцями
":text"	елементи, що є текстовими полями
":file"	елементи, завантаження файлів, що є полями
":checked"	вибрані елементи (із статусом checked). Це можуть бути елементи типу <checkbox> або <radio>

У складі бібліотеки є набір функцій для зміни значень атрибутів і властивостей, які задаються в таблицях стилів (табл. 6).

Таблиця 6

Атрибути і властивості CSS

Функція	Дія
.attr()	повертає/змінює (залежно від числа параметрів) значення атрибута в елементах на сторінці
.removeAttr()	видаляє атрибут у елементів на сторінці
.addClass()	додає клас елементам на сторінці
.removeClass()	видаляє клас в елементах на сторінці
.toggleClass()	змінює наявність класу в елементах на протилежне (додає/видаляє)
.hasClass()	перевіряє наявність класу хоч би в одного з вибраних елементів
.val()	повертає/змінює (залежно від числа параметрів) значення атрибута value в елементах на сторінці
.css()	повертає/змінює (залежно від числа вхідних параметрів) CSS параметри елемента
.height(), .innerHeight(), .outerHeight()	повертає/змінює висоту елемента
.width() .innerWidth().outerWidth()	повертає/змінює ширину елемента
.position() .offset()	повертає/змінює позицію елемента
.scrollTop() .scrollLeft()	повертає/змінює величину скролінгу (прокрутки) елемента

Приклад сторінки, що демонструє їх використання, наведений далі (pr12.htm).

```
<html>
<head>
<title>Зміна значень атрибутів</title>
<STYLE TYPE="text/css">
#box {width:200;height:200;border:"2px solid #0000FF";
background-color: #FCFFB2;color: #3D91FF;}
</style>
<script src="jquery.js" type="text/javascript"></script>
<script>
function f1()
{
$("span").attr("style", "color:red");
}
function f2()
```

```
{
my=$("#lab");
my.css("color", "green");
}
</script>
</head>
<body>
<div id="box">
<spanid="lab">Перший розділ з текстом</span>
<span style="color:blue">Другий розділ з текстом</span>
</div>
<button onclick="f1()">f1</button>
<button onclick="f2()">f2</button>
</body>
</html>
```

Маніпуляції з елементами полягають у видаленні, додаванні і зміні вмісту, відповідні функції наведені в табл. 7.

Таблиця 7

Маніпуляції з елементами

Функція	Дія
.html()	Повертає/змінює (залежно від числа параметрів) html-вміст елементів на сторінці
.text()	Повертає/змінює (залежно від числа параметрів) текст, що знаходиться в елементах на сторінці
.append() .appendTo()	Додає заданий вміст у кінець елементів на сторінці
.prepend() .prependTo()	Додає заданий вміст у початок елементів на сторінці
.after() .insertAfter()	Додає заданий вміст після елементів на сторінці
.before() .insertBefore()	Додає заданий вміст перед елементами на сторінці
.wrap() .wrapAll()	Оточує елементи на сторінці заданими html-елементами
.wrapInner()	Оточує вміст елементів на сторінці заданими html-елементами
.detach() .remove()	Видаляє елементи на сторінці
.empty()	Видаляє вміст елементів на сторінці
.unwrap()	Видаляє батьківські елементи, у ході цього їх вміст залишається на місці
.replaceWith() .replaceAll()	Замінює одні елементи сторінки на інші (нові або такі, що вже існують)
.clone( )	Повертає копію вибраних елементів сторінки

Для полегшення дій з обробки подій до складу бібліотеки включена група функцій для установки і видалення обробників (on(), off()), а також ряд допоміжних, наприклад, функції для конкретних подій.

Установка обробника для елементу з id="foo":

```
$('#foo').on('click',function(){alert('Ви натиснули на елемент "foo"');
```

Установка і видалення обробника:

```
function handler() {                    // оголошення функції,
alert('Обробник встановлений');         // яка стане обробником
};
$('#foo').on('click', handler);         // установка обробника для foo
        …
$('#foo').off('click', handler);        // видалення обробника
```

Детальніше про управління обробкою подій можна ознайомитися у довідці [12].

Функції створення ефектів. jQuery має ряд функцій, що виконують анімаційні ефекти з елементами сторінки: приховування, поява, переміщення елементів (табл. 8).

Таблиця 8

Функції створення ефектів

Функція (метод)	Дії
show([speed[, callback]])	показати елемент
hide([speed[, callback]])	приховати елемент
fadeIn(speed[, callback])	показати елемент шляхом зміни його прозорості
fadeOut(speed[, callback])	приховати елемент шляхом зміни його прозорості
slideDown(speed, callback)	показати елемент, спустивши його зверху
slideUp(speed, callback)	показати елемент, піднявши його знизу

Параметр callback – це ім'я функції, яка буде викликана після завершення ефекту.

Бібліотека дозволить створювати і свої, складніші ефекти, які засновані на зміні CSS властивостей. Ці зміни можуть відбуватися по-різному, плавно або миттєво, сповільнюватися, прискорюватися або виконуватися рівномірно. Для їх створення використовується функція animate({"имя_вл":"тип_дії"},"тривалість"). Серед властивостей можуть використовуватися тільки такі, у яких значенням є число. Наприклад, $("#mydiv").animate({height: "hide"}, 300).

Важливою особливістю більшості методів jQuery є можливість зв'язувати їх у ланцюжки для послідовного виконання. Це дозволяє використовувати такий запис $("#bigIt").empty().attr("class", "noContent").

У результаті виконання цих дій у елемента з ідентифікатором bigIt буде видалено весь вміст, а потім йому буде приписаний атрибут class="noContent".

Ланцюжки можуть складатися зі значного числа методів. Таким чином, можна легко описати всі дії, що відбуваються з вибраними елементами, уникнувши використання великого числа тимчасових змінних.

Приклад коду сторінки (pr13.htm) з анімацією і ланцюжками методів наведений далі.

```
<html>
<head>
<title>Анимация</title>
<STYLE TYPE="text/css">
#box {width:200;height:200;border:"2px solid #0000FF";background-color: #FCFFB2;color: #3D91FF;}
</style>
<script src="jquery.js" type="text/javascript"></script>
<script>
function f1()
{
$("#box").animate({height:"hide"}, 300).text("НОВИЙТЕКСТ") .animate({height: "show"}, 300);
}
function f2()
{
$("#box").animate({width:"hide"}, 300).text("СТАРИЙТЕКСТ") .animate({width: "show"}, 300);
}
</script>
</head>
<body>
<div id="box">
<span id="lab">СТАРИЙ ТЕКСТ</span>
</div>
<button onclick="f1()">f1</button>
<button onclick="f2()">f2</button>
</body>
</html>
```

Виконання скриптів на сторінці має бути синхронізовано з роботою браузера щодо інтерпретації HTML-коду і побудові дерева DOM. Для цього може бути використана подія onload. Вона відбувається після того, як сторінка сформована повністю, включаючи завантаження всіх зображень, флеш-банерів і відеороликів.

```
…
window.onload = function(){
  // виклик потрібних функцій скрипта
};
…
```

Крім того, у складі функцій jQuery є метод ready, виклик якого здійснюється у момент готовності дерева DOM (відбувається раніше, ніж onload).

```
…
$(document).ready( function(){
  // виклик потрібних функцій скрипта
});
…
```

Далі наведений код сторінки (pr14.htm), що демонструє різні способи задавання обробників переривань і ефекти.

```
<html>
<head>
<title>Створення ефектів і задавання обробників</title>
<STYLE TYPE="text/css">
#box {width:200;height:200;border:"2px solid #0000FF";background-color:
#FCFFB2;color: #3D91FF;}
</style>
<script src="jquery.js" type="text/javascript"></script>
<script>
$(document).ready(function() {
$("#b1").click(function f1() {$("#box").hide(300);});
$("#b2").bind("click",function f2() {$("#box").show(300);});
$("#b3").click(function f3() {$("#box").fadeOut(1000);});
$("#b4").click(function f4() {$("#box").fadeIn(1000);});
});
</script>
</head>
<body>
```

```
<div id="box">
<spanid="lab">Перший розділ з текстом</span>
<span style="color:blue">Другий розділ з текстом</span>
</div>
<button  id="b1">сховати</button>
<button  id="b2">показати</button>
<button  id="b3">розчинити</button>
<button  id="b4">проявити</button>
</body>
</html>
```

Окрім маніпуляцій з вибраними елементами, jQuery дозволяє працювати з самим набором: змінювати його, а також працювати з елементами окремо. Наприклад,

$("div").parent() поверне батьківські елементи всіх div;

$("div").children() поверне дочірні елементи всіх div;

$("#someId").next() поверне елемент, який знаходиться відразу після someId.

Насправді тих методів, які працюють з набором, значно більше, їх повний список можна знайти у відповідному розділі довідки [12].

Для того, щоб явно вказати на поелементну обробку набору використовується метод .each(). Як параметр цей метод приймає опис функції, яка буде автоматично викликана для кожного елемента набору. Також як параметр функції передається індекс елемента в наборі, починаючи з 0, а сам елемент доступний як властивість об'єкта this.

Наприклад:

```
$ (‘Img’). each (function (n)  {
                                this.alt = ‘рисунок номер’+n+'id= ‘+ this.id;
                                }
                );
```

У результаті виконання цього фрагмента будуть вибрані всі малюнки на сторінці, і для кожного з них виконуватиметься функція, яка сформує заміщуючі написи.

Література: основна [2; 3]; ресурси мережі Інтернет [11; 12].

Тема 5. Візуалізовані засоби створення WEB-документів та публікація WEB-вузла

Самостійна робота № 5. Робота у середовищі Microsoft Expression WEB

Мета роботи: освоїти типові дії щодо створення сайтів у середовищі програми Microsoft Expression WEB.

У результаті виконання самостійної роботи у студента формуються такі **компетентності:** здатність створювати сайти у візуалізованих середовищах та самостійно вивчати нові програмні продукти.

Результатом виконання самостійної роботи є звіт із виконання завдання та документи, які створені під час виконання пунктів завдання.

Завдання для самостійної роботи

1. Вивчити довідкові матеріали до самостійної роботи.
2. Виконати всі приклади, наведені у довідкових матеріалах.
3. Виконати настройку програми для подальшої роботи.
4. Створити сайт і розмістити в ньому три сторінки з різними елементами.
5. Відформатувати створені сторінки за допомогою таблиць стилів.
6. Створити на кожній сторінці свої динамічні елементи.

Контрольні запитання для самодіагностики

1. Назвіть відомі вам програми для візуалізованого створення сайтів.
2. Сформулюйте послідовність дій під час створення документів у візуалізованому середовищі.
3. Сформулюйте рекомендації щодо налаштування середовища програми.
4. Опишіть дії зі створення сайта та розміщення в ньому нових сторінок.
5. Опишіть дії щодо форматування сторінок за допомогою таблиць стилів.
6. Опишіть можливості програми з розробки клієнтських скриптів.

Методичні рекомендації

Налаштування програми Expression Web виконуються у вікнах, що відкриваються командами Application Options і Page Editor Options з меню Tools. Для початку роботи зазвичай достатньо установок за замовчуванням.

У головному вікні програми (рис. 5) знаходяться засоби, які необхідні для виконання основних дій.

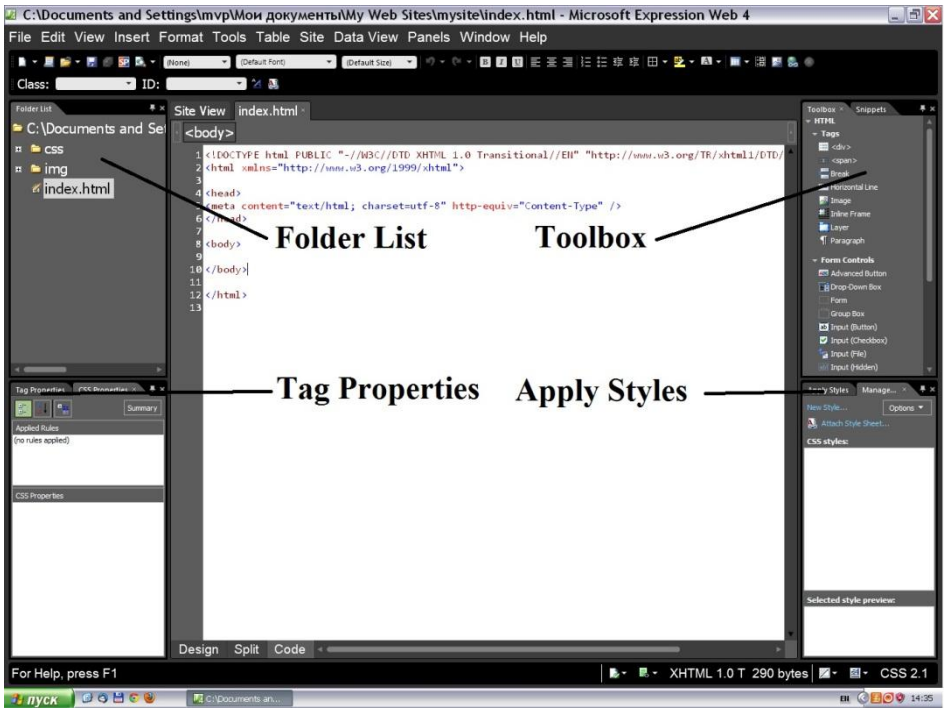

Рис. 5. **Вікно програми Expression Web**

У центрі розташоване вікно редагування, в якому подані вкладки, відповідні відкритим документам (вкладки вздовж верхнього краю вікна редагування відповідають відкритим WEB-сторінок і сайтів). Панелі завдань та інструментів із боків вікна редагування містять найбільш вживані інструменти. Якщо необхідно, ці панелі можна приховати, а потім знову показати. За замовчуванням Expression Web показує чотири панелі завдань: Folder List (Список папок), Tag Properties (Властивості тегів), Apply Styles (Застосувати стилі) і Toolbox (Набір інструментів).

Для вікна редагування передбачено три режими. У режимі конструювання (Design view), відображається вид сторінки, відповідне вікно браузера. У режимі програмування (Code view) показується код сторінки,

а в комбінованому режимі (Split view) – вікно поділяється на дві частини (відображується код та зовнішній вигляд). Кнопки для перемикання знаходяться в лівому нижньому кутку вікна. Для створення окремої сторінки необхідно вибрати команду New з меню File, а у вікні – тип створюваної сторінки.

Для створення нового сайта слід вибрати пункт New Site з меню Site, потім у вікні вибирається тип створюваного сайта. Наприклад, Empty Web Site (порожній сайт), вказується місце збереження папок сайта, натиснути ОК. У вікні Site View необхідно створити необхідну структуру папок. Структура сайта (комплект папок і файлів) відображається у вікні сайта і панелі Folder List.

У створений сайт можна додати порожню сторінку або копію однієї зі створених раніше (для збереження єдності стильового оформлення і однотипних елементів). Для створення копії необхідно викликати контекстне меню на необхідній сторінці і вибрати команду New From Existing Page (Створити на основі існуючої). У вікні редагування з'явиться нова сторінка без назви з таким же вмістом і форматуванням, як сторінка-шаблон. Потім створену сторінку необхідно зберегти, задавши місце і відредагувавши ім'я.

Властивості створюваної сторінки (заголовок, список ключових слів, кодування тощо) задаються командою Properties із меню File. Розміщення тексту на сторінках і його структурування виробляється з використанням таких же прийомів, як у текстових редакторах. Абзаци тексту можуть бути перетворені в заголовки і списки за допомогою команд із меню або кнопок панелі інструментів. Для вставки на сторінку зображень, посилань, розділів та інших елементів необхідно скористатися відповідними командами з меню Insert.

Будь-який з вставлених елементів можна відредагувати за допомогою вкладки Tag Properties на панелі завдань. У Expression Web є безліч інструментів для роботи зі CSS-стилями. Вкладка Manage Styles (Управління стилями) дозволяє створити та організувати стилі. Вкладка Apply Styles (Застосування стилів) дозволяє подивитися зразки кожного стилю. Панель CSS Properties (Властивості CSS) дозволяє вносити зміну до створені правила форматування. При цьому пропонуються три способи впорядкування властивостей CSS: зведення всіх тегів у вибраній області сайта, каскадне подання, що показує, як стилі пов'язані між собою, і перерахування за алфавітом або за категоріями. До CSS також мають

відношення дві панелі інструментів (зазвичай, у верхній частині вікна). Панель інструментів Style Application (Застосування стилів) використовується головним чином у тих випадках, коли необхідно створювати стилі в ручному режимі. Панель інструментів Style (Стилі) включає випадають меню для застосування стилів до класів і ідентифікаторів, а також кнопки для створення стилів і приєднання таблиць стилів.

Щоб підготувати середовище для роботи з CSS необхідно: включити панелі завдань Tag Properties, CSS Properties, Apply Styles і Manage Styles; вибрати в меню View пункт Visual Aids і відзначити всі прапорці, крім двох, що відносяться до ASP.NET; вибрати в меню View пункт Quick Tag Selector; вибрати в меню View пункт Toolbars і відзначити панель Style Application або Style. Для створення нового правила необхідно виконати команду New Style з меню Format або натиснути кнопку New Style на панелі Apply Styles або Manage Styles. Відкриється вікно для задавання всіх параметрів: використовуваний селектор, місце розміщення, всі властивості.

Для редагування вже створених правил потрібно виділити селектор в панелі Manage Styles або коді сторінки, а потім задати необхідні декларації в панелі CSS Properties. Для додання сторінці динамічних властивостей (зміна зовнішнього вигляду, реакція на дії користувача) створюються і вбудовуються в сторінку обробники подій. У програмі Expression Web передбачена технологія Behaviors, що спрощує цей процес. Behaviors (поведінки) – це набір різних скриптів, що виконують типові дії з елементами (зміна зображень, приховування та показ елементів тощо). Дії з Behaviors (вибір, зв'язування з подією) виконуються за допомогою інструментів панелі Behaviors. Панель відкривається командою Behaviors з меню Panels.

Щоб додати поведінку необхідно в режимі Design виділити елемент (тег елемента з'являється на панелі завдань Behaviors). Потім розкрити список доступних поведінок (кнопка Insert) і вибрати потрібну. Відкриється діалогове вікно, в якому можна задати параметри, що визначають дію. На панелі завдань Behaviors з'явитися рядок, що містить подія і дія, прив'язані до елемента. З елементом можуть бути пов'язані кілька поведінок.

Для зміни параметрів необхідно двічі клацнути на імені поведінки в панелі, відкриється вікно задавання параметрів. Для зміни події необхідно розкрити список подій в рядку поведінки на панелі завдань і вибра-

ти нову подію. Для видалення поведінки необхідно виділити рядок і натиснути кнопку Delete в панелі завдань.

Література: основна [4].

Самостійна робота № 6. Робота у середовищі Dreamweaver

Мета роботи: освоїти типові дії щодо створення сайтів у середовищі програми Dreamweaver.

У результаті виконання самостійної роботи у студента формуються такі **компетентності:** здатність створювати сайти у візуалізованих середовищах та самостійно вивчати нові програмні продукти.

Результатом виконання самостійної роботи є звіт із виконання завдання та документи, які створені під час виконання пунктів завдання.

Завдання до самостійної роботи

1. Вивчити довідкові матеріали до самостійної роботи.
2. Виконати всі приклади, наведені у довідкових матеріалах.
3. Ознайомитися з можливими настройками і виконати налаштування програми для подальшої роботи.
4. Розробити і створити в середовищі Dreamweaver структуру сайта.
5. Створити не менше трьох пов'язаних сторінок, використовуючи різні режими роботи.
6. Відформатувати створені сторінки за допомогою CSS.
7. Розмістити на сторінках динамічні елементи, використовуючи behaviors.

Контрольні запитання для самодіагностики

1. Дайте порівняльну характеристику програм Dreamweaver і Microsoft Expression WEB.
2. Дайте характеристику можливостей програми Dreamweaver з управління сайтами.
3. Назвіть настройки, які доцільно виконати перед початком роботи зі створення сайта.
4. На що впливає установка перемикача **Use CSS instead of HTML tags**?

5. Як виконати розміщення та редагування елементів сторінки в режимі **design**?

6. Як пов'язані між собою режими **design** і **code**?

7. Як створити нове правило для таблиці стилів?

8. Як змінити існуючу таблицю стилів?

9. Як задати поведінку (behaviors) для елемента?

10. Як підключити власний скрипт для дій з елементом?

Методичні рекомендації

Вікно програми оформлено стандартно, містить заголовок, головне меню, поле документів, ряд додаткових вікон (рис. 6).

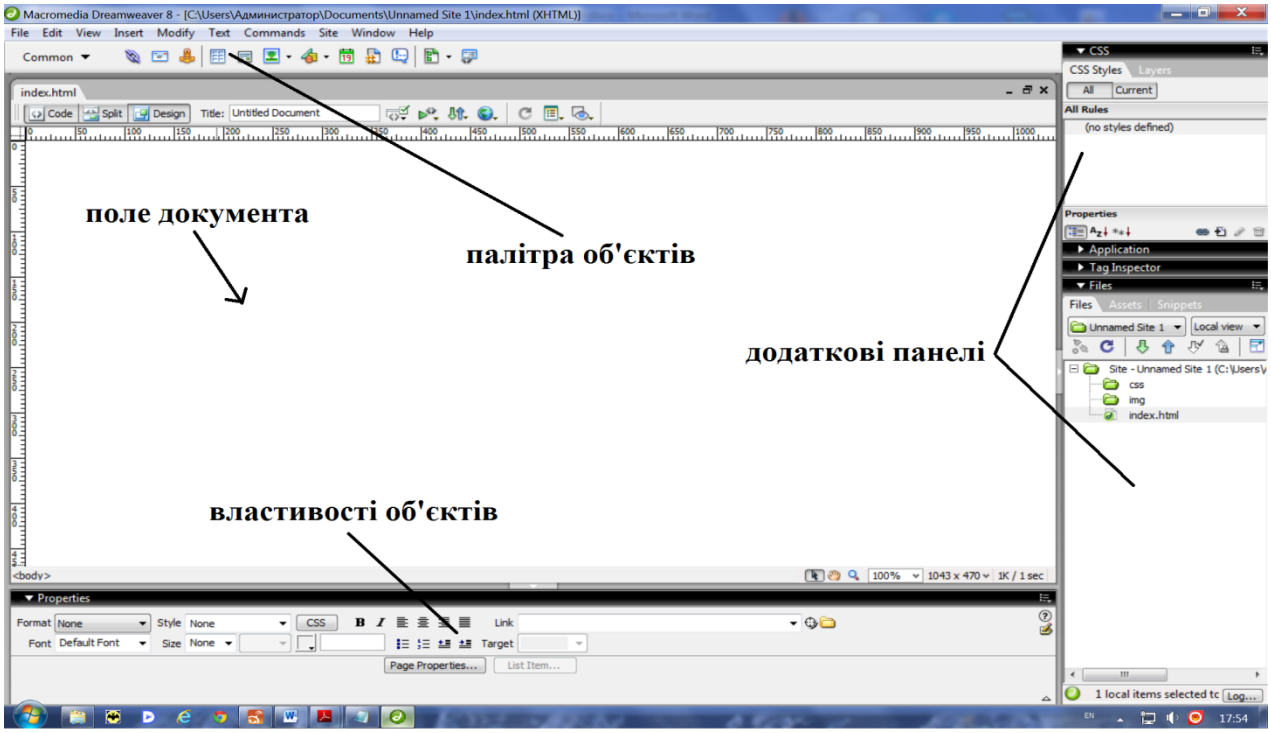

Рис. 6. **Вікно програми**

Поле документа слугує для відображення WEB-сторінки. Одночасно може бути відкрито декілька сторінок, корінці відкритих сторінок у верхній частині поля дозволяють переглядати одну з них. Вікно сторінки може відображатися в такому з трьох режимів: code (відображається код розмітки), design (відображається як у браузері), split (поєднує два перших подання), перемикачі знаходяться у верхній лівій частині вікна.

Основними елементами інтерфейсу є головне меню, палітра об'єктів (містить піктограми основних елементів для розміщення на сторінці),

панель властивостей об'єкта (для задавання і зміни основних характеристик елементів, виділених на сторінці), ряд додаткових панелей (історія, стилі і т. д.). Управління елементами інтерфейсу (включити/виключити) здійснюється з меню Window.

Для створення сайта можна рекомендувати таку послідовність дій:

налаштування програми;

створення структури папок сайта;

створення сторінок, розміщення елементів;

форматування сторінок;

створення функціональних елементів.

Налаштування програми зібрані у вікні команди Preferences з меню Edit. Вікно містить безліч вкладок із різними елементами управління. Перед роботою рекомендується встановити такі з них:

1. На вкладці General зняти відмітку з перемикача Use CSS instead of HTML tags.

2. На вкладці New Document вибрати необхідні значення в полях Default Document Type (DTD), Default document, Default extension, Default encoding.

3. На вкладці Fonts в поле Fonts settinge встановити "Кириллица".

4. Інші налаштування можна встановлювати в міру необхідності.

Для створення нового сайта можна скористатися майстром, який запускається зі стартової сторінки кнопкою Dreamweaver site... або командою Manage Sites... зі списку Files в панелі Files (у вікні, вибрати New...). Майстер виконає відповідні налаштування і створить папку. У ході цього можна відразу розмістити папку на сервері. Для автономної роботи можна просто створити папку в необхідному місці.

Після цього за допомогою інструментів панелі Files (кнопки, контекстне меню) можна створити необхідну структуру папок (для таблиць стилів, для малюнків тощо). Слід зазначити, що приступаючи до створення сайта, необхідно чітко уявити собі його структуру, розміщення основних матеріалів по каталогах.

Для створення сторінок можна використовувати вікно команди New... з меню File. У вікні можна вибрати тип і шаблон для створюваної сторінки. Після цього сторінку необхідно зберегти стандартним чином. Порожню сторінку можна створити і за допомогою інструментів панелі Files. Для видалення сторінки її необхідно виділити в панелі Files, і натиснути клавішу Delete.

Перед розміщенням елементів доцільно задати основні властивості сторінки (задаються за допомогою атрибутів тега body). Для цього необхідно зайти в меню Modify, вибрати пункт Page Properties і задати необхідні значення.

Усі основні дії під час створення і редагування HTML-сторінки виконуються в полі документа.

У режимі code можна вводити код сторінки, використовуючи підказки і автозаповнення.

У режимі design сторінка створюється шляхом розміщення на ній елементів з візуалізацією результатів. Елементи вибираються з меню Insert і Text або в панелі інструментів (палітра об'єктів). Для різних елементів можуть відкриватися додаткові діалогові вікна, в яких задаються параметри, що описують їх особливості.

Для швидкого доступу до редагування параметрів об'єкта призначена панель властивостей Properties. Залежно від поточного виділеного об'єкта вона змінює свій зовнішній вигляд, відображаючи набір властивостей саме цього об'єкта. Слід зазначити, що у разі рекомендованих налаштувань (включений Use CSS instead of HTML tags) параметри задаються виключно як значення атрибутів тегів.

Одним із широко використовуваних у HTML засобів є таблиці. Причому, дуже часто вони використовуються не тільки традиційно – як метод подання табличних даних, але і як засіб позиціонування елементів і верстки WEB-сторінок.

Додати таблицю в редагований документ в Dreamweaver можна кількома способами. Можна вибрати кнопку Table на панелі об'єктів або скористатися відповідною командою з меню Insert. Після цього на екрані з'являється діалогове вікно, в якому необхідно задати параметри майбутньої таблиці.

Після того, як таблиця створена, осередки заповнюються вмістом. Крім того, можна змінювати параметри таблиці: додавати і видаляти рядки і стовпці, об'єднувати і ділити осередки, міняти розміри всієї таблиці й окремих рядків і стовпців, колір фону. Виконуються ці дії з використанням панелі Properties або мишею так, як це робиться в текстовому редакторі Word.

У Dreamweaver для роботи зі стилями CSS призначена панель стилів (команда включення панелі CSS Styles знаходиться в меню Window). Для створення нового стилю потрібно натиснути кнопку New CSS Rule на

панелі стилів або вибрати команду New... з контекстного меню, що з'являється під час клацання правою клавішею миші в будь-якому місці панелі стилів. У діалоговому вікні необхідно вибрати тип селектора – список Selector Type.

Class пов'язує правило зі значенням атрибута CLASS. При цьому в полі Name необхідно вказати значення атрибута CLASS. Воно повинно починатися з символу крапки.

Tag пов'язує параметри форматування з ім'ям тега, яке необхідно вибрати в списку Tag.

Advanced дозволяє використовувати значення атрибута ID.

Далі необхідно визначити, де буде розташований опис стилів – список Define in. Під час вибору New Style Sheet File правило буде збережено в окремому файлі, посилання на який (тег link) буде вставлено в заголовок документа (якщо з редагованою сторінкою вже пов'язані які-небудь файли з описом стилів, то їх імена будуть перераховані в списку).

Під час вибору This document only опис стилів буде збережено безпосередньо в редагованому документі (тег style).

Після того, як всі параметри визначені, натискається кнопка ОК, після чого відкриється діалогове вікно визначення параметрів нового стилю. Параметри розподілені на групи. Якщо якісь із параметрів не мають значення для створюваного стилю, то відповідні їм поля повинні залишитися незаповненими.

У разі необхідності відформатувати сторінку з використанням уже існуючої таблиці стилів (файл із розширенням .css), необхідно в панелі стилів клацнути по кнопці Attach Style Sheet або вибрати команду Attach Style Sheet в контекстному меню.

Для внесення змін до опису стилів слід вибрати необхідний стиль у панелі CSS і натиснути кнопку Edit Style..., при цьому відкривається діалогове вікно визначення параметрів стилю.

Редагування також можливо безпосередньо в панелі стилів у списку властивостей.

Видалити правило можна за допомогою команди Delete з контекстного меню обраного стилю. Так само можна видалити виділений стиль на панелі стилів за допомогою кнопки Delete CSS Rule.

Для полегшення роботи зі створення сайтів користувачам, не знайомим з програмуванням, в Dreamweaver передбачений заздалегідь

певний набір WEB-сценаріїв, які можна вставити в сторінку. Ці сценарії називаються поведінками (behaviors). Серед поведінок є такі, які поміщають в рядок статусу довільний текст, приховують або, навпаки, показують вільно позиціонований контейнер, міняють зображення на WEB-сторінці і т. д. Поведінки Dreamweaver під час створення прив'язуються до конкретного елемента сторінки і події, у відповідь на що виконується відповідний йому сценарій. Наприклад, якщо поведінка прив'язана до зображення і події "закінчення завантаження", то воно виконається відразу після закінчення завантаження файла цього зображення. До одного елемента сторінки можуть бути прив'язані кілька поведінок, кожна – до своєї події.

Для задавання поведінки необхідно виділити на сторінці елемент, натиснути кнопку Add Behavior і вибрати зі списку доступних необхідну поведінку.

Список Behaviors містить:

Call Javascript – виклик зовнішнього скрипта;

Change Property – зміна властивостей об'єкта (кольори, наприклад);

Check Browser – перевірка типу браузера;

Check Plugin – перевірка наявності плагіна (наприклад, для програвання Flash);

Control ShockWave or Flash – управління флеш;

Drag Layer – переміщення шару;

Go To Url – перехід за посиланням;

Jump Menu, Jump Menu Go – зміна меню;

Open Browser Window – відкриття вікна браузера з заданими розмірами та іншими параметрами;

Play Sound – відтворення звуку;

Popup Message – виведення повідомлення із заданим текстом;

Swap Image – зміна однієї картинки іншою (ефект перекочування);

Swap Image Restore – відновлення вихідної картинки;

Validate Form – перевірка введених у форму даних.

Література: основна [1].

Рекомендована література

Основна

1. Дронов В. А. Самоучитель Macromedia Dreamweaver 8 / В. А. Дронов. – СПб. : БХВ-Петербург, 2006. – 320 с.

2. Методичні рекомендації по виконанню лабораторних робіт з навчальної дисципліни "Основи проектування WEB-видань" для студентів спеціалізації "Комп'ютеризовані технології та системи видавничо-поліграфічних виробництв" усіх форм навчання укл. В. П. Молчанов, Т. Ю. Андрющенко. – Х. : Вид. ХНЕУ, 2009. – 84 с.

3. Молчанов В. П. Основи проектування WEB-видань : конспект лекцій / В. П. Молчанов. – Х. : Вид. ХНЕУ, 2008. – 168 с.

4. Хестер Н. Создание Web-сайтов в Microsoft Expression Web / Н. Хестер. – М. : ДМК Пресс, 2007. – 252 с.

Додаткова

5. Нильсен Я. Дизайн Web-страниц. Анализ удобства и простоты использования 50 узлов / Я. Нильсен, М. Тахир; пер. с англ. – М. : Издательский дом "Вильямс", 2002. – 336 с.

6. Сакс Т. Дизайн и архитектура современного вебсайта. Опыт профессионалов / Т. Сакс, Г. Мак-Клейн; пер. с англ. – М. : Издательский дом "Вильямс", 2002. – 304 с.

Ресурси мережі Інтернет

7. Карманное руководство по написанию SVG [Электронный ресурс]. – Режим доступа : http:// http://css-live.ru/category/svg.

8. Справочник CSS [Электронный ресурс]. – Режим доступа : http://htmlbook.ru/css.

9. Справочник по HTML [Электронный ресурс]. – Режим доступа : http://htmlbook.ru/html.

10. Справочник HTML5 Canvas [Электронный ресурс]. – Режим доступа : http://xiper.net/manuals/canvas/.

11. Справочник JS [Электронный ресурс]. – Режим доступа : http://javascript.ru/manual.

12. jQuery [Электронный ресурс]. – Режим доступа : http://jquery.page2page.ru/index.php5/.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Методичні рекомендації до самостійної роботи з навчальної дисципліни "ОСНОВИ ПРОЕКТУВАННЯ WEB-ВИДАНЬ"

для студентів напряму підготовки 6.051501 "Видавничо-поліграфічна справа" всіх форм навчання

Самостійне електронне текстове мережне видання

Укладач **Молчанов** Віктор Петрович

Відповідальний за випуск *Пушкар О. І.*

Редактор *Бутенко В. О.*

Коректор *Ковальчук М. А.*

План 2015 р. Поз. № 106 ЕВ. Обсяг 56 с.

Видавець і виготівник – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Леніна, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***