МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ОСНОВИ ПРОЕКТУВАННЯ WEB-ВИДАНЬ

Методичні рекомендації до лабораторних робіт
для студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2018

УДК 004.415.26(07.034)
О-75

Укладач В. П. Молчанов

Затверджено на засіданні кафедри комп'ютерних систем і технологій.
Протокол № 10 від 04.04.2018 р.

Самостійне електронне текстове мережеве видання

Основи проектування WEB-видань : методичні рекомендації
О-75 до лабораторних робіт для студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня [Електронний ресурс] / уклад. В. П. Молчанов. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 91 с.

Матеріали, подані в роботі, є методологічною та методичною основою для опанування студентами інструментальних засобів WEB-технологій та прийомів їхнього застосування з метою створення документів для мережі Інтернет.

Рекомендовано для студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня.

УДК 004.415.26(07.034)

Вступ

Лабораторні заняття з дисципліни "Основи проектування WEB-видань" проводяться з метою закріплення теоретичних знань і придбання практичних навичок в розробленні документів для мережі Інтернет. Вони утворюють лабораторний практикум, спланований за єдиним задумом, який складається з дев'яти робіт, що охоплюють усі теми дисципліни.

Лабораторні заняття проводять після викладання лекційного матеріалу та передбачають попередню підготовку студентів до виконання завдань. Підготовка полягає у вивченні теоретичного матеріалу, завдання на лабораторну роботу та підготовку необхідних матеріалів (рисунків, параметрів форматування для таблиць стилів, текстів програм і тому подібне).

Усі лабораторні заняття з дисципліни проводяться фронтально, кожний студент працює за окремим комп'ютером.

На початку заняття викладач здійснює контроль підготовленості студентів, як правило, шляхом перевірки правильності відповідей на контрольні запитання, рідше – у формі усної бесіди за темою заняття. Для контролю може використовуватись і тестування. Обов'язково перевіряється наявність матеріалів для виконання роботи. За відсутності матеріалів, необхідних для виконання роботи, і знань, які не дозволяють виконати роботу, студент до роботи не допускається, і йому пропонується виконати необхідну підготовку. У такому випадку саму роботу студент виконує в додатковий час.

Кожне заняття розраховано на 6 годин навчального часу. Студент повинен виконати всі дії відповідно до завдання та зберегти результати у прийнятному вигляді.

За наслідками контролю готовності студентів до роботи, обсягу та правильності її виконання, якості збережених результатів і відповідей на контрольні запитання викладач виставляє оцінку.

Студент, що пропустив лабораторне заняття або був не допущений до нього, зобов'язаний виконати відповідну роботу під час самостійної підготовки та відзвітувати. Повторна здача робіт, які не були прийняті, проводиться під час додаткових занять.

Оцінки, отримані студентом за окремі лабораторні заняття, враховуються у процесі проставлення поточної модульної оцінки з навчальної дисципліни.

Змістовий модуль 1. Створення WEB-документів

Тема 1. Проектування WEB-сайта

Лабораторна робота 1
Проектування WEB-сайта

Мета – вивчення характеристики змісту етапів і планування робіт зі створення сайтів.

Після виконання лабораторної роботи студент повинен:

знати:

змістовність етапів створення сайтів;

склад і зміст документів, які визначають розробку;

вимоги до дизайну документів для сервісу WWW;

уміти:

планувати роботу зі створення сайтів;

приймати основні дизайнерські рішення, створювати концепцію сайта.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з планування роботи зі створення сайта та взаємодії з замовником.

Загальні відомості

Створення сайта – достатньо складний багатоетапний процес. Практика, що склалася, дозволяє розглядати його як послідовність таких етапів:

планування сайта;

розроблення дизайну, включаючи створення елементів оформлення (логотипи, флеш, шрифти і т. п.);

HTML-верстання сайта;

контент-наповнення сайта;

розміщення сайта на хостинги та тестування.

На етапі планування замовник спільно з розробником формулюють цілі та завдання проекту, визначають його роль і цільову аудиторію, розробляють загальну стратегію просування проекту. У ході цієї роботи широко використовується бриф – документ, що складається із запитань

до замовника, відповіді на які дозволяють розробнику скласти уявлення про бачення проекту замовником. Правильно поставлене завдання дозволить розробнику врахувати особливості замовлення та оптимізувати сайт.

Цільова аудиторія – це ті люди, для яких створюється сайт, тобто відвідувачі, в яких зацікавлений власник сайта. Аналіз аудиторії необхідний для врахування їх особливостей і переваг. Від цього безпосередньо залежатиме і відвідуваність, і кількість потенційних і реальних клієнтів.

У ході спілкування із замовником, з'ясування цілей публікації і цільової аудиторії розробник формулює концепцію майбутнього сайта, в якій у довільній формі описує своє бачення результату. Зазвичай концепція включає опис ключових рішень за проектом. Концепція має бути зрозумілою та не містити спеціальних технічних деталей. Концепція узгоджується з замовником.

На основі концепції і побажань замовника розробляється технічне завдання (ТЗ). Воно слугує основою для планування розробки за термінами й оцінювання її вартості. У цьому документі чітко прописуються всі важливі технічні деталі й особливості майбутнього проекту. ТЗ містить інформацію про назву сайта, дизайн і навігацію, ширину сторінок і оптимізації їх відображення в різних браузерах і режимах, функції та інші особливості ресурсу, а також обов'язково – терміни виконання робіт. Саме ТЗ керуватимуться дизайнери, програмісти й інші фахівці в ході своєї роботи.

ТЗ повинне якнайповніше специфікувати всі елементи розробки. Це основа для планування, тестування та здачі проекту. Задача ТЗ – максимально детально визначити всі аспекти робіт на сайті, створити єдине бачення (це дуже важливо) проекту з боку замовника та виконавця. Обов'язково повинні враховуватися моменти, пов'язані з підготовкою необхідної інформації та узгодження етапів роботи, особливо ті, що виконуються різними виконавцями.

Після складання та перевірки виконавцями ТЗ затверджує замовник. Затвердження є невід'ємною частиною договору. У разі виникнення спірних або конфліктних ситуацій, що стосуються обсягів виконаних робіт, це важливий документ для врегулювання конфлікту, оскільки він є основним для розв'язування всіх питань, що виникають під час передання продукту замовнику.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) знайти в Інтернет визначення термінів "концепція", "бриф", "технічне завдання" і зразки відповідних документів.

Під час виконання лабораторної роботи необхідно:

1) вибрати й обговорити з викладачем тему своєї розробки;

2) сформулювати цілі та завдання своєї публікації;

3) сформулювати специфічні риси майбутніх відвідувачів, які будуть враховуватися під час проектування;

4) обґрунтувати структуру майбутнього сайту, сформулювати чинники, що впливають на вибір структури;

5) розробити бриф і концепцію для своєї публікації;

6) сформулювати загальну дизайнерську концепцію, обрати стиль майбутньої публікації;

7) Підготувати план контенту.

Контрольні запитання

1. Сформулюйте напрям розширення сфери використання мережі Інтернет.
2. Знання яких програмних засобів і для чого можуть знадобитися розробнику WEB-документів?
3. Опишіть процес взаємодії сервера та браузера.
4. Яку структуру може мати сайт?
5. Що таке юзабіліті?
6. Що таке бриф?
7. Які питання відображаються в концепції сайта?
8. Сформулюйте змістовність етапів реалізації проекту створення WEB-сайта.

Тема 2. Розмітка тексту з використанням HTML

Лабораторна робота 2
Розміщення тексту на WEB-сторінках

Мета – вивчення мови HTML, прийомів розмітки тексту та створення простих Web-сторінок.

Після виконання лабораторної роботи студент повинен:

знати:

вимоги до дизайну документів для сервісу WWW;

структуру HTML-документа;

програми та методи створення документів для сервісу WWW;

склад елементів, що розміщуються на документах сервісу WWW, і засоби їх створення, особливості форматів використовуваних файлів;

мову розмітки гіпертексту HTML5;

уміти:

створювати WEB-сторінки з використанням мови розмітки гіпертексту;

розміщувати на сторінках текстовий і графічний контент.

У результаті виконання лабораторної роботи у студента формуються **компетентності** зі створення WEB-сторінок і розміщення на них тексту та зображень.

Загальні відомості

Для створення Web-сторінок можуть використовуватися різні засоби: програми з візуальним середовищем створення сторінок, спеціальні редактори, а також звичайні текстові редактори (наприклад, Блокнот). У роботі з останнім рекомендується така послідовність дій:

відкрити Блокнот, ввести HTML-код сторінки;

зберегти у відповідній папці з необхідним ім'ям і розширенням htm (для початкової сторінки index.htm), вікно Блокнота не закривати;

відкрити браузер, з яким необхідно працювати;

через меню знайти та відкрити сторінку;

оцінити вид сторінки, прийняти рішення про необхідні зміни;

переключитися в Блокнот, внести зміни в код, виконати команду для зберігання;

переключитися в браузер, виконати команду для оновлення.

Три останні пункти виконувати до отримання необхідного результату.

Структура HTML-документа. Сторінці, яка відображається у вікні браузера, відповідає HTML-документ (файл, який містить розмічений текст). Структура HTML-документа – це вкладені один в один контейнери. Власне, сам документ – це один великий контейнер, який починається з тега <HTML> і закінчується тегом </HTML>.

У загальному випадку структура документа містить два контейнера (**HEAD** і **BODY**) і має вигляд:

```
<HTML>
    <HEAD>
      <TITLE> назва документа </TITLE>
    ...
    </HEAD>
    <BODY>
    ...
    </BODY>
</HTML>
```

Для розміщення на сторінці вікна, в якому відображається інша сторінка, використовується елемент "плаваючий фрейм", який може бути створений за допомогою тега <IFRAME> ... </IFRAME>, що розміщується в контейнері <BODY>. Крім обов'язкового атрибута SRC, він може містити width і height для задання ширини та висоти вікна. Контейнер цього елемента виглядає приблизно так:

```
<IFRAME SRC = "URL" width = "200" height = "400">
Ваш браузер не підтримує фрейми. Цю сторінку можна переглянути
<A HREF="URL"> ТУТ </A>
</IFRAME>
```

Заголовок документа (контейнер HEAD). Усередині тега <HEAD> ... </ HEAD> зазвичай знаходиться назва (<TITLE> ... </ TITLE>) і кілька додаткових тегів. Найчастіше це: LINK, BASE, SCRIPT, META, STYLE.

Тег LINK дозволяє визначати відношення поточного документа до інших документів і ресурсів. Наприклад, він може містити адресу файлу таблиці стилів:

```
<LINK REL=STYLESHEET HREF = "http://www.mys.com/mysheet.css" TYPE =
"text/css>
```

Елемент BASE визначає базову адресу для відносних посилань, наприклад:

```
<BASE href = "http://www.acme.com">
  ...
 <IMG SRC = "icons / logo.gif">
```

У цьому випадку зображення буде завантажуватися з файлу http://www.acme.com/icons/logo.gif. За відсутності елемента BASE в якості бази використовується місцезнаходження поточного документа.

Тег SCRIPT містить сценарій (програму мовою JavaScript або VBAScript) для створення будь-якого ефекту на екрані.

Тег META містить службову інформацію, яка не відображається під час перегляду Web-сторінки. Усередині тега містяться два атрибути, перший з яких містить ім'я (NAME або HTTP-EQUIV, воно описує тип даних, семантику), а другий (CONTENT) – зміст. З їх допомогою можна визначити деякі властивості документа: інформацію про автора, список ключових слів і т. п. Ці властивості стануть доступні для пошукових засобів і програм обслуговування.

Для NAME найчастіше зустрічаються такі значення:

Author – відомості про автора (особистий сайт);

Copyright – відомості про організацію (корпоративний сайт);

Description – опис сторінки (до 200 символів);

Keywords – список ключових слів;

Generator – інформація про засіб створення;

Revisit – необхідна періодичність індексації (число);

Robots – управління індексацією (index, noindex, follow і ін.);

URL – адреса основного "дзеркала" сайту.

Приклади:

<META NAME = "Author" CONTENT = "Molchanov">

<META NAME = "keywords" CONTENT = "Інтернет, HTML, WWW, керівництво, публікація, гіпертекст">

<Meta name = "Description" content = "Інформація про комп'ютерне залізо: тести, відгуки, коментарі експертів.">

<Meta name = "URL" content = "http://www.site.ru/">

Для http-equiv можуть використовуватися такі значення:

Content-Language – мова (ru, en, fr і т.д.);

Content-Script-Type – мова скриптів;

Content-Style-Type – мова таблиці стилів;

Content-Type – тип документа з кодуванням символів (charset = ..);

Expires – дата для управління кешуванням (якщо поточна менше, то можна брати з кешу);

PICS-Label – тип змісту для обмеження;

Refresh – затримка та перехід на інший URL.

Приклади:

<Meta http-equiv = "Content-language" content = "ru">

<Meta http-equiv = "Content-Script-Type" content = "text / javascript">

<Meta http-equiv = "Content-Type" content = "text / html; charset = windows-1251">

<Meta http-equiv = "Expires" content = "Wed, 26 Feb 1999 8:21:57 GMT">

<Meta http-equiv = "Refresh" content = "3; URL = http: //www.site.ru/">

У будь-якому місці HTML-документа, в тому числі в заголовку може міститися коментар – ігнорований браузером текст:

<! - текст коментаря ->

Коментар може бути поміщений і в спеціальний парний тег <COMMENT> текст коментаря </ COMMENT>

У цілому заголовок документа може виглядати таким чином:

```
<! - довільний коментар ->
<!DOCTYPE HTML>
<COMMENT> ще один коментар </comment>
<HTML>
<HEAD>
<TITLE> Структура Web-сторінки </title>
<LINK REL = STYLESHEET HREF = "http://www.mys.com/mysheet.css"
TYPE="text/css>
<META NAME = "Author" CONTENT = "Molchanov">
<META NAME = "Keywords" CONTENT = "WWW, HTML, document, element">
<Meta http-equiv = "Content-Type" content = "text / html; charset = windows-1251">
</HEAD>
…
```

Тіло документа (контейнер BODY). У документі дозволений тільки один тег BODY. Його атрибути діють на весь документ. За їх відсутності браузер використовує значення за замовчуванням. Основні атрибути цього тега, їх призначення і значення наведені в табл. 1.

Таблиця 1

Призначення атрибутів тега BODY

Ім'я атрибута	Значення	Призначення
background	"шлях до файлу"	Створення фонового зображення
bgcolor	#RRGGBB	Колір фона
text	#RRGGBB	Колір тексту
link	#RRGGBB	Колір посилань
vlink	#RRGGBB	Колір використаних посилань
alink	#RRGGBB	Колір останнього посилання

Колір може задаватися шістнадцятковим числом, що містить яскравість відповідних колірних компонентів (червоний, зелений, блакитний),

наприклад, # FFAA99, можна також використовувати ім'я кольору, наприклад, red, blue тощо.

За необхідності завдання одного з атрибутів BGCOLOR, TEXT, LINK, VLINK або ALINK рекомендується вказувати значення всіх. Інакше, наприклад, специфікований фоновий колір може збігтися з призначеним для користувача кольором для тексту за замовчуванням, і текст стане невидимим.

Колір тексту можна змінювати локально за допомогою відповідних тегів, однак колір фону встановити локально не можна, він є єдиним для всього документа.

Атрибути BGCOLOR і BACKGROUND можна використовувати одночасно. У цьому випадку зазвичай браузер віддає перевагу BACKGROUND, але якщо зображення фону неможливо завантажити або користувач відмовився від малюнків, буде використане значення BGCOLOR.

Розміщення тексту. Незважаючи на важливість оформлювальних елементів, основне інформаційне навантаження сторінки є текстовим. Саме текстовий зміст цікавить відвідувача в першу чергу. Тегів, що можуть містить текст, дуже багато. Основні з них наведені в табл. 2.

Таблиця 2

Теги для створення текстових елементів

Ім'я тега	Змістовність тега
P	Абзац тексту
H1, H2, H3, H4, H5, H6	Заголовки
UL, OL, DL	Списки
TABLE	Таблиця
div, article, footer, header, main, nav, section	Блок тексту
span	Текст

Вид тексту може бути змінений за допомогою тегів форматування. За використання тегів логічного форматування (табл. 3) конкретні параметри визначаються браузером.

Таблиця 3

Теги логічного форматування

Ім'я тега	Змістовність тега
EM	Виділення
STRONG	Ще більш сильне виділення
DFN	Позначає, що цей термін має визначення
VAR	Частина тексту (звичайне слово) є змінною
CITE	Назва книги або статті, яка цитується
BLOCKQUOTE	Цитування
CODE	Код програми або його еквівалент (наприклад, HTML)
KBD	Текст, що повинен друкуватися на клавіатурі користувача (звичайно використовується для інструкцій)

Теги фізичного форматування (табл. 4) дозволяють задавати конкретні параметри (вид, розмір, шрифт) безпосередньо в коді документа.

Таблиця 4

Теги фізичного форматування

Ім'я тега	Змістовність тега
TT	"Телетайпний" текст, тобто текст одного розміру
I	Курсив
B	Жирний
U	Підкреслення
STRIKE	Закреслений текст
BIG	Великий шрифт
SMALL	Малий шрифт
SUB	Підрядковий текст
SUP	Надрядковий текст
FONT	Встановлення розміру та кольору шрифту
BASEFONT	Базовий розмір шрифту

Список відрізняється від звичайного тексту тим, що з виведенням у вікні браузера виконується автоматична нумерація або маркування йо-

го елементів. Зі внесенням змін нумерація автоматично коректується. Існує кілька різновидів списків: нумеровані, ненумеровані та із визначеннями.

Ненумерований список:

```
<UL TYPE=... COMPACT>
    <LH>Заголовок
    <LI>пункт 1 списку
    <LI>пункт 2 списку
    <LI>пункт 3 списку
</UL>
```

Атрибут TYPE визначає вид маркера (disc, square, circle), COMPACT робить список більш компактним.

Нумерований список:

```
<OL TYPE= ... START=... COMPACT>
    <LH>Заголовок
    <LI>пункт 1
    <LI>пункт 2
    <LI>пункт 3
</OL>
```

Атрибут TYPE визначає стиль нумерації (1 – арабські цифри, a – малі літери, A – прописні літери, I – великі римські цифри, i – малі римські цифри), значення атрибута START визначає початкове значення послідовності.

Список з визначеннями:

```
<DL COMPACT>
    <DT>текст пункту 1
    <DD>визначення  1(виводиться зі зсувом)
    <DT>текст пункту 2
    <DD>визначення  2(виводиться зі зсувом)
</DL>
```

Елементами списку можуть бути інші теги. Списки можуть бути вкладеними.

Таблиці зручні для структурування даних, а багато Web-дизайнерів використовують їх також для точного розміщення елементів Web-сторінок у вікні браузера, або формі.

Таблиця в мові HTML задається за допомогою парного тегу <TABLE>. Вона може містити заголовок таблиці, обумовлений парним тегом <CAPTION>, і рядок таблиці, що задають за допомогою парних тегів <TR>. Закриваючі теги </TR> можна опускати.

Кожен рядок таблиці містить комірку таблиці, які можуть відноситися до двох різних типів. Комірки в заголовках стовпців і рядків задають парним тегом <TH> (текст у них буде виділений), а звичайні комірки – парним тегом <TD>. Закриваючі теги </TH> й </TD> можна опускати. Наприклад, "порожня" таблиця із двома рядками та двома стовпцями може бути задана таким чином:

```
<TABLE>
    <CAPTION>Порожня таблиця</CAPTION>
    <TR><TH><TH>
    <TR><TD><TD>
</TABLE>
```

Кожна комірка може містити довільний текст, а також будь-який інший контейнер, припустимий в "тілі" документа. Зокрема, комірка таблиці може містити вкладену таблицю або зображення. Атрибути, що визначають вигляд таблиці, наведені в табл. 5.

Таблиця 5

Призначення атрибутів, що визначають вигляд таблиці

Атрибут	Елемент	Призначення
ALIGN	Таблиця, заголовок, рядок, комірка	Вирівнювання таблиці по горизонталі; вирівнювання даних по горизонталі; розміщення заголовка над або під таблицею
VALIGN	Рядок, комірка	Вирівнювання по вертикалі
WIDTH	Таблиця, комірка	Ширина
HEIGHT	Комірка	Висота
COLSPAN	Комірка	Довжина в кілька стовпців
ROWSPAN	Комірка	Довжина в кілька рядків
BGCOLOR	Таблиця, комірка	Колір фону
CELLSPACING	Таблиця	Зазор між комірками
CELLPADDING	Таблиця	Зазор між змістом комірки і її межею
BORDER	Таблиця, комірка	Відображення межі комірок і зовнішньої рамки таблиці
BACKGROUND	Таблиця	Фоновий малюнок

Розміщення зображень. Для графічних елементів Web-сторінок використовують такі растрові та векторні формати: GIF, JPEG, PNG і SVG. Для вибору формату зображення в першу чергу беруть до уваги обсяг отриманого файлу, і в другу – якість зображення.

Зображення зберігаються в окремих файлах. Для вставки використовується тег <IMG>, що повинен містити обов'язковий атрибут SRC, що задає адресу (URL) файлу у відносній або абсолютній формі.

<IMG SRC="picture1.gif">

<IMG SRC="http://www.wwroot.picture1.gif">

Під час інтерпретації тега IMG браузер за замовчуванням використовує реальні розміри зображення. Якщо необхідно змінити масштаб, застосовують атрибути WIDTH і HEIGHT, що задають ширину та висоту (у пікселях), відповідно.

Важливим атрибутом з точки зору пошукових систем є ALT, яким задається текст, що виводиться при неможливості відобразити малюнок. Його рекомендується використовувати обов'язково.

Призначення основних атрибутів тегу <IMG> наведені в табл. 6.

Таблиця 6

Значення атрибутів тегу <IMG>

Атрибут	Значення	Примітка
WIDTH	число	Ширина в пікселях
HEIGHT	число	Висота в пікселях
ALIGN	BOTTOM MIDDLE TOP	Для розміщення зображення в рядку тексту
ALIGN	LEFT RIGHT	Для обтікання зображення текстом
HSPACE	число	Проміжок між текстом і зображенням по горизонталі
VSPACE	число	Проміжок між текстом і зображенням по вертикалі
ALT	текст	Альтернативний текст

Зображення можуть також використовуватися в якості фону сторінки, таблиці або комірки таблиці. Для цього в відповідний тег повинен бути вставлений атрибут background = "URL".

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) підготувати ескізи сторінок для створення сайтів;

3) створити необхідну кількість зображень для фону й елементів оформлення.

Під час виконання лабораторної роботи необхідно:

1) освоїти спільну роботу з додатками Блокнот і браузером:

1.1) створити сторінку з фоном і декількома довільними елементами;

1.2) доповнити сторінку мета-тегами, що містять інформацію про автора, ключові слова та зміст. Випробувати мета-тег для переадресації сторінок;

1.3) створити сторінку для використання в якості шаблону для подальшої роботи:

2) розміщення тексту:

2.1) створити сторінку, яка містить відформатований різними засобами текст (теги фізичного та логічного форматування);

2.2) створити сторінку, яка містить структурований з використанням різних засобів текст (списки, тег PRE та інші), випробувати підстановки;

3) навчитися розміщувати на сторінках зображення:

3.1) вивчити та випробувати різні засоби створення фону сторінки;

3.2) розмістити на сторінках ілюстрації, задати різну взаємодію зображення з текстом, вивчити різні режими відображення зображень;

4) навчитися використовувати таблиці:

4.1) створити таблицю з об'єднаними осередками, що містить форматований текст;

4.2) створити модульну сітку для сторінок на основі таблиці. Застосувати створену сітку для сторінки, створити в кожної області свій фон;

5) створити сторінку за ескізом на рис. 1.

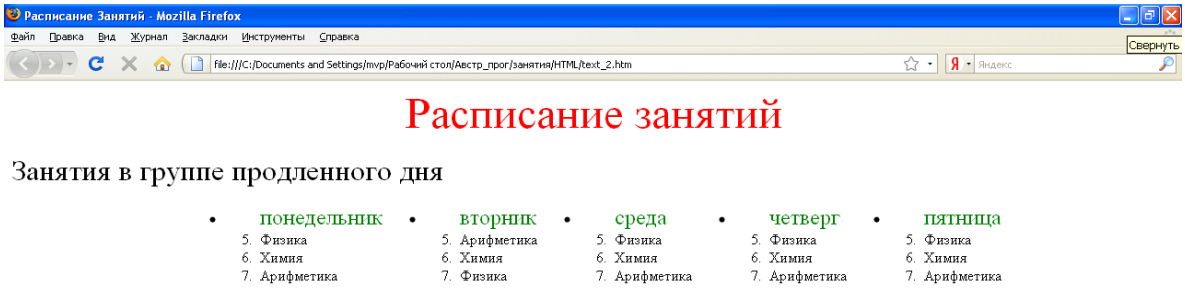

Рис. 1. **Ескіз сторінки**

Результатом виконання кожного пункту завдання повинен стати окремий файл.

Контрольні запитання

1. Назвіть основні синтаксичні конструкції мови HTML.
2. Дайте визначення понять "тег", "елемент", "контейнер".
3. Охарактеризуйте основні теги.
4. Запишіть приклад використання тегів, що визначають структуру документа.
5. В якій частині екрана виводиться напис, вміщений у тег <title> Документ </title>?
6. Назвіть засоби, що дозволяють змінювати розміри та шрифти написів, що виводяться на сторінці.
7. Які теги можна розміщувати в осередках таблиць? Запишіть тег для таблиці з трьома осередками.
8. Назвіть види списків, які можуть бути присутніми на сторінці. Записати приклад.
9. Поясніть, як розміщуються зображення на Web-сторінках? Перерахуйте основні атрибути тега <IMG>.
10. У яких тегах може бути присутнім URL файлу з рисунком? Запишіть приклад.

Лабораторна робота 3
Дослідження сторінок складної структури

Мета – вивчення засобів створення елементів навігації, прийомів розмітки тексту та створення зв'язаних Web-сторінок.

Після виконання лабораторної роботи студент повинен:

знати:

основні типи елементів навігації та засоби їх створення;

основні типи посилань та їх базування;

призначення та засоби створення форм;

правила та прийоми використання тегів <EMBED> і <OBJECT>, їх основні атрибути;

уміти:

створювати посилання, меню та карти посилань;

використовувати якорі та створювати електронний зміст;

створювати елементи інтерфейсу користувача;

розміщувати на сторінках мультимедійні об'єкти.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з розроблення сайтів і розміщення на сторінках різноманітних об'єктів.

Загальні відомості

Механізм посилань забезпечує основну властивість, що відрізняє гіпертекстовий документ від звичайного. Посилання дозволяють натисканням завантажувати у вікно зовсім інший документ, або іншу частину поточного. Для цього використовується тег <A>...</A>. Усередині нього розміщають елемент, на зображенні якого повинно виконуватися натискання для переходу. Тег містить обов'язковий атрибут HREF, значенням якого є URL, або адреса ресурсу іншого типу (електронної пошти, FTP та ін.).

<A HREF="http://www.site.com/index.htm"> клацни ТУТ</A>

<A HREF="mailto:victor@molchanov.eu.org"> пошта</A>

<A HREF="file:///D:/root/tur.gif"> подивитися малюнок</A>

Видима частина гіперпосилання може бути малюнком, **<A HREF="http://www.site.com/index.htm"> <IMG SRC="picture1.gif"></A>.**

Посилання не можуть знаходитися всередині інших. Закриваючий тег обов'язковий.

Тег <A> може містити кілька необов'язкових атрибутів, найчастіше використовуються TARGET і TITLE. Перший указує вікно, у якому буде відкрита сторінка. Значенням атрибута є ім'я вікна (для фрейму те, що вказувалося в атрибуті NAME тегу FRAME) або спеціальне значення:

_self – відкрити всередині поточного вікна;

_parent, _top – відкрити у вікні без фреймів;

_blank, new – відкрити в новому вікні.

Значенням атрибута TITLE є текст, що буде виведений наведенням курсору на посилання.

Посилання може бути виконане і на певне місце всередині сторінки. Для цього відповідне місце позначають за допомогою того ж тегу <A>…</A>, але з атрибутом NAME. Значення цього атрибута – довільна послідовність латинських літер і цифр без пробілів (ім'я), розглянута як мітка. Такий тег у літературі називають анкер або якір. Для посилання

на мітку її ім'я вказується як значення параметра HREF, але перед ім'ям повинен бути символ #. Наприклад, можна створити електронну книжку зі змістом на початку, використовуючи такий шаблон.

```
<UL>
<LI><A HREF="#gl1">РОЗДІЛ 1</A>
<LI><A HREF="#gl2">РОЗДІЛ 2</A>
<LI><A HREF="#gl3">РОЗДІЛ 3</A>
</UL>
…
<A NANE="gl1"></A>
…
<A NANE="gl2"></A>
…
<A NANE="gl3"></A>
```

Для реалізації можливості зв'язування посилань із різними частинами зображення разом з тегом IMG використовують тег MAP, що задає конкретні посилання. Ім'я цього тегу і є значенням атрибута USEMAP:

```
<IMG SRC="URL" USEMAP="#имя_карты"
...
<MAP NAME="ім'я_карти">
<AREA HREF="...." ... >
...
</MAP>
```

Усередині тегу MAP перебуває група тегів AREA, кожний з яких визначає одну область зображення та пов'язане з нею посилання.

Область може мати форму прямокутника (RECT), кола (CIRCLE) або багатокутника довільної конфігурації (POLY).

Значення координат (x та y) задаються в пікселях від верхнього лівого кута зображення (значення y зростає вниз). Координати (або одна з них) можуть бути задані також у відсотках. Відсотки беруться відповідно щодо ширини або висоти зображення. Коло задається координатами центра та радіусом, прямокутник – двох кутів, багатокутник – усіх вершин.

Із зображеннями карт зручно працювати в графічному редакторі, що підтримує координатну сітку (у найпростішому випадку Paint). Курсором виділяють потрібні точки, в рядку стану зчитують значення координат.

Карти можна використовувати для створення оригінальних меню, наприклад, у вигляді смуг уздовж вікна або з довільною геометрією, з написами, виконаними власними шрифтами.

Значення та зміст атрибутів тега AREA наведені в табл. 7.

Таблиця 7

Атрибути тегу AREA

Ім'я атрибута	Можливі значення	Зміст	Примітки
SHAPE	RECT, CIRCLE, POLY	Обрис області	За замовчуванням RECT
COORDS	Рядок або форма, обумовлена SHAPE	Координати області	Обов'язковий, за винятком використовуваного за замовчуванням SHAPE
HREF	URL	Адреса документа	Діє як гіпертекстовий зв'язок
NOHREF	NOHREF	Область не використовується	За необхідності заборонити перехід
ALT	Рядок	Текстовий опис області	Не обов'язковий

Форма на Web-сторінці є набором елементів управління. Серед них кілька видів полів: однорядкові та багаторядкові вікна для введення тексту, групи радіокнопок, квадратів і меню. Користувач у процесі роботи з Web-сторінкою заповнює форму й активізує певну процедуру обробки.

Сама обробка може полягати у відсиланні даних на один із серверів мережі або на поштову адресу. Дані можуть бути оброблені також за допомогою програм, вбудованих у Web-сторінку.

Для створення форм на сторінках використовується парний тег FORM, усередині якого містяться теги, що створюють елементи (INPUT, SELECT й TEXTAREA).

Обов'язковим є тільки action. Наприклад, для відсилання вмісту форми електронною поштою варто задати action="mailto:foo@bar.com", для відсилання на сервер – action="http://www.acme.com/cgi-bin/register.pl".

Тег INPUT можна використовувати для створення безлічі полів форм: однорядкових текстових полів, полів для введення паролів, квадратів з позначкою, радіокнопок, кнопок запуску та скидання, прихованих полів, полів завантаження файлів і зображень-кнопок. Елементи, створювані тегом INPUT, не можуть містити інших елементів. Призначення основних атрибутів тегу наведені в табл. 8.

Таблиця 8

Атрибути тегу INPUT

Ім'я атрибута	Значення	Призначення
type	text	Однорядкове текстове поле (за замовчуванням)
	password	Поле введення пароля
	checkbox	Перемикач (квадрат з позначкою)
	radio	Перемикач (радіокнопка)
	submit	Кнопка для відправлення даних
	reset	Кнопка для приведення полів у вихідний стан
	button	Кнопка для генерації події
	file	Елемент для вибору файлу у файловій системі
size	число	Розмір поля в символах
name	рядок символів	Ім'я елемента
value	рядок символів	Напис, виведений у початковому завантаженні

Атрибут name використовують для визначення імені, що буде привласнено вмісту поля у ході передавання оброблювачеві. Атрибут повинен бути присутнім у всіх елементах, дані з яких будуть відсилатись.

Атрибут value зручний для ініціалізації поля, а також для задання тексту на кнопках submit і reset.

Значення type=checkbox застосовують для створення логічних елементів, передача значення яких визначається станом "відмічено – не відмічено" (квадрат з позначкою). Таких квадратів у формі може бути кілька. В останньому випадку використовують декілька пар атрибутів "name – value" з однаковим атрибутом name та різними атрибутами value. Кожен відзначений квадрат генерує власну пару ім'я/значення в складі переданих даних, навіть якщо це призводить до появи імен-двійників. Атрибут checked потрібен для ініціалізації квадрата з позначкою. Наприклад:

```
<input type=checkbox checked name=uscitizen value=R1>
<input type=checkbox name=uscitizen value=R2>
```

Значення type=radio необхідний для створення елементів, що задають тільки одне значення з фіксованого набору альтернатив (радіокнопки). Кожна радіокнопка в групі повинна мати той самий атрибут name та різні value. Пари ім'я/значення генеруються тільки відміченою радіокнопкою. Одна радіокнопка в кожній групі повинна ініціалізуватися відміченою шляхом використання атрибута checked. Наприклад:

```
<input type=radio checked name=age value="16-21">
<input type=radio name=age value="21-35">
```

Значення type=submit визначає кнопку, на яку користувач натискає, щоб передати вміст форми на обробку. Текст на кнопці встановлюється з атрибута value. Якщо зазначено атрибут name, то пара ім'я/значення, створена кнопкою, буде включена в дані, які передаються. В одну форму можна включати кілька кнопок запуску.

Значення type=reset створює кнопку для приведення всіх полів форми у вихідний стан: <input type=reset value="СКИДАННЯ">.

Для позиціювання елементів у вікні можна використати теги <P>, <TABLE>, <PRE>. Наприклад:

```
<FORM ACTION="mailto:victor@molchanov.eu.org" METHOD=POST>
<P>введіть ім'я
<input type=text size=20 name=user value="your name">
<p>введіть пароль
<input type=password size=12 name=pw>
<TABLE CELLSPACING=40>
<TR><TD><P>задайте режим доступу
    <P><input type=checkbox name=uscitizen value=r1>режим 1
    <P><input type=checkbox checked name=uscitizen value=r2 >режим 2
  <TD><P>задайте діапазон
    <P><input type=radio checked name=age value="16-21">16-21
    <P> <input type=radio name=age value="21-35">21-35
</TABLE>
<P><input type=reset value="СКИДАННЯ">
<input type=submit value="ВІДПРАВЛЕННЯ">
</FORM>
```

Тег SELECT використовується для створення меню, з яких можна вибрати один або кілька елементів. Меню з вибором одного елемента зазвичай показується на екрані як таке, що випадає, наприклад:

```
<SELECT name="evaluation">
<OPTION value=1>Дуже вбого
<OPTION value=2>Бідно
<OPTION value=3>Середньо
<OPTION value=4>Досить добре
<OPTION value=5>Прекрасно
</SELECT>
```

Кожен обраний пункт передається оброблювачеві у вигляді пари ім'я/значення. Тег OPTION може мати атрибут selected, у присутність цього атрибута пункт меню є відміченим у процесі початкового завантаження документа. Включати цей атрибут більше ніж в один пункт меню, що допускає вибір тільки одного пункту, – помилка. Атрибут value вста-

новлює значення, що буде передане оброблювачеві з ім'ям, яке визначено атрибутом name елемента SELECT

Тег TEXTAREA використовується для створення багаторядкових полів введення тексту, введення та відсилання довгих текстів. Текст, розміщений після відкриваючого тегу, показується у вікні з первинним завантаженням документа. Число символів у рядку й кількість рядків задаються атрибутами cols й rows. Наприклад,

<TEXTAREA name=Comments rows=5 COLS=72>
Тут можете ввести більш докладний коментар
</TEXTAREA>

На закінчення ще раз підкреслимо, що для взаємодії з формою потрібна програма на сервері або клієнті. Крім того, для забезпечення додаткових дій з формами (різні ефекти та застосування) вони доповнюються скриптами.

Об'єкти, що підлягають відображенню на Web-сторінках, створюються окремо та зберігаються у файлах власних форматів.

Якщо браузер не в змозі обробити отримані дані самостійно або за допомогою вбудованих модулів, то можна запустити зовнішній додаток, який зазвичай вказується в налаштуваннях. Зовнішні додатки, наприклад RealAudio, оброблюють вміст у своєму власному вікні.

До тегів, які орієнтовані на вбудовування об'єктів, можна віднести: <OBJECT>, <EMBED>, <APPLET>, <SCRIPT>, <BGSOUND> (APPLET й SCRIPT використовують для вбудовування програм).

Тег **<OBJECT>** є універсальним і дозволяє вставляти в документ графічне зображення, відеокліп, аплет **JAVA** або елемент управління **Active;** може створювати вікно. Хоча найчастіше він використовується для вбудовування елементів Active й аплетів.

Його основні атрибути :

title – текст спливаючої підказки;

declare – об'єкт завантажується, але не активізується;

classid – ідентифікатор класу або екземпляра об'єкта;

codebase – посилання на об'єкт або базовий URL;

data – URL для завантаження необхідних даних або об'єкта;

type – тип вмісту (MIME-об'єкту) відповідно до data (може приймати значення image/gif, image/jpeg, video/mpeg, video/mp4, audio/mpeg, audio/mp4, text/css, text/html, text/xml і т. д.);

codetype – тип даних відповідно до атрибута CLASSID;

standby – текст, виведений у процесі завантаження об'єкта;

align – режим вирівнювання об'єкта щодо рядка поточного тексту або як окремого елемента (text-top, middle, textmiddle, left і т.д.);

width, height – ширина та висота об'єкта у вікні браузера;

alt – альтернативний вміст, що буде візуалізований у випадку, якщо користувач не може або не хоче відобразити об'єкт.

У ряді випадків для відображення об'єкта потрібно передати йому які-небудь параметри. Для цього використовується спеціальний тег <PARAM>, розташований всередині <OBJECT>. Тег містить три атрибути:

name – ім'я параметра;

value – значення параметра;

valuetype – тип параметра (data, object, ref).

Значеннями параметра name можуть бути:

"FileName" – URL;

"AutoStart" – {true, false};

"ShowDisplay" – {true, false};

"AnimationAtStart" – {true, false};

"TransparentAtStart" – {true, false};

"src" – URL;

"hidden" – {true, false}.

Усередині тегу можуть перебувати інші елементи (наприклад, текст, малюнок або інші об'єкти), але вони інтерпретуються тільки за неможливості активізувати об'єкт.

Тег можна використовувати для відображення малюнка (альтернатива тегу IMG):

<OBJECT data="http://www.arcus.lv/dimas/roma.jpg" type="image/jpeg" width=150 height=150></OBJECT>

Можна відкрити вікно з іншою сторінкою (альтернатива фреймам):

<OBJECT data="http://www.arcus.lv/dimas/index.html" type="text/html" width=200 height=150></OBJECT>

Відкриття фрейму:

<IFRAME SRC="..." width="..." height ="..." scrolling={YES,NO,AUTO}
frameborder={1,0}>
альтернативний текст
</IFRAME>

Ще одним тегом для вбудовування об'єктів, орієнтованим в основному на мультимедійні об'єкти, є <EMBED>. Тег може містити такі атрибути:

height – висота;

width – ширина;

autostart – можливість запуску при завантаженні (true, false);

hidden – сховати елемент керування;

loop – повторення (true, false);

hidden – сховати панель керування (true, false);

src – URL мультимедіа файлу;

pluginspage – URL плагіну;

bgcolor – фон об'єкта;

type – тип мультимедійного файлу;

alt – альтернативний зміст.

Наприклад, для вбудовування флеш-файлу тег може виглядати так:

```
<EMBED src="file.swf" menu=true quality=high bgcolor=#000066
    WIDTH=760
    HEIGHT=410 TYPE="application/x-shockwave-flash"
    PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash">
</EMBED>
```

Тег <BGSOUND> використовується для створення фонового звуку, але вважається застарілим.

Варто мати на увазі, що більшість мультимедійних файлів будуть відкриті браузером після натиснення на посилання, що містить URL мультимедійного файлу. Браузер запустить зовнішній додаток, який відповідає формату файлу. У деяких варіантах можлива видача запиту: відкрити або зберегти. Це буде ніби перехід на іншу сторінку, а наведені теги дозволяють саме вмонтувати, тобто зв'язати зі змістом відображуваної сторінки в межах вікна браузера.

Вбудовування звуку. Звук може вбудовуватися в HTML-документ по-різному, наприклад, за допомогою універсальних тегів OBJECT або EMBED. Однак для цієї мети є спеціальний тег audio.

Приклади використання:

```
<EMBED SRC=abc.mp3 HIDDEN AUTOSTART=true LOOP=true>
<EMBED SRC=abc.mp3 WIDTH=170 HEIGHT=25>
```

У першому випадку відтворення почнеться автоматично відразу ж після відкриття Web-сторінки, причому програвання буде зациклено. У другому випадку буде відображений компонент управління, що забезпечує запуск, зупинку, паузу.

```
<OBJECT DATA="ім'я файлу" TYPE="audio/формат"></OBJECT>
```

Атрибут DATA визначає файл (URL) – джерело музики, а атрибут TYPE визначає його тип (формат). Формат WAVE позначається як audio/wav, AU – audio/basic, AIFF – audio/x-aiff, mp3 – audio/mpeg.

Можливе використання таких атрибутів, як ALIGN (top, middle, bottom, left, right), WIDTH, HEIGHT, HSPACE, VSPACE.

Ім'я файлу можна передати і як параметр:

```
<OBJECT TYPE="audio/mpeg">
  <param name="FileName" value="all\dam.mp3">
  </object>
```

Для тега audio можна використовувати два варіанта вбудовування звуку:

```
<audio src="v_a/dam.mp3" controls="controls">
Ваш браузер не підтримує теги audio!
</audio>
<audio controls="controls">
  <source src="elvis.mp3" type='audio/mpeg; codecs="mp3"'>
  <source src="elvis.ogg" type='audio/ogg; codecs="vorbis"'>
</audio>
```

У першому варіанті за неможливості обробити об'єкт буде виведений напис, який міститься в тегу. У другому варіанті буде зроблено дві спроби відтворення звуку з різних джерел. Атрибут controls указує на необхідність відобразити панель управління.

Вбудовування флеш. Для відображення файлів формату swf браузер повинен містити спеціальний модуль – так званий FlashPlayer.

Саме вбудовування об'єкта в сторінку здійснюється з використанням тегів **Object** й **Embed**, причому Embed вкладається всередину Object. Цим забезпечується сумісність із найбільшим числом браузерів. Приклад коду:

```
<OBJECT classid="clsid:D27CDB6E-AE6D-llcf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shoekwave/cabs/flash/
swmyflash.cabttversion=5, 0,0,0" WIDTH=400 HEIGHT=300>
  <PARAM NAME=movie VALUE="myflash.swf">
  <PARAM NAME=quality VALUE=high>
  <PARAM NAME=bgcolor VALUE=#000033>
  <EMBED src="myflash.swf" TYPE="application/x-shockwave-flash"
   quality=high bgcolor=#000033 WIDTH=400 HEIGHT=300
PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?Pl_P
rod_Version=ShockwaveFlash">
  </EMBED>
 </OBJECT>
```

Вбудовування відео. Об'єкт відео можна вмонтувати в сторінку за допомогою тегів EMBED, OBJECT або video. Наприклад:

```
<object data="kino.avi" type="video/avi"  width="300" height="200">
  <param name="movie" value="kino.avi">
  <a href="kino.avi">скачати файл</a></p>
</object>
```

Або так:

```
 <EMBED src="kino.avi" width="300" height="200" autostart="true" loop="true">
```

Якщо використовується який-небудь нестандартний формат, то за допомогою атрибута PLUGINSPAGE бажано вказати браузеру, де знайти необхідний модуль для його показу (на той випадок, якщо він ще не встановлений). Наприклад:

```
<EMBED SRC="video.mov" HEIGHT=176 WIDTH=136 PLUGINSPAGE =
"http://quicktime.apple.com">.
```

З інших атрибутів можна звернути увагу на VOLUME, SCALE, HIDDEN (корисні тільки для відео зі звуком), AUTOPLAY (щоб почати програвання без будь-яких дій з боку користувача), CONTROLLER (для додавання до відеокартинки панелі управління), PLAYEVERYFRAME (заборонний пропуск кадрів під час програвання навіть у системі з недостатньою швидкодією; причому звук автоматично відключається) і, нарешті, TARGET. Значення більшості параметрів досить очевидні, деякі з них вимагають числових значень (наприклад, VOLUME), а деякі – булевих (TRUE або FALSE).

Тег video використовується для вбудовування таким чином:

```
<video src="pr6.webm" width="320" height="240" controls></video>

<video width="320" height="240" controls>
<source src="pr6.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"'>
</video>
```

Варто пам'ятати, що в різних браузерах ті самі значення можуть призводити до різних результатів. Тому обов'язковим є тестування сторінки в різних браузерах.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) підготувати структуру сайту, ескізи сторінок і заготівку сторінки з модульною сіткою для розміщення елементів навігації та об'єктів;

3) підготувати ескіз для карти посилань;

4) підготувати звукові файли форматів wav, mp3, невеликі відеофайли форматів mp4, wmv, avi, mpeg і необхідну кількість зображень для розміщення на WEB-сторінках .

Під час виконання лабораторної роботи необхідно:

1) створити посилання:

1.1) створити та випробувати посилання з написами та зображенням на Web-сторінці та ресурси інших типів;

1.2) створити карту посилань за власним задумом;

2) створити форму:

2.1) створити форму з використанням різних за змістом елементів (паролі, перемикачі, файли для відправки, дати і т. д.);

2.2) створити форму за власним задумом для реалізації інтерфейсу користувача (для голосування, тестування та іншого призначення);

3) дослідити засоби вбудовування об'єктів:

3.1) створити сторінку, що містить вікно на основі тегу <OBJECT>. Порівняти можливості двох способів створення вікон (iframe й OBJECT);

3.2) створити сторінку з об'єктами, що програють звукові файли різних форматів (використати й <EMBED>, <OBJECT>, <audio>);

3.3) створити сторінку з об'єктами, що програють відеофайли різних форматів (використати й <EMBED>, <OBJECT>, <video>);

3.4) дослідити відображення сайту з об'єктами в різних браузерах і розмістити власні виводи на окремій сторінці.

Примітка! *Результатом виконання всіх пунктів завдання повинен стати сайт (створити початкову сторінку та переходи з неї на групи сторінок для кожного пункту).*

Контрольні запитання

1. Які теги використовуються для створення посилань?
2. Назвіть атрибути тега form.
3. Що таке об'єкт, розташований на Web-сторінці?
4. Назвіть теги, що дозволяють відображати об'єкти.
5. Поясніть можливості та відмінності у використанні тегів <OBJECT>, <EMBED>.

6. Поясніть призначення атрибутів тегу <OBJECT>.

7. Що таке фоновий звук на Web-сторінці? Запишіть тег для створення фонового звуку.

8. Чи можна створити фоновий звук тегом <audio>? Якщо так, то яким чином?

9. За допомогою яких тегів можна вмонтувати флеш–фільм в Web-сторінку?

10. У чому відмінність відтворення відеофайлу через посилання на відповідний ресурс використанням тегу <video>?

Змістовий модуль 2. Форматування WEB-документів

Тема 3. Використання стильових специфікацій

Лабораторна робота 4
Дослідження процесу форматування контенту

Мета – вивчення властивостей елементів і засобів форматування Web-сторінок з використанням різних варіантів таблиць стилів.

Після виконання лабораторної роботи студент повинен:

знати:

синтаксис таблиць стилів;

властивості основних елементів;

прийоми вбудовування таблиць стилів у текст сторінки;

правила сумісного використання декількох таблиць стилів;

уміти:

вбудовувати таблиці стилів у HTML-документ;

форматувати текст, використовуючи таблиці стилів;

використовувати для форматування селектори різних типів.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з форматування змісту сторінок за допомогою таблиць стилів.

Загальні відомості

Технологія використання таблиць стилів передбачає кілька способів вбудовування таблиць:

таблиця стилів може бути розміщена в окремому файлі, що зв'язується з HTML-документом (тег LINK);

таблицю стилів можна вмонтувати в HTML-документ (тег STYLE);

стиль можна імпортувати з файлу в тег STYLE;

стиль можна задати для окремого елемента, помістивши у відповідний тег атрибут STYLE;

під час перегляду користувач може задати власну таблицю стилів у браузері.

Для встановлення зв'язку з таблицею використовується тег LINK, наприклад:

<LINK REL=STYLESHEET TYPE="text/css"
HREF="http://www.myserver.com/mysheet.css">.

Значення перших двох атрибутів цього тегу є зарезервованими іменами, які необхідні для того, щоб повідомити браузеру, що на цій сторінці буде використовуватися CSS. Третій атрибут – HREF указує на файл, що містить опис стилів.

Для включення таблиці стилів у документ використовується тег:

<STYLE>
опис стилів
</STYLE>

Тег може містити атрибут TYPE (text/css або text/javascript). Розміщається <STYLE>...</STYLE>, як правило, всередині тегу HEAD. Таблиця стилів створюється для кожної сторінки, однак є можливість її імпортування з файлу. Для цього посилання на джерело стилів розміщується всередину тегу STYLE таким чином:

<STYLE TYPE="text/css">
@import url(http://www.myserver.com/mystyle.css);
</STYLE>

Задання стилю для окремого елемента використовується для перевизначення стилю, визначеного в таблицях. У відповідному тегу задається атрибут STYLE (наприклад, STYLE="color:blue"). Параметри стилю, задані в атрибуті, перекривають визначені поза тегом.

Синтаксис таблиць стилів. Таблиця стилів містить набір правил (rule). Правило складається із селектора (у найпростішому випадку це ім'я тегу) та декларації (береться у фігурні дужки). Декларація сконструйована з пар <ім'я властивості>:<значення>, кілька декларацій розділяють крапкою з комою. Наприклад, H1 {color: red; font-size: 14pt; text-align: center;}

Можна об'єднати кілька селекторів, згрупувавши їх. Тоді відповідні елементи отримають однакові властивості, наприклад, P,LI {font-size: 12pt;}

Селектор визначає елементи сторінки, на які розповсюджується правило. Специфікація передбачає багато різних селекторів, однак на практиці найчастіше використовуються такі:

класові селектори (Class Selectors) – використовують значення атрибута class;

ID селектори (ID Selectors) – використовують значення атрибута id;

контекстні селектори (Contextual Selectors) – використовують відношення вкладеності елементів;

псевдокласи та псевдоелементи (Pseudo Classes, Pseudo element).

Класові селектори. Класовий селектор складається з імені тегу й імені класу, з'єднаних крапкою, наприклад: H1.bl {color:blue; size:20pt} або .bl {color:blue} діє на <H1 CLASS="bl">TEXT</H1>.

Елемент у якості значення атрибуту class може мати кілька імен, вони записуються через пробіл і розглядаються як синоніми. До такого елемента будуть застосовані правила, пов'язані з кожним ім'ям.

ID селектори. Цей селектор починається із символу #. Наприклад, #rd {color:blue; size:20pt} діє на <H1 ID="rd">TEXT</H1>.

У тегу обидва атрибути (CLASS й ID) можуть використовуватися одночасно, довизначаючи стиль елемента.

Контексні селектори складаються із простих селектрів, розподілених пробілом (усі описувані до цього селектори були простими селекторами). Вони застосовуються до елементів, зв'язаних спадкоємними відносинами, та задають властивості тільки конкретного дочірнього елемента. Наприклад, OL LI {list-style-type: decimal} задає правило для елементів LI, які вкладені у OL, а UL LI {list-style-type: square} – у UL. У разі використання елементи в нумерованому списку (OL) будуть мати один стиль, а в ненумерованому (UL) – інший.

Псевдокласи розрізняють типи одного елемента (наприклад, посилання в різних станах: активна, уже відвідувалася, ще не відвідувалася), створюючи у ході визначення власні стилі для кожного з них. Псевдокласи елемента <a href=" "> : link (посилання), active (активне посилання), visited (відвіданий раніше URL), hover (псевдоклас, що виникає піднесенням курсору до посилання). Наприклад:

a:link,a:visited {color:blue} – посилання відвідано раніше;

a:active {color:red} – активне посилання;

a:hover {text-decoration:none} – на посилання наведено курсор.

Псевдоелементи є частинами інших елементів, задаючи цим частинам відмінний від елемента в цілому стиль (наприклад, перший рядок в абзаці або перша буква). Наприклад:

P:first-line {color: purple } – перший рядок абзацу;

H1:first-letter { color: red } – перша буква.

Властивості елементів. У ході розроблення специфікації на таблиці стилів параметри, що визначають зовнішній вигляд елементів, отримали назву властивостей. Є властивості, які притаманні всім або більшості елементів, наприклад, висота та ширина. А є специфічні, наприклад, розмір шрифту для символів. Для кожної властивості специфіковане ім'я, наприклад, margin (відступ). Ряд імен є складними та записуються через риску, наприклад: margin-top (верхній відступ), border-left-width (товщина лівої рамки). Для кожної властивості визначені значення, які виражені або числом (наприклад, розмір), або зарезервованим ім'ям (right, left, none тощо). Для задання розмірів указується число з позначенням одиниці вимірювання: можна використати px (пікселі), in (дюйми), cm (сантиметри), mm (міліметри), pt (1pt = 1/72in), pc (1pc = 12pt).

Область розміщення. Область розміщення будь-якого елемента – це прямокутник з основним змістом і ряд прилеглих полів (рис. 2).

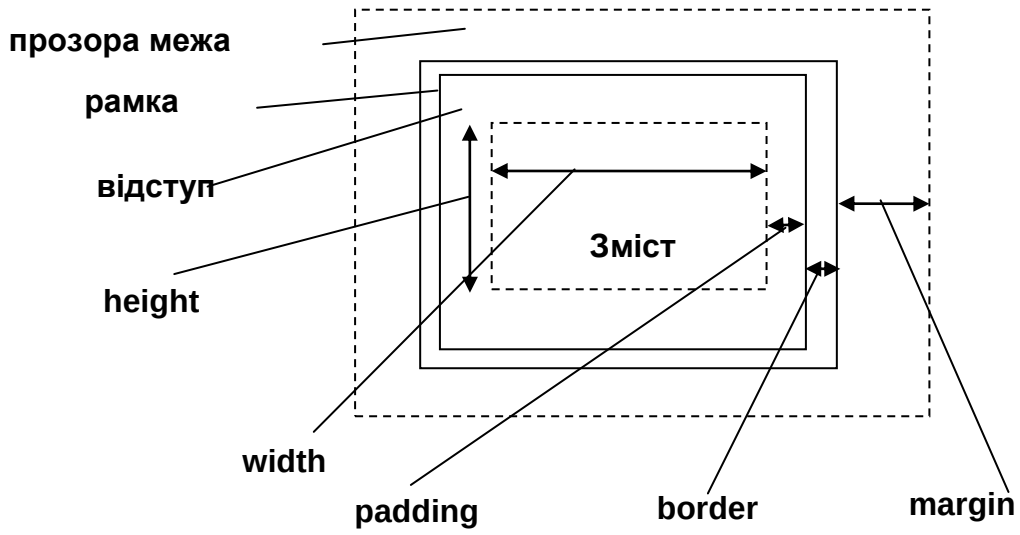

Рис. 2. **Властивості області розміщення елемента**

У табл. 9 наведені імена основних властивостей цієї групи та значення, які вони приймають.

Таблиця 9

Властивості, що характеризують область розміщення елемента

Ім'я	Значення	Примітка
margin	<число><розмірність> або <число>% або auto	Розмірність – px, in, cm, mm, pc (пікселі, дюйми, см, мм, пункти), наприклад, 25mm; % від базового розміру
padding		
border-width		
border-color	black, coral, orange тощо або через rgb	#rrggbb (наприклад, #00cc00) rgb(x,x,x), де x=0...255, rgb(x%,x%,x%), де x=0.0...100.0
border-style	none, dotted, dashed, solid, double, groove, ridge, inset, outset	Різні варіанти рамок: відсутня, пунктирна, штрихова тощо

У разі вживання імен у такому вигляді вони позначають всю область. За необхідності слід указати одну зі сторін, для чого використовується складене ім'я (padding-left, border-left, border-left-color і т. д.).

Позиціювання елементів. Властивості управління позиціюванням забезпечують можливості маніпулювання розташуванням елементів усередині вікна або батьківського контейнера. Основні з них наведені в табл. 10.

Таблиця 10

Властивості, використовувані для позиціювання

Ім'я	Значення	Примітка
display	block, inline, inline-block, list-item, flex, none	визначає тип елемента
position	absolute, fixed, relative, static	Задає систему позиціювання
top, right, left, bottom	<число>	Положення елемента
float	right, left, none	Обтікання
z-index	<число>	Порядок відображення
visibility	visible, hidden, collapse	Управління видимістю
overflow	auto, hidden, scroll, visible	Режим відображення вмісту

Властивість display визначає, як елемент треба показати у документі стосовно сусіднього в потоці. Специфікація CSS визначає багато можливих значень для цієї властивості (block, inline, inline-table, list-item, none, table, flex і т. д.), однак більшість браузерів підтримують такі:

block – елемент показується як блочний; відбувається перенесення рядків на початку та наприкінці вмісту, тобто в його рядках навіть за наявності місця інші елементи не розміщуються;

inline – елемент відображається як вбудований у рядок; вміст елементів починається з того місця, де закінчився попередній;

inline-block – створюється блочний елемент, який обтікається іншими елементами, його внутрішня частина форматується як блоковий елемент, а сам елемент – як вбудований;

list-item – елемент виводиться як блочний і додається маркер списку;

flex – автоматичне розташування дочірніх елементів у блоку;

none – тимчасово видаляє елемент із документа; займане їм місце не резервується; Web-сторінка формується так, ніби елемента не було.

Властивість position визначає спосіб позиціювання елемента щодо вікна браузера або батьківських об'єктів на Web-сторінці. Може приймати значення:

absolute – елемент видаляється зі звичайного потоку документа подібно іншим елементам; його положення задається атрибутами left, top, right й bottom щодо краю вікна браузера;

fixed – це значення схоже на absolute, прив'язується до зазначеної параметрами left, top, right й bottom точці на екрані, але не змінює свого положення навіть з прокручуванням сторінки; різні браузери реагують на цю властивість дуже по різному;

relative – положення елемента встановлюється щодо його вихідного місця; додавання атрибутів left, top, right й bottom змінює позицію елемента, зрушуючи його в той або інший бік від первісного розташування залежно від застосовуваного параметра;

static – елементи відображаються як звичайно, використання параметрів left, top, right й bottom не змінює місця розташування. За замовчуванням – static.

Властивості фону. Управління фоном забезпечує група властивостей background, вони застосовуються до всіх елементів і не успадковуються (табл. 11).

Таблиця 11

Властивості для управління фоном

Ім'я	Значення	Примітка
background	<набір значень>	Узагальнена властивість
background-attachment	fixed, scroll	Режим прокручування
background-color	<колір>, transparent	Кольори фону
background-image	url(шлях до файлу), none	Фоновий малюнок
background-position	left, center, right, top, bottom	Розташування малюнка
background-repeat	repeat-x, repeat-y, no-repeat	Повторюваність малюнка

Колір фону задається так, як у тегах: або ім'ям (наприклад, "red"), або шістнадцятковим значенням (наприклад, #6669FF), або за допомогою функції rgb, наприклад: rgb(100,100,150), або rgb(50%,60%,50%). За значенням transparent (використовується за замовчуванням) установлюється прозорий фон, через який видно фон батьківського елемента.

Властивості текстових елементів. Для управління зображенням текстових елементів використовують дві групи властивостей font (табл. 12) і text (табл. 13).

Таблиця 12

Властивості шрифтів

Ім'я	Значення	Примітка
font	<набір значень>	Узагальнена властивість
font-family	<список імен>	Список імен шрифтів або сімейств
font-size	xx-small, x-small, small	Розмір шрифту
font-style	normal, italic, oblique	Накреслення шрифту
font-variant	normal, small-caps	Подання малих літер
font-weight	normal, bold, bolder, lighter, 100, 200 ...	Насиченість шрифту

Таблиця 13

Властивості тексту

Ім'я	Значення	Примітка
letter-spacing	<число>, normal	Інтервал між символами
line-height	<число>, <число>%, normal	Міжрядковий інтервал
text-align	left, right, center, justify	Горизонтальне вирівнювання
text-decoration	none, underline, overline, line-through, blink	Варіант оформлення
text-indent	<число>, <число>%	Відступ першого рядка
vertical-align	baseline, sub, super, top-text, top, middle, bottom, bottom-text, <число>%	Вирівнювання по вертикалі щодо батька або навколишнього тексту
word-spacing	<число>, normal	Інтервал між словами

В цілому властивості тексту схожі на аналогічні параметри стилів в текстових редакторах.

Властивості списків. Є окрема група властивостей для управління видом списків. Властивості цієї групи застосовні до елементів зі значенням list-item для властивості display (табл. 14). Усі вони успадковуються.

Таблиця 14

Властивості списків

Ім'я	Значення	Примітка
list-style	<набір значень>	Узагальнена властивість
list-style-image	url ("шлях"), none	Зображення маркера
list-style-position	outside, inside	Розміщення маркера
list-style-type	disc, circle, square, ...	Вид маркера

Таким чином засоби форматування текстових елементів дозволяють ефективно управляти поданням текстових даних на сторінці.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) підготувати необхідні елементи оформлення.

Під час виконання лабораторної роботи необхідно:

1) дослідити особливості спільної дії декількох таблиць стилів:

1.1) створити набір різних таблиць стилів, що відрізняються, наприклад, кольором для абзаців. Дослідити взаємодію таблиць (зовнішня, вбудована, імпортована, в тегу, користувальницька) у різних сполученнях;

1.2) випробувати властивість !important для зміни ваги визначень;

2. вивчити властивості блочних елементів:

2.1) створити кілька блочних елементів з різними параметрами області розміщення, підібравши їх таким чином, щоб візуально оцінити кожен параметр і взаємне розташування елементів;

2.2) дослідити дії різних браузерів за зміною розмірів вікна та вмісту з різними значеннями display, visibility, overflow, position;

2.3) створити сторінку, яка містить дві (три) колонки, з використанням властивості display: flex;

3) вивчити властивості рядкових елементів:

3.1) створити елемент SPAN із властивістю display:inline, випробувати розміщення в ньому різних елементів (текст, зображення);

3.2) створити власний оригінальний стиль для абзацу та для заголовка другого рівня, використовуючи властивості групи FONT;

3.3) створити власний оригінальний стиль для абзацу й для заголовка третього рівня, використовуючи властивості групи TEXT;

4. вивчити використання різних селекторів:

4.1) створити таблицю стилів, що містить набір правил на основі селекторів усіх типів та їх комбінацій, випробувати дію;

4.2) створити сторінки з різними варіантами використання фону (розмножити, центрувати, непрокручуваний і т. д.) у різних елементів;

5) позиціювати елементи:

5.1) створити сторінку з декількома елементами, накладеними один на інший. Випробувати різні значення властивості z-index для управління видимістю елементів;

5.2) створити сторінку з абсолютно позиціонованими елементами, випробувати управління видимістю;

5.3) створити елемент для відображення флеш-фільму, який розташований незалежно від прокручування (випробувати в різних браузерах).

Контрольні запитання

1. Що таке селектор?
2. Сформулюйте рекомендації щодо використання селекторів CLASS та ID.
3. Що таке правило?
4. Сформулюйте основні вимоги синтаксису таблиць стилів.
5. Чи можуть декілька тегів мати однакові значення атрибута ID?
6. У чому полягає основна властивість блочних елементів?
7. Поясніть призначення властивості float. Які значення вона може приймати?
8. Що таке узагальнена властивість? Як записати її значення?
9. Сформулюйте рекомендації з використання таблиць стилів. У яких випадках та які використовувати (зовнішні, у тегах і т. д.)?
10. Чи може сторінка містити декілька тегів STYLE?
11. Чи може тег містити декілька атрибутів STYLE?
12. Поясніть який текст буде отриманий у результаті застосування правила P {font: bold italic large Palatino, serif }? Запишіть у вигляді набору значень для окремих властивостей.

Тема 4. Форматування за допомогою CSS

Лабораторна робота 5
Форматування сторінок з використанням таблиць стилів

Мета – вивчення засобів і прийомів форматування Web-сторінок з використанням таблиць стилів.

Після виконання лабораторної роботи студент повинен:

знати:

властивості основних елементів;

прийоми створення різних ефектів за допомогою таблиць стилів;

уміти:

форматувати зміст WEB-сторінки за допомогою таблиць стилів;

створювати ефекти, використовуючи таблиці стилів;

управляти розміщенням елементів з використанням таблиць стилів.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з форматування змісту сторінок за допомогою таблиць стилів.

Загальні відомості

Псевдокласи. Псевдоклас є доповненням до селектора, яке визначає певний стан елемента (отримання фокусу, наведення курсору, порядок елементів і т. п.). Записується через двокрапку (наприклад, h1: hover). Завдяки цій конструкції можна виділяти підмножини елементів, що знаходяться в одному стані, та змінювати форматування за динамічної зміни стану. Специфікація CSS передбачає більше тридцяти псевдокласів, однак частина з них не підтримується браузерами.

На практиці найчастіше використовують такі:

link – посилання, які не відвідувалися;

visited – посилання, які відвідувалися;

active – активний стан елемента (наприклад, посилання шляхом натискання кнопки мишки);

hover – стан елемента з наведенням курсору;

target – елемент, який обраний у посиланні за id (наприклад, перша вкладка – посилання на елемент з id="anchor");

focus – отримання фокусу елементом (наприклад, установка курсору в текстовому полі);

first-child – перший дочірній елемент;

last-child – останній дочірній елемент;

only-child – єдиний дочірній елемент;

nth-child(n) – дочірній елемент номер n, наприклад, p:nth-child(4);

not() – виключення елементів, селектор яких вказаний як параметр;

empty – порожні елементи (без дочірніх);

first-of-type – перший елемент даного типу;

last-of-type – останній елемент даного типу;

checked – відмічені чекбокси та вибрані радіокнопки.

Псевдокласи active, hover, target, focus можуть використовуватись для динамічного змінювання стилів (у відповідь на дії користувача мишкою). Наприклад, із заданням div: active {background-color: red;} елементи div будуть змінювати колір фону на червоний у момент натискання на них правою кнопкою мишки.

Переходи. Переходи дозволяють створювати ефект плавної зміни значень CSS-властивостей (анімувати зміну). Для цього можна встановлювати список змінюваних властивостей, їх значення, управляти швидкістю зміни значень властивостей.

Зміна властивостей відбувається з настанням певної події, яка описується відповідним псевдокласом. Найчастіше використовується hover.

Для управління переходом використовуються властивості:

transition-property:"список властивостей";

transition-duration:"тривалість переходу";

transition-timing-function:"функція зміни часу переходу";

transition-delay:"затримка переходу";

transition:"узагальнене властивість".

Приклад стилів, що забезпечують зміну двох властивостей (background і border-radius) з наведенням курсору:

```
.box {  background: # 2db34a;
        border-radius: 6px
        transition-property: background, border-radius;
        transition-duration: 1s;
        transition-timing-function: linear;}
.box: hover {background: # ff7b29;
        border-radius: 50%;}
```

Такий же перехід з використанням узагальненої властивості:

```
.box {background: # 2db34a;
      border-radius: 6px;
      transition: background .2s linear, border-radius 1s ease-in 1s;}
.box: hover {color: # ff7b29;
      border-radius: 50%;}
```

Анімація. Для переходів використовують два стани властивостей: початковий і кінцевий. Для змін з великим числом станів застосовується анімація. Для опису проміжного стану є ключові кадри, які містять значення властивостей. Рекомендується анімувати властивості з числовими значеннями. Опис кадрів поміщається в правило @keyframes ІМ'Я {кадри}. Кожен ключовий кадр ідентифікується числом з %, що означає положення кадру на осі часу. ІМ'Я слугує для зв'язку правила з загальними властивостями анімації:

animation-name:"ІМ'Я;"
animation-duration:"тривалість";
animation-timing-function:"функція зміни часу";
animation-delay:"затримка початку";
animation-iteration-count:"число циклів";
animation-direction:"напрямок розвитку анімації";
animation-fill-mode:"стилі до початку анімації і після її завершення".

Властивості можуть бути об'єднані в узагальненій – animation. Наприклад, такі правила забезпечать рух об'єкта (абзацу тексту) по периметру прямокутника:

```
@keyframes around {
  0% {left: 0; top: 0; }
  25% {left: 240px; top: 0; }
  50% {left: 240px; top: 140px; }
  75% {left: 0; top: 140px; }
  100% {left: 0; top: 0; }}
p {animation: around 4s linear 0s infinite;
  background: turquoise;
  color: white;
  height: 60px;
  line-height: 60px;
  margin: 0;
  position: absolute;
  text-align: center;
  width: 60px;}
```

Трансформації. Трансформації є змінами виду (перетворення) елементів під час їх відображення в браузері. Зміни зводяться до таких про-

цедур: зсув, масштабування, обертання, спотворення (нахил об'єкта, зміна нахилу осей).

Трансформації відбуваються щодо точки, яка за замовчуванням знаходиться в центрі елемента (50 %, 50 %). За допомогою властивості transform-origin можна змінити положення центру трансформації.

Основні функції трансформації:

translate(), translateX(), translateY() – зміщення, в параметрах можна використовувати одиниці довжини (позитивні і негативні),%;

scale(), scaleX(), scaleY() – масштабування, параметр – число;

rotate() – поворот на заданий кут (deg, grad, rad або turn);

skew(), skewX(), skewY() – спотворення, задається кут від відповідної осі (deg, grad, rad).

Допускається використовувати кілька перетворень за раз, наприклад, поворот і масштабування розміру елемента одночасно. Для їх об'єднання значення задаються списком у властивості transform один за іншим без використання ком.

Наприклад, властивості transform-origin: 0 0; transform: translate (100px, 100px) rotate (30deg) забезпечать розміщення елемента зі зміщенням 100 пікселів по кожній осі та поворотом на 30 градусів щодо верхнього лівого кута області елемента.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) підготувати необхідні елементи оформлення.

Під час виконання лабораторної роботи необхідно:

1) форматувати сторінки за зразком:

1.1) створити сторінку вигляду, показаного на рис. 3, розмістивши всі стильові специфікації в зовнішньому файлі;

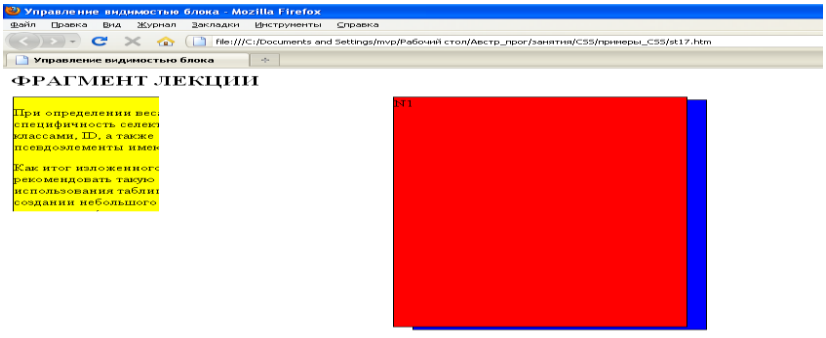

Рис. 3. **Вигляд сторінки у вікні браузера**

1.2) створити HTML-сторінку, яка містить текст і таблицю стилів, що забезпечує його подання у вигляді, наведеному на рис. 4;

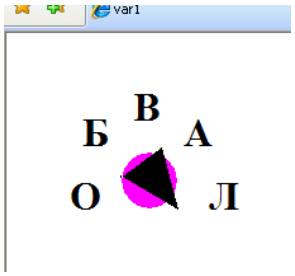

Рис. 4. **Вигляд сторінки у вікні браузера**

2) створити ефекти за допомогою CSS:

2.1) створити контейнер з текстом, розміщеним у двох колонках, трьох колонках (запропонувати кілька варіантів реалізації);

2.2) створити ефект для елементів з використанням псевдокласу target і переходів;

2.3) розробити та реалізувати власний ефект з використанням CSS (анімації або трансформації);

3) створити сторінки з використанням шаблонів:

3.1) створити сторінку, використовуючи один із вбудованих шаблонів (MS Expression), розмістити власний контент, оформити відповідно до власних уявлень про дизайн;

3.2) створити власну сторінку із запропонованого викладачем шаблону (або знайденого у мережі).

Контрольні запитання

1. Охарактеризуйте ефекти, які можна створювати за допомогою CSS.
2. Сформулюйте рекомендації щодо використання селекторів CLASS і ID.
3. Чи можуть кілька тегів мати однакові значення атрибута ID?
4. У чому полягає основна властивість блокових елементів?
5. Поясніть призначення властивості float. Які значення вона може приймати?
6. Що таке узагальнена властивість? Як записати її значення?

7. Чи можна позиціонувати елементи SPAN?

8. Чи може бути присутнім фон у стічних елементів?

9. Назвіть кілька застосувань для управління видимістю елементів.

10. Чи викликає зміна видимості елементів зміну в їх взаємному розташуванні?

11. Які властивості мають значення repeat-y? hidden?

Змістовий модуль 3
Створення динамічних WEB-сторінок

Тема 5. Верстання сторінок

Лабораторна робота 6. Верстання WEB-сторінок

Мета – вивчення засобів і прийомів верстання Web-сторінок з використанням таблиць стилів.

Після виконання лабораторної роботи студент повинен:

знати:

правила взаємодії елементів на сторінках;

особливості розміщення текстових і блокових елементів на сторінці;

властивості, що забезпечують позиціонування елементів;

підходи до верстання;

засоби адаптивного верстання;

уміти:

виконувати верстання сторінок відповідно до дизайнерського макету;

використовувати технологічні засоби верстання;

управляти розміщенням окремих елементів на сторінці.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з верстання WEB-сторінок за допомогою сучасних засобів.

Загальні відомості

Верстання сторінок на основі макета. Верстання сторінок виконується на основі макета, створеного дизайнером. Макет визначає зовнішній вигляд сторінки, розташування елементів з контентом, необхідне оформлення (фонові малюнки, зображення, логотип, типографіку і т. д.).

Контент розміщується в блоках, що займають усю ширину вікна (смугах) або його частину (колонках). Число колонок зазвичай не буває більше трьох.

Додатково задаються вимоги до поведінки елементів зі зміною умов перегляду. Сформована практика передбачає кілька підходів до реалізації поведінки елементів зі зміною розмірів вікна. Найчастіше використовують фіксоване та гумове верстання або їх комбінацію.

Для фіксованого верстання всі розміри задаються в пікселях. Завдяки цьому сторінка буде точно відповідати макету.

У гумовому верстанні частина розмірів задається у відносних одиницях, завдяки чому ці розміри будуть змінюватися зі зміною розмірів вікна (батьківських елементів). У якості відносних одиниць використовують відсотки (%), соті частки ширини та висоти вікна перегляду (vw, vh), розміри шрифту (em, rem). З використанням % значення обчислюється щодо належної якості батьківського елемента, em – від властивості font-size батька, rem – від властивості font-size кореневого елемента (html), vw, vh – від ширини та висоти вікна.

Для запобігання некоректного відображення сторінки у разі надмірного збільшення або зменшення елементів на дуже великих або малих екранах використовують властивості min-width, max-width, min-height, max-height. Вони обмежують максимальні та мінімальні розміри елементів за умови занадто великих змін розмірів вікна перегляду (наприклад, на дуже великих екранах).

Для створення колонок з контентом і для управління їх розміщенням використовуються обтічні блоки (властивості float і clear) або блоки з display: flex.

Серед верстальників традиційним підходом до створення колонок було використання властивостей float (обтікання) і clear (заборона обтікання). Для додання макету необхідних властивостей доводилося застосовувати спеціальні штучні прийоми. Підтримка в браузерах властивості display: flex дозволяє виконати верстання, поклавши вирішення всіх питань на браузер.

Наприклад, наступний фрагмент коду створить на сторінці три колонки.

```
...
<style>
 div {border: solid 1px # e7e7e7; padding: 12px;}
.grid_row {display: flex; justify-content: center}
```

```
.grid_item {flex: 1 0 200px; }
</style>
<body>
  <div class = "grid_row">
    <div class = "grid_item">
      Блоки можна об'єднати в рядки або розбити на колонки.
      Для кожного flexbox-елемента можна задати певний
      порядок.
    </ div>
    <div class = "grid_item"> 2 </div>
     <div class = "grid_item"> 3 </div>
 </div>
…
```

Використовувані властивості та прийняті ними значення наведені в табл. 15

Таблиця 15

Властивості блоків для створення колонок

Ім'я властивості	Набути значення	Дія
min-width	число	Обмежує мінімальну ширину
max-width	число	Обмежує максимальну ширину
min-height	число	Обмежує мінімальну висоту
max-height	число	Обмежує максимальну висоту
float	left, right	Робить елемент обтічним
clear	left, right, both	Початок заборони обтікання
display	flex	Автоматичне розташування дочірніх блоків
flex-direction	row, column, …	Напрямок осей
justify-content	flex-start, flex-end, center, …	Тип вирівнювання дочірніх блоків на головній осі
align-content	flex-start, flex-end, center, …	Тип вирівнювання дочірніх блоків на другій осі
flex-grow	число	Ступінь збільшення ширини дочірнього блоку
flex-shrink	число	Ступінь зменшення ширини дочірнього блоку
flex-basis	число px	Базова ширина дочірнього блоку

Адаптивне верстання сторінок. Зі зростанням різноманітності пристроїв, з яких користувачі виходять у мережу Інтернет, а особливо з поширенням мобільних пристроїв, на які припадає все більша частина запитів на WEB-сторінки, істотно змінюються умови перегляду сторінок. Різниця в характеристиках екранів мобільних пристроїв, настільних комп'ютерів і телевізорів (на них теж можна переглядати WEB-сторінки) настільки значна, що забезпечити однаковий вигляд сторінок (кросплатформеність) неможливо. Тому вимога кросплатформеності верстання поступилась більш універсальній вимозі адаптивності або чутливості (responsive). За адаптивного підходу в разі зміни умов перегляду можуть змінюватися не тільки розміри, але і розташування, зовнішній вигляд і навіть склад елементів.

Для реалізації адаптивного підходу можуть бути використані різні засоби з арсеналу WEB-технології:

створення окремих версій WEB-сайту для окремих видів пристроїв (різні адреси);

метатеги (з атрибутом name = "viewport");

генерація різних версій сайту після аналізу запиту на боці сервера;

скрипт на боці клієнта;

використання медіазапитів (CSS3 Media Queries);

фреймворки (наприклад, BootStrap).

Найбільший інтерес для верстання становлять два останніх (модуль Media Queries специфікації CSS3 і фреймворки).

Основною особливістю браузерів мобільних пристроїв (за відсутності метатега з атрибутом name = "viewport") є відображення сторінок з тією ж шириною, що і на екрані ПК (зазвичай вона складає близько 980 пікселів). Після цього браузер змінює розмір шрифтів і контенту, щоб вмістити сторінку на екрані. У результаті користувачі бачать різні шрифти на різних пристроях, і для роботи з сайтом їм доводиться збільшувати його масштаб.

Медіазапити дозволяють використовувати для різних умов перегляду різні правила CSS (таблиці стилів): це фільтри, які можна застосовувати до стилів CSS. Вони дозволяють вибирати стилі на підставі характеристик пристрою, пов'язаних з відображенням контенту, включаючи тип, ширину, висоту, орієнтацію, роздільність і навіть глибину кольору. За формою це умовні конструкції, що управляють застосуванням правил, заданих у таблицях стилів. Вони можуть застосовуватись як до файлів із CSS у цілому, так і до окремих правил.

Сама умовна конструкція містить логічний вираз, складений зі змінних, відповідних типів пристроїв (табл. 16) і медіафункцій (табл. 17), які задають окремі характеристики, з'єднані логічними операторами (and, not ","). Наприклад, вираз all and (min-width: 480px) буде істинним для будь-якого пристрою, ширина області перегляду якого більше 480 пікселів. Якщо вираз істинний, то буде використаний відповідний файл <link rel="stylesheet" media="all and (min-width: 480px)" href = "small.css" /> або правило @media all and (min-width: 480px) {width: 100%}.

Таблиця 16

Імена типів пристроїв

Ім'я типу пристрою	Опис
all	Усі типи
braille	Пристрої, засновані на системі Брайля
embossed	Принтери, що друкують за системою Брайля
handheld	Смартфони й аналогічні їм апарати
print	Принтери та інші друкувальні пристрої
projection	Проектори
screen	Екран монітора
speech	Мовні синтезатори
tty	Пристрої з фіксованим розміром символів
tv	Телевізори

Таблиця 17

Основні медіафункції

Ім'я функції	Опис
max-width	Ширина не більше ніж
min-width	Ширина не менше ніж
max-height	Висота не більше ніж
min-height	Висота не менше ніж
max-color	Глибина кольору не більше ніж
min-color	Глибина кольору не менше ніж
max-resolution	Роздільна здатність не більше ніж
min-resolution	Роздільна здатність не менше ніж
orientation	Орієнтація (landscape, portrait)

За допомогою медіазапитів можна створювати адаптивні сторінки, тобто сторінки, які будуть по-різному відформатовані для відображення на різних за характеристиками пристроях. Наприклад, наступний фрагмент таблиці стилів забезпечить використання шрифтів різного розміру в заголовках на різних за розмірами екранах:

```
@media all and (max-width: 760px) {h1 {font-size: 12pt}}
@media all and (min-width: 760px) and (max-width: 1200px) {h1 {font-size: 16pt}}
@media all and (min-width: 1200px) {h1 {font-size: 20pt}}
```

Таким же чином можна адаптувати сторінку, змінюючи за необхідності ширину, кількість колонок, розміри зображень, тексту та інші властивості. Використовуючи властивості visibility та display, можна управляти відображенням елементів.

Для створення дизайну, що дозволяє найкращим чином відображати сайт на різних пристроях, можна рекомендувати такі дії для адаптації сторінки за допомогою медіазапитів:

зменшення кількості колонок (стовпців) і поступове скасування обтікання у міру зменшення екрану пристрою;

використання властивостей max-width і min-width, щоб не допускати надмірного збільшення та зменшення ширини блоків контейнерів;

зменшення полів і відступів на мобільних пристроях (наприклад, нижніх відступів між заголовком і текстом, лівого відступу для списків і т.п.);

вибір прийнятного розміру шрифту для мобільних пристроїв;

перетворення лінійних навігаційних меню в таки, що розкриваються;

приховування другорядного вмісту на портативних пристроях за допомогою display: none;

підключення фонових зображень зменшених розмірів для мобільних пристроїв.

Наприклад, наступний фрагмент коду створить на екрані більшому за 900px сторінку з чотирма колонками. На екрані >600px, але <900px – з двома колонками в рядку. На екрані <600px – контент розміститься в одну колонку.

```
...
<style type = "text / css">
div {border: solid 1px # e7e7e7;}
@media screen and (min-width: 900px) {
    #sections_left {float: left; width: 24%;}
    #main {float: left; width: 40%;}
    #sections {float: left; width: 20%}
    #news {float: right; width: 15%;}        }
```

```
@media screen and (max-width: 900px) {
    #sections_left {float: left; width: 39%;}
    #main {float: left; width: 60%;}
    #sections {clear: both; float: left; width: 49%}
    #news {float: right; width: 50%;}       }
@media screen and (max-width: 600px) {
    div {min-width: 300px}
    #sections_left {width: 100%;}
    #main {width: 100%;}
    #sections {width: 100%}
    #news {width: 100%;}                    }
</style>
…
<body>
<div id = "sections_left">
...
</div>
<div id = "main">
...
</div>
<div id = "sections">
...
</div>
<div id = "news">
...
</div>
```

Верстання сторінок з використанням фреймворків. Використання фреймворків фахівцями для розроблення коду можна вважати загальною тенденцією. Не є винятком і верстальники. Розроблено велику кількість продуктів даного класу, орієнтованих на CSS і HTML коди, наприклад: Blueprint, 960 Grid System, Bootstrap, css-framework, Topcoat і т. д. Найбільш популярним є Bootstrap. Він сумісний з усіма основними сучасними браузерами і підтримує адаптивність.

Ефект від його використання досягається за рахунок готових таблиць, що містять різні правила для елементів сторінки. Розробнику достатньо створити код розмітки та застосувати необхідні імена класів (значення атрибута class).

У використанні фреймворка значення атрибута class у тегах є набором імен, розділених пробілами. Будь-яке з них, використане в класовому селекторі, зв'яже правило з цим елементом. Такий підхід дозволяє легко додавати власне форматування. Достатньо написати правило та додати через пробіл ім'я в значення атрибуту.

Для використання фреймворка необхідно підключити таблицю стилів і дві бібліотеки, завантажені з сайту розробника:

```
<link href = "css/bootstrap.min.css" rel = "stylesheet">
<script src = "js/jquery.min.js"> </script>
<script src = "js/bootstrap.min.js"> </script>
```

Вміст, який форматуватиметься з використанням цих правил, має перебувати в контейнері з класом container або container-fluid (для гумового макета).

Заготівля сторінки для верстання з використанням Bootstrap може виглядати так:

```
<!doctype html>
<html>
  <head>
   <meta charset = "utf-8">
   <title> Заготівля </title>
   <link href = "css/bootstrap.min.css" rel = "stylesheet">
  </head>
  <body>
   <div class = "container"> / * container-fluid для гумового макета * /
...
   </div>
   <script src = "js/jquery.min.js"> </script>
   <script src = "js/bootstrap.min.js"> </script>
  </body>
</html>
```

Основною моделлю розміщення контенту є послідовність смуг, розділених на колонки. Колонка може займати як всю смугу, так і її частину (смуга може містити до дванадцяти колонок, а ширина колонки задається в числі 1/12 часток від ширини смуги).

Адаптація до ширини екрану забезпечується використанням імен екранів (xs – смартфон, менше 768px; ms – планшет до 992px; md – настільний комп'ютер, до 1200px; lg – великий екран, понад 1 200px).

Для вказівки ширини колонок і адаптації її до розміру екрана використовують імена, що складаються з трьох частин (col, ім'я екрану, число часток ширини), наприклад, col-md-5 (означає, що ширина колонки буде 5/12 від ширини смуги на моніторах з розміром до 1 200 пікселів).

З умовами перегляду можна також зв'язати видимість елемента, для цього використовують класи hidden і visible. Наприклад, hidden-xs приховає елемент на смартфонах, а visible-lg забезпечить видимість тільки на великих екранах.

За структурою контент повинен відповідати моделі. Наприклад, необхідно створити сторінку такого вигляду, як на рис. 5. Причому заголовок і підвал повинні займати всю ширину, а колонки адаптуватися (на смартфонах – на всю ширину, на планшетах – по 6/12, на інших – 4/12 і 8/12).

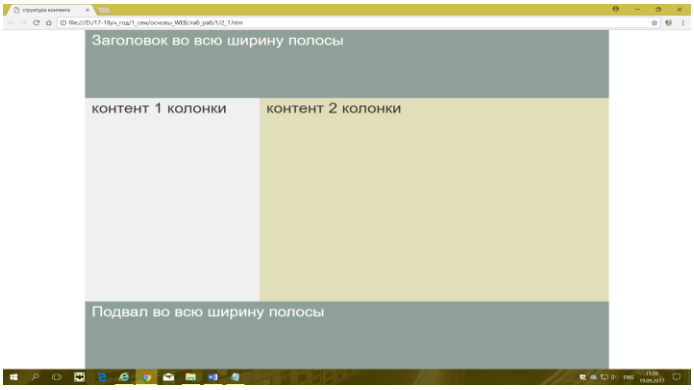

Рис. 5. **Розміщення елементів сторінки**

Контент повинен бути структурований так:

```
<div class="container my_con"> /*container-fluid   для гумового макета*/
   <div class="row my_header">
      <div class="col-xs-12">Тема на всю ширину смуги</div>
   </div>       <div class="row my_col">
      <div class="col-xs-12 col-sm-6 col-md-4 my_1col"> контент 1 колонки</div>
      <div class="col-xs-12 col-sm-6 col-md-8 my_2col"> контент 2 колонки</div>
                  </div>
   <div class="row my_footer">
      <div class="col-xs-12"> Підвал під всю ширину смуги</div>
   </div>
</div>
```

Крім сітки фреймворк містить стилі для форматування більшості елементів, які зустрічаються на сторінках.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) підготувати необхідні елементи оформлення.

Під час виконання лабораторної роботи необхідно:

Обов'язково! *При виконанні всіх пунктів завдань зі створення сторінок на основі ескізів має бути використане власне оформлення!*

1) верстання сторінок на основі макета:

1.1) зверстати горизонтальне меню. З наведенням курсора на меню воно має стати прозорим, крім обраного пункту (рис. 6);

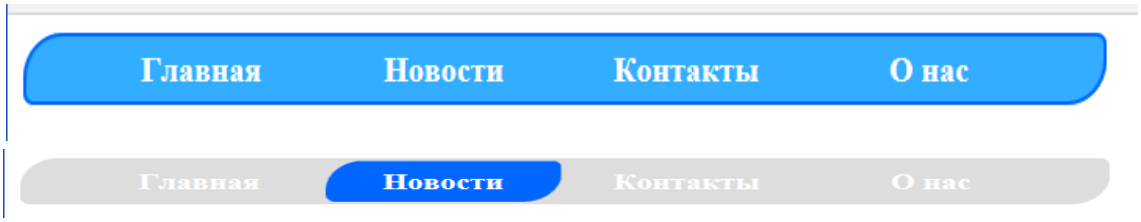

Рис. 6. **Вигляд меню**

1.2) зверстати сторінку з гумовою шириною та фіксованим в нижній частині вікна підвалом (рис. 7). Створити два варіанти (використовувати float і flex);

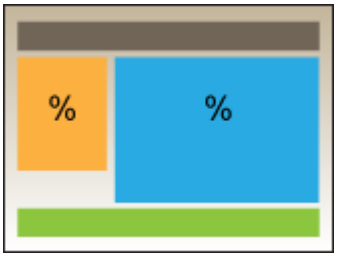

Рис. 7. **Вигляд сторінки**

1.3) зверстати сторінку, розмістивши заголовок (логотип, назву, форму для пошуку), горизонтальне меню, три колонки та підвал (рис. 8). Створити два варіанти (використовувати float і flex);

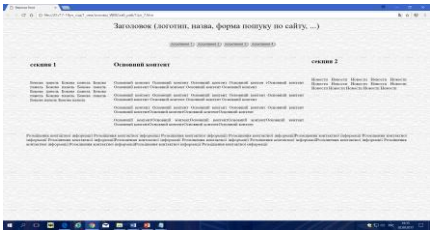

Рис. 8. **Вигляд сторінки**

2) адаптивне верстання сторінок:

2.1) зверстати сторінку, що забезпечує відображення тексту в дві колонки для великих екранів (> 768px) і в одну для маленьких. Використовувати медіазапити;

2.2) зверстати сторінку з кнопкою у верхній частині, яка з'являється під час перегляду на мобільному пристрої та відкриває спеціальне меню;

2.3) змінити одну зі сторінок, створених у процесі виконання пункту 1.3, зробивши її адаптивною (на мобільних пристроях контент виводиться в одну колонку, на планшетах – у дві, на великих екранах – у три);

3) верстання сторінок з використанням фреймворків:

3.1) створити, використовуючи Bootstrap, варіант сторінки з бічною панеллю навігації, що складається з кнопок поруч з основним контентом для великих екранів і з однією кнопкою та меню, яке випадає – для маленьких;

3.2) реалізувати завдання з пункту 2.3, використовуючи Bootstrap;

3.3) використовуючи Bootstrap, зверстати сторінку відповідно до наведеного макету (рис. 9);

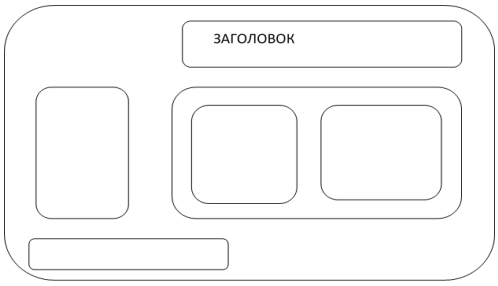

Рис. 9. **Макет сторінки**

4) верстання сторінок за шаблоном:

4.1) знайдіть макет HTML-сторінки в форматі psd (або візьміть у викладача);

4.2) переконайтеся, що макет нарізаний (за необхідності наріжте), збережіть потрібні зображення та зверстайте сторінку.

Контрольні запитання

1. Сформулюйте вимоги до верстальника (за компетентностями).

2. Яке програмне забезпечення може знадобитися верстальнику для роботи?

3. Що є основою для виконання верстки WEB-сторінок?

4. Розкрийте змістовність поняття "верстка WEB-сторінок".

5. Які вимоги пред'являються до верстання?

6. Дайте характеристику використовуваних підходів до виконання верстання.

7. Порівняйте між собою засоби створення багатоколонних макетів WEB-сторінок (float і display: flex).

8. У чому відмінність одиниць вимірювання em і rem?

9. Що таке чутливий дизайн, адаптивний дизайн?

10. Які засоби можна використовувати для створення чутливих сторінок, адаптивних сторінок?

Тема 6. Основи використання мови JavaScript

Лабораторна робота 7. Розроблення сценаріїв для WEB-сторінок

Мета – вивчення середовища виконання, прийомів створення сценаріїв для Web-сторінок і методів зв'язування сценаріїв з подіями різних елементів Web-сторінки.

Після виконання лабораторної роботи студент повинен:

знати:

правила вбудовування сценаріїв у текст сторінки;

типи даних, допустимих у сценарії, і їх особливості;

основні оператори й операції;

порядок виконання сценаріїв;

методи зв'язування подій з обробниками;

уміти:

виконувати перетворення типів;

використовувати основні оператори;

розробляти прості сценарії та створювати обробники подій.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з розміщення та налагодження сценаріїв на Web-сторінках.

Загальні відомості

Сценарії або скрипти (script – сценарій) є програмами на мовах JavaScript або Visual Basic Script (VBScript), що є підмножинами відповід-

них об'єктно-орієнтованих мов програмування. Ці програми включаються безпосередньо в HTML-документ у вигляді вихідних текстів. Виконуються вони браузером і дозволяють змінювати зовнішній вигляд вікна, а також створювати різні ефекти у вікні.

Для розміщення коду сценаріїв у тексті сторінки слугує спеціальний парний тег SCRIPT. Він може містити або текст сценарію, або посилання на файл із розширенням js, що містить сценарій. Сам тег може розміщатись у будь-якому місці коду сторінки. Відмінність полягає в доступності імен та можливості їх використання. Крім того, текст сценарію може бути присутнім як значення окремих атрибутів інших тегів, наприклад, HREF у посиланнях.

У загальному випадку тег виглядає таким чином:

```
<SCRIPT LANGUAGE="JavaScript">
// текст сценарію
</SCRIPT>
```

З розміщенням сценарію в окремому файлі, в тегу розміщується шлях та ім'я файлу, наприклад:

```
<SCRIPT LANGUAGE="JavaScript" SRC="root/my.js">
</SCRIPT>
```

Усередині файлу my.js міститься тільки текст сценарію, тегів немає.

Виконання сценаріїв. Оператори сценарію, які розміщуються в тегу SCRIPT, виконуються, коли браузер інтерпретує вміст тегу.

Оператори, розміщені у функціях, виконуються після виклику функцій (після звертання до функції). Виклик відбувається або в тегу (там, де записано звернення), або коли відбувається подія, з якою зв'язана функція (таку функцію називають оброблювачем події).

Події. У процесі взаємодії користувача зі сторінкою або під час зміни стану документу відбуваються різні події. Вони генеруються з пересуванням мишки, натисканням кнопок мишки або введенні із клавіатури тексту в документ, а також зі змінами стану документа (завантаження документа, зображень або об'єктів, поява помилки на сторінці, перехід фокуса від одного елемента до іншого тощо).

Деякі дії викликають послідовності подій. Наприклад, з клацанням мишкою послідовно відбуваються три події: onmousedown (натискання), onmouseup (відпускання), onclick (власне клацання). Є події специфічні для конкретних елементів (наприклад, onload для зображень), а є загальні (можуть бути в будь-якого елемента, наприклад, onclick). Крім того, необхідно мати на увазі, що різні браузери можуть підтримувати різне число подій для різних елементів.

Найбільш часто використовують події наведені в табл. 18.

Таблиця 18

Список подій

Об'єкт	Подія	Момент виникнення
window	onload	Закінчення завантаження сторінки
	onunload	Вихід зі сторінки для завантаження нової
image	onload	Закінчення завантаження зображення
link	onclick	Натиснення мишкою
	onmouseout	Відведення курсору з посилання
	onmouseover	Наведення курсору на посилання
Елементи форми	onblur	Втрата фокусу
	onfocus	Отримання фокусу
	onselect	Вибір тексту всередині поля text, textarea
	oncange	Зміна значення всередині поля
	onclick	Натиснення мишею на кнопці або перемикачі

Зв'язування обробників з подіями може відбуватися за замовчуванням (наприклад, для клацання на посиланні) за допомогою спеціальних атрибутів у тегах (наприклад, onClick="…"), а також динамічно в ході виконання сценарію.

Існуючі дії за замовчуванням можна перевизначити, змінивши на довільний сценарій. Наприклад, для посилання:

```
<a href="javascript:window.alert('Do you speak English?')">
```

Для зв'язування обробників з подіями в тегах використовують спеціальні атрибути, утворені від імен подій (onClick, onLoad і т.д.). Наприклад, якщо в тексті документа оголошені функції foo() і main(), то їх можна вказати в якості обробників подій:

```
<input type=button value="Натиснути тут" onClick="foo();">
<BODY onLoad ="main()">
```

У тег можна помістити не тільки виклик функції, але і сам код:

```
<INPUT TYPE=button VALUE="Тип програми?"
onClick="window.alert(window.navigator.appName);">.
```

Усі події є властивостями відповідних об'єктів, значенням цієї властивості є покажчик функції. Завдяки цьому можна виконувати динамічне зв'язування подій з обробниками. Імена властивостей вводяться в нижньому регістрі та починаються із префікса on.

```
<HTML>
  <HEAD>
    <TITLE>Приклад покажчика функції</TITLE>
  </HEAD>
  <BODY>
    <INPUT TYPE=BUTTON ID="myButton" VALUE="Click here">
    <SCRIPT LANGUAGE="JavaScript">
      // Приєднання покажчика функції до myButton.
      // При натисканні кнопки myButton відображається вікно повідомлення.
      document.all.myButton.onclick =  new Function("alert('Hello');");
    </SCRIPT>
  </BODY>
</HTML>
```

Можна призначити покажчик функції прямо у властивості:

```
<HTML>
  <HEAD>
    <TITLE>Призначення покажчика функції</TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      // Визначення функції clicked.
      function clicked() {
        alert("Clicked");
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <INPUT TYPE=BUTTON ID="myButton" VALUE="Click here">
    <SCRIPT LANGUAGE="JavaScript">
      // Призначення функції clicked оброблювачеві події onclick.
      document.all.myButton.onclick = clicked;
    </SCRIPT>
  </BODY>
</HTML>
```

Із призначенням покажчика функції використовується тільки ім'я функції. Круглі дужки та додаткові параметри не потрібні.

Можна призначити події за кількох обробників. Для цього використовуються функції API element.addEventListener(event,handler) і element.removeEventListener(event,handler). У цьому випадку ім'я події записується без префікса on. Наприклад,

```
document.getElementById("myB1").addEventListener("click", clicked);
document.getElementById("myB1").removeEventListener("click",clicked);
```

Робота з таймером. Виклик функції можна задати в скрипті, використовуючи звертання до таймера:

періодичний виклик функції через n мсек – id=setInterval("ім. ф.",n);

скасування викликів – clearInterval(id);

однократний виклик функції через n мсек – setTimeout("ім. ф.",n).

Найбільш часто в JavaScript використовуються класи Data, Image, Array, String, Math.

Об'єкти Data створюються конструктором var d=new Date(). З використанням конструктора без параметрів створюється об'єкт із поточними на момент виконання програми значеннями часу та дати. За наявності параметрів значення формується на їх основі:

var sDate=new Date("Month dd,yyyy hh:mm:ss")

var sDate=new Date("Month dd,yyyy")

У об'єкта є кілька методів, серед них: sDate.toLocaleString() – виведення у національному форматі.

Клас Array. Масиви в JavaScript також є об'єктами. Для них визначено кілька методів. Найчастіше використовуються такі: join, reverse, sort. Join об'єднує елементи масиву в рядок символів. У якості аргументу у цьому методі задається роздільник:

```
colors = new Array("red","white","blue")
string = acolors.join("+")
```

У результаті виконання присвоювання значення рядка символів string створюється рядок: "red + white + blue".

Інший метод, reverse, змінює порядок елементів масиву на зворотний, а метод sort відсортує їх у порядку зростання.

Для масивів визначена властивість length, вона містить число елементів масиву, його можна використати в програмі:

```
color = new Array("red","white","blue");
alert(color.length);
```

Об'єктам і масивам у JavaScript властива певна двоїстість. Тому об'єкти та масиви надають два способи доступу до їх змісту: пряме посилання на зміст як властивість за допомогою точкової (.) нотації або посилання на індекс у масиві за допомогою дужок ([index]). Індекс у масиві JavaScript може бути як цілочисельним, так і значенням типу String, що представляє ім'я властивості.

Клас String. Рядки також є об'єктами. Вони створюються шляхом присвоювання рядкового значення змінної або з використанням конструктора: **varSt=new String("рядок символів");**

Клас містить властивість length і багато методів. З них найчастіше використовують cartAt(i), який повертає символ, що стоїть на і-му місці, indexOf("...",i) – шукає входження підрядка; пошук починається з позиції і повертає номер позиції початку першого входження.

Під час роботи з рядками можна використовувати регулярні вирази (свого роду шаблони для пошуку певних комбінацій символів). Прикладом використання регулярного виразу може слугувати відомий шаблон для пошуку файлів *.htm. Регулярні вирази пишуться з використанням спеціального синтаксису. Будуються вони з літералів і символів: \ – ознака нелітералу; **.** – будь-який символ; ^ – початок рядка; $ – кінець рядка; {число} – попередній символ повинен зустрітися конкретне число разів; * – попередній символ може зустрічатися скільки завгодно разів або не зустрінеться взагалі; + – попередній символ може зустрічатися один і більше разів; ? – попередній символ повинен зустрітися один раз або не зустрінеться взагалі тощо. Круглі дужки означають, що комбінація повинна запам'ятовуватися. Регулярні вирази полягають у /.../. Приклади найпростіших регулярних виразів:

/www/ або /w{3}/ – три w поспіль;

/^www/ або /^w{3}/ – три w на початку рядка;

/.$/ – будь-який символ наприкінці рядка;

/\/$/ – слеш наприкінці рядка;

/\\$/ – зворотний слеш наприкінці рядка.

Об'єкти класу String мають кілька методів, що використовують регулярні вирази як аргументи:

replace(р. в.,"рядок") – повертає рядок із заміною заданих послідовностей рядком символів;

search(р. в.) – пошук заданої послідовності символів, повертає номер позиції першого входження.

Наприклад, у результаті виконання наступного оператора зворотний слеш наприкінці рядка (якщо він є) буде замінений на прямий:

```
St=St.replace(/\\$/,"/");
```

Клас Math. Об'єкти цього класу не вимагають створення. Його властивостями є математичні константи, а методами – математичні функції. Незважаючи на те, що на JavaScript рідко розробляють програми обчислювального характеру, у деяких випадках вони можуть бути корисними.

Звернення – Math.ім'я_функції(аргументи). Наприклад, після виконання оператора **alert(Math.random())** у вікні діалогу з'явиться псевдовипадкове значення, тобто з кожним повторним виконанням воно буде іншим.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) обрати варіант індивідуального завдання, підготувати алгоритми та тексти сценаріїв для всіх пунктів завдання.

Під час виконання лабораторної роботи необхідно:

1) виконати розміщення та виконання сценаріїв:

1.1) розробити сценарій за індивідуальним завданням (узгодити з викладачем);

1.2) випробувати різні варіанти розміщення сценарію (в теги та в файли);

1.3) випробувати різні варіанти виконання сценаріїв (виконання операторів у ході інтерпретації і за викликом функцій);

1.4) вивчити засоби налагодження в використовуваному середовищі розробки (точки зупину, покроковий режим, перегляд значень, виведення налагоджувальних повідомлень та ін.). Для роботи з простим редактором треба вивчити інструменти розробника в браузері;

2) дослідити різні способи зв'язування обробників з подіями, сформулювати рекомендації щодо їх застосування:

2.1) перевизначити стандартний обробник (на посиланні);

2.2) розмістити у тегу атрибути подій (onClick, onLoad і т. д.);

2.3) динамічно зв'язати обробник з подією;

2.4) зв'язати кілька обробників з однією подією;

3. Використання вбудованих об'єктів:

3.1) розробити та випробувати сценарій, який використовує об'єкт Date;

3.2) розробити та випробувати сценарій, який змінює властивості об'єкта Image;

3.3) розробити та випробувати сценарій з використанням об'єкта String і регулярних виразів.

Варіанти завдань до лабораторної роботи

Завдання для п.1.

1. Введіть масив з N цілих чисел. Сформуйте новий масив, кожен елемент якого дорівнює елементу введеного масиву, збільшеному на значення, що вводиться з клавіатури.

2. Заповніть випадковими значеннями масив з N цілих чисел. Обчисліть суму елементів масиву з парними номерами.

3. Заповніть масив випадковими значеннями. Визначте число елементів масиву, що потрапляють в інтервал, введений з клавіатури.

4. Заповніть масив випадковими значеннями. Введіть значення. Обчисліть суму елементів, більших за введене значення, та число таких елементів.

5. Введіть з клавіатури значення елементів масиву з N дійсних чисел. Сформуйте новий масив, кожен елемент якого дорівнює квадрату елемента введеного масиву.

6. Розробіть програму для обчислення дисперсії щодо масиву випадкових значень:

$$D = \frac{\sum_{i=1}^{n} (x_i - M)^2}{n-1}, \text{ де } M = \sum_{i=1}^{n} x_i / n, \quad x_i \text{ – випадкове значення.}$$

7. Введіть з клавіатури масив, знайдіть і виведіть мінімальний елемент.

8. Введіть з клавіатури масив, знайдіть і виведіть максимальний елемент.

9. Створіть масив з прізвищами та роком народження. Додайте в кінець прізвище і рік, введені з клавіатури.

10. Введіть з клавіатури масив, упорядкуйте його за зростанням, замініть останній елемент рядком "FF".

Завдання для п. 2.

1. Розробіть скрипт для зміни зображення, зв'яжіть з відведенням курсора.

2. Розробіть скрипт для зміни розмірів зображення, зв'яжіть з натисканням клавіші на клавіатурі.

3. Розробіть скрипт для зміни положення зображення, зв'яжіть з наведенням курсора (елемент, що тікає від курсора).

4. Розробіть скрипт для введення рядка та пошуку в ній фіксованої послідовності символів, зв'яжіть з клацанням мишкою над елементом.

5. Розробіть скрипт для пошуку в рядку номера телефону, зв'яжіть з кнопкою в формі.

6. Розробіть скрипт для перевірки, чи є рядок адресою електронної пошти; призначте як обробник зв'язан динамічно з будь-яким елементом за власним вибором.

7. Розробіть скрипт для перевірки, чи містить рядок слово Харків, зв'яжіть динамічно з будь-якою подією за власним вибором.

8. Розробіть скрипт для виведення поточного значення часу великими цифрами, зв'яжіть динамічно та прив'яжіть до якоїсь умови (подія таймера з перевіркою часу).

9. Розробіть скрипт для виведення поточного значення часу, вбудуйте в посилання з написом: "Показати час".

10. Розробіть скрипт для заливки кольором клітин таблиці шляхом клацання мишкою.

Завдання для п. 3 (розробіть самостійно).

3.1. Може містити одну з таких дій: обчислення часового інтервалу між подіями, порівняння дат, порівняння дати з поточною (оцінювання актуальності), обчислення часу до настання події, пошук за ознакою потрапляння в часовий інтервал і т.п.

3.2. Може містити аналіз доступних властивостей, зміну одної з доступних властивостей тощо.

3.3. Повинно містити власну задачу на використання регулярних виразів (пошук даних у рядку, зміна рядків і т. п.).

Контрольні запитання

1. Чи можуть у тегу SCRIPT міститися виконувані оператори, виклики функцій? Якщо так, то вкажіть момент і число повторення їх виконання.

2. Які мови використовуються для написання сценаріїв?

3. Які функції зазвичай реалізуються на клієнті?

4. Що таке "use strict"?

5. У чому полягає основна особливість використовуваних типів даних?

6. Назвіть імена функцій для перетворення даних.

7. Сформулюйте особливості використання масивів у JS.

8. Запишіть оператор, використовуючи який можна отримати значення поточної дати та часу.

9. Чи може бути параметр функції масивом?

10. Чи може функція повертати в якості значення масив?

11. Що таке асоціативний масив? У чому його відмінність від традиційного?

12. Запишіть приклад вираження для перебору значень асоціативного масиву.

13. Назвіть вбудовані об'єкти мови JS.

14. Які оператори мови JS призначені для роботи з об'єктами?

15. Запишіть різні варіанти доступу до властивостей об'єкта Image?

16. Що таке регулярні вирази? Як вони використовуються в JS?

17. Запишіть приклад з фрагментом коду, який ілюструє використання регулярних виразів.

18. Чи припустимий вираз dd=Date()-Date();? Якщо так, то яке значення отримає змінна dd?

19. Назвіть способи зв'язування обробників з подіями.

20. Що означає зв'язати обробник з подією динамічно?

Лабораторна робота 8. Оброблення подій

Мета – вивчення середовища виконання, прийомів створення сценаріїв для Web-сторінок і можливостей клієнтського JavaScript зі зміни елементів Web-сторінки.

Після виконання лабораторної роботи студент повинен:

знати:

властивості та методи використання об'єкта event;

способи доступу до об'єктів документа;

способи доступу до властивостей таблиць стилів;

уміти:

використовувати об'єкт event у обробниках;

створювати ефекти, використовуючи зміну властивостей стилів.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з розроблення обробників подій для WEB-сторінок.

Загальні відомості

Об'єкт Event. У процесі оброблення події браузер передає в функцію обробника в якості параметра об'єкт Event, який містить усі дані про подію. Найчастіше використовуються такі властивості об'єкта:

type – ім'я події;

target – елемент, який згенерував подію;

currentTarget – елемент, який викликав обробник події;

timestamp – час, коли відбулася подія;

button (для миші) – повертає число, яке вказує, яка кнопка миші була натиснута;

clientX, clientY (для мишки) – повертають інформацію про розташування курсору щодо елемента;

screenX, screenY (для мишки) – повертають інформацію про розташування курсору щодо лівого верхнього кута екрану;

charCode (для клавіатури) – повертає код символу Unicode натиснутої клавіші (для події keypress).

Доступ до об'єктів сторінки. Доступ до об'єктів для виконання дій з елементами може здійснюватися шляхом переміщення по дереву DOM із використанням структури дерева (масиви посилань, зображень, форм та ін.). Наприклад, document.images[0].width – ширина першого зображення на сторінці; document.links[1].href – URL другого посилання на сторінці.

Доступ до необхідного єлемента може бути отриманий з використанням посилань в ієрархії елементів:

el.children – посилання на масив дочірніх елементів;

el.firstElementChild, el.lastElementChild – посилання на перший і останній дочірні елементи;

el.previousElementSibling, el.nextElementSibling – посилання на сусідів;

el.parentElement – посилання на батьківський елемент.

Практично всі браузери підтримують колекцію (асоціативний масив) all. Колекція all є властивістю об'єкта document і містить посилання на вузли елементів у порядку їх розташування у вхідному потоці браузера.

Доступ здійснюється за номером, id, name або ім'ям тега:

document.all (8);

document.all ("img5");

document.all.tags ("H1").

Найбільш зручним методом вважається доступ до елементів з використанням спеціальних методів об'єкта document:

getElementById (id) – пошук елемента із id;

getElementsByName (name) – пошук за значенням атрибута name;

getElementsByTagName (tag) – пошук за ім'ям тега;

getElementsByClassName (class) – пошук за значенням атрибута class;

querySelector (css-sel) – пошук у селекторі (поверне перший елемент);

querySelectorAll (css-sel) – пошук у селекторі (поверне всі елементи).

Доступ до властивостей таблиць стилів. Для форматування HTML-документів використовують таблиці стилів. Із кожним елементом зв'язаний набір параметрів (стиль), які визначають його зовнішній вигляд. Відповідно до цієї концепції об'єкти елементів мають як властивість об'єкт style, що містить усі правила форматування (кожній припустимій властивості таблиці стилів відповідає властивість об'єкта style).

З урахуванням особливостей синтаксису JavaScript імена властивостей записуються без роздільника "-", а початок кожної складової імені, крім першої, починається із великої літери. Наприклад, у CSS ім'я властивості пишеться border-bottom-color, а в скрипті ця ж властивість має ім'я borderBottomColor. Значення всіх властивостей є даними рядкового типу.

Для роботи з таблицями використовується об'єкт styleSheets, який містить колекцію таблиць стилів. Його властивостями є href (адреса файлу таблиці), disabled (false – дозвіл на використання таблиці, true – заборона) та type (визначає синтаксис таблиць text/css або text/javascript). Однак href доступна тільки для читання, тому для зміни файлу потрібно використовувати елемент DOM, який відповідає тегу link: document.getElementsByTagName("link")[0].href="my.css".

Доступ до стилів елемента здійснюється через властивість style. Це об'єкт, і його властивості можна динамічно змінювати. Буде змінюватись і зовнішній вигляд конкретних елементів. Однак спочатку ці властивості будуть визначені тільки в тому випадку, якщо правила для властивостей задані в тегах атрибутом STYLE, тобто без урахування каскадування таблиць (зовнішні, імпортовані, вбудовані). Після присвоєння значення властивість буде визначена так, ніби вона задана в тегу. Треба також пам'ятати, що значення всіх властивостей мають рядковий тип.

Наприклад, для тегу <img name="im1" src="690.jpg" border="0"> вираз document.im1.style.width=parseInt(document.im1.style.width)/2+"px" буде помилковим, тому що style.width не визначено. А для тегу <img name="im1" STYLE="width:100px" src="690.jpg" border="0"> він буде правильним, і ширина зменшиться вдвічі.

Для отримання значення властивостей елементів з урахуванням декількох таблиць, які визначені поза тегами, потрібно використовувати

спеціальний метод getComputedStyle(element). Наприклад: element =document. getElementById("im1");
element.style.width= parseInt(getComputedStyle(element).width)/2+"px";

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) обрати варіант індивідуального завдання, підготувати алгоритми та тексти сценаріїв для всіх пунктів завдання.

Під час виконання лабораторної роботи необхідно:

1) використати властивості і методи об'єкта event:

1.1) вивести список властивостей і ознайомитись із властивостями, вибрати властивість для використання;

1.2) розробити і налагодити скрипт з використанням обраної властивості;

2) використати властивості і методи об'єктів документа:

2.1) дослідити доступ до об'єктів (за ієрархією в дереві, за допомогою об'єкта all, функцій getElementsByName, getElementsByTagName, getElementById) і різні механізми доступу до властивостей (колекції, масиви, об'єкти);

2.2) створити динамічне код документа, що змінює зовнішній вигляд (фон, кольор і т. п.) залежно від типу браузера, дати, часу доби або вибору користувача;

2.3) розробити скрипт підтримки інтерфейсу користувача за індивідуальним завданням;

3) подати доступ до властивостей таблиць стилів:

3.1) розробити скрипт для створення ефекту (зміна значення властивості за сигналом таймеру);

3.2) створити сторінку з елементами управління, що забезпечують зміну положення, розмірів і оформлення одного з елементів (абзац, блок, малюнок і т. д.) або групи елементів, зв'язати з подією динамічно.

Варіанти завдань до лабораторної роботи

Завдання для п.1.

1. Визначте елемент, що викликав подію (створіть сторінку з декількома елементами, виведіть у консоль ім'я події, ім'я тега та значення атрибута name).

2. Виділіть кольором елемент, який викликав переривання після клацання правою кнопкою мишки.

3. Виділіть рамкою елемент, який викликав переривання після клацання лівою кнопкою мишки.

4. Розмістіть центр елемента щодо точки клацання мишкою.

5. Перемістіть елемент в точку клацання мишкою.

6. Розпізнайте натиснуту клавішу мишки, результат виведіть у консоль у вигляді протоколу роботи.

7. Розпізнайте натискання на клавіатурі клавіші Esc.

8. Реалізуйте переміщення об'єкта клавішами управління курсором.

9. Розробіть скрипт для виведення символів натиснутих клавіш у рядок стану.

10. Розробіть скрипт для перетягування елементів мишкою.

Завдання для п.2.

1. Створіть форму з елементами, що описують зовнішній вигляд сторінки, та скрипт, який встановлює відповідні параметри.

2. Створіть форму з елементами, що описують шрифтові параметри сторінки, та скрипт, який встановлює ці значення.

3. Розробіть калькулятор для перерахунку цін з однієї валюти в іншу.

4. Розробіть форму та скрипт для перевірки правильності її заповнення (регулярні вираження).

5. Розробіть кілька форм для тестування та скрипт для послідовного їх подання після відповіді на всі питання.

6. Розробіть скрипт для розміщення на сторінці елемента, раніше відсутнього в розмітці.

7. Створіть інструменти, аналогічні кнопкам вікна браузера (онови-ти, додому, переміщення раніше завантаженими сторінками тощо).

8. Створіть на сторінці розділ і елементи для управління редагуванням (використовуйте атрибут contentEditable та виділення змін).

9. Створіть на сторінці редагований розділ (скопіюйте в текстове поле, відредагуйте, збережіть у вихідному елементі).

10. Створіть панель для управління оформленням зображення на сторінці.

Завдання для п. 3 (розробіть самостійно).

1. Приклади ефектів: мерехтіння елемента, плавна зміна кольору, трансформації, розчинення і прояв, геометричні спотворення, змазування і т.п.

2. Повинна змінюватися група параметрів, створюючи ефект адаптації до запитів користувача. Наприклад, форма пошуку повинна знаходиться в тому місці, на яке її перемістив користувач і не прокручуватися, допускаючи повторне переміщення.

Контрольні запитання

1. Назвіть відомі вам властивості об'єкта Window.
2. Як звернутися до властивостей вікна фрейма?
3. Запишіть фрагмент коду, що забезпечує завантаження у вікно нової сторінки.
4. Для чого використовують властивість all? Якими браузерами вона підтримується?
5. Який з варіантів доступу до форми на ім'я є правильним document.id_form або document.forms("id_form")?
6. Значенням якого типу є властивість document.im1.style.width? document.im1.style.pixelWidth?
7. Для чого використовується об'єкт event? Його властивості?
8. Що виконується для звернення до методу getAttribute("ім'я") об'єкта style? Що визначає параметр "ім'я"?
9. Сформулюйте правило запису імен властивостей, що задаються в таблицях стилів?
10. Чи можна зі скрипта поміняти таблицю стилів для сторінки? Якщо так, то яким чином?

Змістовий модуль 4. Засоби створення WEB-сайтів

Тема 7. Створення динамічних елементів та ефектів

Лабораторна робота 9. Використання об'єктів у скриптах

Мета – вивчення засобів створення сценаріїв для Web-сторінок і їх можливостей щодо підвищення ефективності розробки Web-сторінок.

Після виконання лабораторної роботи студент повинен:

знати:

популярні бібліотеки та фреймворки;

основні конструкції з використанням звернень до jQuery;

методи вибору елементів на сторінці;

основні методи бібліотеки jQuery і їх використання;

уміти:

підключати бібліотеку;

використовувати засоби вибору елементів для обробки;

застосовувати ланцюжки методів для обробки;

розширювати бібліотеку.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з розроблення сценаріїв для WEB-сторінок за допомогою jQuery.

Загальні відомості

Підключення jQuery здійснюється шляхом включення в текст сторінки тега script з посиланням (href = "url") на файл з бібліотекою, який може знаходитися як у папці сайта, так і на довільному ресурсі мережі.

Наприклад:

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

або

```
<script type="text/javascript" src="http://yandex.st/jquery/1.6.2/jquery.js">
</script>
```

Після підключення для використання доступні базові можливості jQuery, серед яких: робота з селекторами; робота з атрибутами; обхід дерева DOM; маніпуляції елементами; робота з CSS-властивостями елементів; робота з подіями; візуальні ефекти; взаємодія з ajax; утиліти.

Додаткові можливості доступні після підключення відповідних плагінів (наприклад, gallery.js для створення галерей зображень і т. п.).

Для запуску скрипта після закінчення побудови дерева DOM браузером використовується спеціальний метод ready. Може використовуватися кілька варіантів.

Так:

```
$ (document) .ready (function () { // ініціалізація
                                   });
```

Або так:

```
$ (document) .ready (f1);
function f1 ()
{// ініціалізація
}
```

Або так:

```
 $ (F1);
function f1 ()
{// ініціалізація
}
```

Для вибору елементів, які будуть оброблятися, використовується звернення до бібліотеки $() або jQuery() з передачею в якості параметра одного із селекторів. Як селектор використовуються будь-які селектори CSS. Серед них можна зазначить такі:

"*" – усі елементи;

".className" – елементи з класом className;

"#idName" – елемент (один!) з ідентифікатором idName;

"tagName" – елементи із заданим ім'ям тега;

"first, second, ..." – елементи, які відповідають умовам будь-якого з селекторів first, second, ...;

"outer inner" – елементи з inner, які є нащадками (тобто розташовані всередині) елементів з outer;

"parent>child" – елементи з child, які є прямими нащадками елементів з parent;

"prev+next" – елементи з next, які прямують безпосередньо за елементами з prev;

"prev ~ next" – елементи з next, які слідують за елементами з prev;

"[name]" – елементи, що містять атрибут name;

"[name=value]" – елементи, в яких значення атрибута name збігається з value;

"[name != Value]" – елементи, в яких значення атрибута name не збігається з value;

"[name ^= value]" – елементи, в яких значення атрибута name починається з value;

"[name $= value]" – елементи, в яких значення атрибута name закінчується на value;

"[name *= value]" – елементи, в яких значення атрибута name містить підрядок value;

"[name ~= value]" – елементи, в яких значення атрибута name містить слово value;

"[name |= value]" – елементи, в яких значення атрибута name мають префікс value (дорівнює value або має вигляд: "value– *");

"[first] [second] [..." – елементи, які відповідають усім заданим умовам на атрибути одночасно.

Спільно з селекторами можна використовувати фільтри, за допомогою яких уточнюють результат застосування селекторів (відібрати або прибрати з вибірки частину елементів за будь-якою ознакою). За призначенням і синтаксисом вони дуже схожі на псевдокласи в CSS.

Загальні фільтри:

":focus" – елемент, що знаходиться у фокусі;

":first" – перший знайдений елемент;

":last" – останній знайдений елемент;

":eq ()" – елемент іде під заданим номером серед обраних;

":not (selector)" – усі знайдені елементи, крім зазначених у selector;

":even" – елементи з парними номерами позицій, у наборі обраних елементів;

":odd" – елементи з непарними номерами позицій, в наборі обраних елементів;

":gt ()" – елементи з індексом, який перевищує n;

":lt ()" – елементи з індексом меншим ніж n;

":header" – елементи, які 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;

":lang (language)" – елементи, в яких вказані мови вмісту;

": Root" – елемент, який є кореневим у документі.

Фільтри зі вмісту:

":contains (text)" – елементи, що містять заданий текст;

":empty" елементи без вмісту (без тексту та інших елементів);

":has (selector)" – елементи, які містять хоча б один елемент з selector;

":parent" – непусті елементи.

Фільтри дочірніх елементів (дані селектори фільтрують елементи за їх розташуванням у батьківських елементах):

":first–child" – елементи, розташовані першими в своїх батьківських елементах;

":last–child" – елементи, розташовані останніми в своїх батьківських елементах;

":nth–child()" ":nth–child–last()" – елементи, розташовані певним чином у батьківських елементах (парні, непарні, які йдуть під заданим номером);

":only–child" – елементи, які є єдиними нащадками в своїх батьківських елементах;

":only–of–type" – елементи, які є єдиними нащадками в своїх батьківських елементах, що задовільнюють селектору;

":first–of–type" – ті з обраних елементів, які першими зустрічаються в своїх батьківських елементах;

":last–of–type" – ті з обраних елементів, які останніми зустрічаються в своїх батьківських елементах.

Фільтри елементів форм:

":button" – елементи з тегом button або типом button;

":radio" – елементи, які є перемикачами;

":checkbox" – елементи, які є прапорцями;

":text" – елементи, які є текстовими полями;

":password" – елементи, які є полями введення пароля;

":file" – елементи, які є полями завантаження файлів;

":submit" – елементи, які є кнопками відсилання форми;

":reset" – елементи, які є кнопками очищення форми;

":image" – елементи, які є зображеннями для відсилання форми (input типу image);

":input" – елементи, які є елементами форми (з тегами input, textarea або button);

":selected" – вибрані елементи (зі статусом selected), це можуть бути елементи типу <option>;

":focus" – елементи форми, що знаходяться в фокусі;

":checked" – вибрані елементи (зі статусом checked), це можуть бути елементи типу <checkbox> або <radio>;

":enabled" – активні елементи форми (зі статусом enabled);

":disabled" – неактивні елементи форми (зі статусом disabled).

Використання методів jQuery. Після вибору елемента або групи елементів за допомогою селекторів і фільтрів до них застосовуються різні методи. Методи зазвичай застосовуються до всіх членів колекцій, що повертаються. Наприклад, $("#bigIt").attr("class","noContent") – у елемента буде змінено значення атрибута class.

Важливою особливістю більшості методів jQuery є можливість по-в'язувати їх в ланцюжки:

$("#BigIt").empty().attr("class","noContent") – у елемента буде видалено вміст, а потім змінено значення атрибута.

Методів досить багато, умовно їх можна розподілити на кілька груп.

Зміна вмісту:

html() – повертає/змінює html–вміст елементів;

text() – повертає/змінює текст, що знаходиться в елементах;

append(), appendTo() – додає заданий вміст у кінець елементів на сторінці;

prepend(), prependTo() – додає заданий вміст у початок елементів;

after(), insertAfter() – додає заданий вміст після елементів;

before(), insertBefore() – додає заданий вміст перед елементами;

wrap(), wrapAll(), wrapInner() – оточує елементи на сторінці заданими html–елементами.

Маніпуляції з елементами:

detach(), remove() – видаляє елементи на сторінці;

empty() – видаляє вміст елементів на сторінці;

unwrap() – видаляє батьківські елементи, проте їх вміст залишається на місці;

replaceWith(), replaceAll() – замінює одні елементи сторінки на інші (нові або вже існуючі);

clone() – повертає копію обраних елементів сторінки.

Робота з подіями:

on(), bind(), live(), delegate() – установка обробника;

one() – обробник спрацює тільки по одному разу на кожному з елементів;

off(), unbind(), die(), undelegate() – видалення обробника;

trigger(), triggerHandler() – запускає обробник події;

click(), dblclick(), mousedown(), mouseup(), mouseout(), mouseover(), hover() – установка обробників подій мишки;

keydown(), keyup(), keypress() – події клавіатури;

event – об'єкт, що містить дані про поточну подію, передається всім обробникам подій.

Події форми:

focus() – установлює обробник отримання фокусу або запускає цю подію;

blur() – установлює обробник втрати фокусу або запускає цю подію;

select() – установлює обробник виділення тексту або запускає цю подію;

submit() – установлює обробник відправки форми або запускає цю подію;

change() – установлює обробник зміни елемента форми або запускає цю подію.

Робота з атрибутами:

attr() – повертає/змінює значення атрибута у елементів;

removeAttr() – видаляє атрибут у елементів;

prop() – повертає/змінює значення заданої властивості;

removeProp() – видаляє задану властивість у елементів;

addClass() – додає клас елементів на сторінці;

removeClass() – видаляє клас у елементів на сторінці;

hasClass(clName) – перевіряє наявність заданого класу хоча б у одного з обраних елементів;

val() – повертає/змінює значення атрибута value у елементів на сторінці;

Робота зі стилями:

css() – повертає/змінює CSS властивості 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 (прокрутку) елемента.

Анімація і ефекти:

animate({"ім'я_св":"тіп_дії"},"продовж") – плавна зміна властивостей за заданий час;

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.

Створення елементів, відсутніх у первісній розмітці. Приклад клієнтського додатку, який додає редагований розділ для створення замітoк:

```
<!DOCTYPE html>
<html>
<head>
<script src = "jquery.js" type = "text / javascript"> </ script>
<style>
div {background-color: #cccccc; border-radius: 5px; width: 250px}
</style>
<script>
$(F1);
function f1 ()
{
$ ("#but").click (function() {var newElems = $('div.stud').clone();
$( 'body'). append (newElems);});
}
</script>
</head>
<body>
<button id = "but"> додати </button>
<div class="stud" contentEditable=true>
<img align=left src="im1.png" width=50/><label>фам</label><input size=5/> <br>
<label> ім'я </label> <input size=15 />
</div>
</body>
```

За клацанням на кнопці буде створено ще один розділ, у якому користувач зможе ввести свій текст.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) обрати варіант індивідуального завдання, підготувати алгоритми та тексти сценаріїв для всіх пунктів завдання.

Під час виконання лабораторної роботи необхідно:

1) підключення бібліотеки JQery:

1.1) створити шаблон сторінки з підключеною бібліотекою, розмістивши файл на своєму сайті та пославшись на сайт виробника. Порівняти варіанти;

1.2) випробувати конструкцію, що забезпечує початок використання скриптів бібліотеки тільки після закінчення завантаження сторінки;

1.3) використовувати різні селектори та фільтри (не менше трьох);

2) обробка подій:

2.1) розробити скрипт підтримки інтерфейсу користувача (відповідно до вибраного варіанта);

2.2) використати для зв'язування обробників декілька відповідних методів;

3) розробити інтерфейс локального додатку для накопичення документів (відповідно до обраного варіанту):

3.1) розробити дизайн;

3.2) запропонувати функціонал (склад функцій з управління додатком);

3.3) розробити інтерфейс з використанням різних елементів управління;

3.4) розробити скрипт для підтримки інтерфейсу.

Варіанти завдань до лабораторної роботи

Завдання для п. 1.

1. Створіть форму з елементами, що описують зовнішній вигляд сторінки, та скрипт, який встановлює відповідні параметри.

2. Створіть форму з елементами, що описують шрифтові параметри сторінки, та скрипт, який встановлює ці значення.

3. Розробіть калькулятор для перерахунку цін з однієї валюти в іншу.

4. Розробіть форму та скрипт для перевірки правильності її заповнення (регулярне вираження).

5. Розробіть кілька форм для тестування та скрипт для послідовного їх подання після відповіді на всі запитання.

6. Розробіть скрипт для розміщення на сторінці елемента, раніше відсутнього в розмітці.

7. Створіть інструменти, аналогічні кнопкам вікна браузера (онови-ти, додому, переміщення раніше завантаженими сторінками тощо).

8. Створіть на сторінці розділ і елементи для управління редагуванням (використовуйте атрибут contentEditable та виділення змін).

9. Створіть на сторінці редагований розділ (скопіюйте в текстове поле, відредагуйте, збережіть у вихідному елементі).

10. Створіть панель для управління оформленням зображення на сторінці.

Завдання для п. 2.

1. Розробіть скрипт, який після завантаження забезпечить для всіх посилань за наведенням курсору відображення у спливаючому вікні значення href.

2. Розробіть сторінку, яка містить блоки з анотаціями статей. За клацанням блок повинен плавно заповнити екран повним текстом. Після повторного клацання – плавне зменшення до попередніх розмірів.

3. Створіть сторінку з набором перемикачів (чекбоксів) і текстовим полем, в яке можна ввести число. Якщо введений текст не є числом, створити попередження. Розробіть скрипт, який зі введенням числа в поле та втратою фокусу знаходить і зазначає чекбокс із відповідним номером.

4. Створіть сторінку, яка містить елемент select і набір радіокнопок, кількість кнопок дорівнює кількості пунктів у випадаючому списку. Розробіть скрипт, який з вибором пункту в списку автоматично позначає відповідну радіокнопку.

5. Створіть форму з елементами, що описують зовнішній вигляд сторінки (не менше п'яти різних параметрів різними елементами), та скрипт, який реалізує зміну вигляду сторінки.

6. Створіть сторінку, яка містить блоки з текстом, що вібрують (тремтять) з наведенням курсору.

7. Створіть сторінку, заповнену малюнками (мозаїка з малюнків з хаотичним накладенням). Розробіть скрипт, що видаляє малюнки зі клацанням мишкою. Видалення проводиться шляхом зменшення та стягування в точку.

8. Створіть сторінку, яка містить кілька абзаців з елементами strong і em. Задайте обробники подій наведення мишкою на ці елементи. З наведенням, колір елементів повинен змінюватися від звичайного, а розмір – плавно збільшуватися.

9. Створіть сторінку, яка містить "купку" блоків із текстом. Після клацання верхній блок повинен розчинятися, показуючи нижній.

10. Створіть сторінку, яка містить дві "купки" блоків з текстом і зображеннями (розгорнута книжка). Розробіть скрипт, що імітує перегортання сторінок під час клацання мишкою.

Завдання для п. 3.

Необхідно створити:

1) щоденник;
2) календар планування робіт;
3) анкети відвідувачів;
4) збірку кулінарних рецептів;
5) список адрес і телефонів друзів;
6) список товарів;
7) календар погоди;
8) перелік замовлень на доставку товарів;
9) список праць;
10) каталог книг особистої бібліотеки.

Контрольні запитання

1. Яку користь для розроблення надає використання бібліотек?
2. У чому відмінність бібліотеки від фреймворка?
3. Назвіть кілька відомих вам бібліотек.
4. Який розмір має бібліотека jQuery?
5. Назвіть основні групи функцій ядра jQuery. Назвіть кілька відомих вам розширень цієї бібліотеки.
6. Як здійснюється звернення до функцій jQuery?
7. Поясніть як можна вибрати потрібні елементи сторінки. Що таке селектори та фільтри?
8. Сформулюйте основні особливості застосування методів jQuery.
9. Назвіть і запишіть приклади з відомими вам методами.
10. Як включити в бібліотеку свій метод?

Лабораторна робота 10. Створення динамічних сторінок

Мета – вивчення засобів створення сценаріїв для Web-сторінок і технологій, що підвищують ефективність розроблення Web-сторінок.

Після виконання лабораторної роботи студент повинен:

знати:

засоби, методи та технології для створення WEB-сторінок і WEB-сайтів;

нові інтерфейси HTML5 (Canvas, localStorage, Web workers), що розширюють можливості клієнтських скриптів;

уміти:

створювати динамічні зображення на WEB-сторінках;

використовувати локальні сховища даних у браузері;

організовувати обчислення в фоновому режимі.

У результаті виконання лабораторної роботи у студента формуються **компетентності** зі створення сайтів за допомогою сучасних технологічних засобів.

Загальні відомості

Для відображення динамічно створюваних зображень слугує елемент Canvas, описуваний однойменним тегом. Тег може містити атрибути height і width, а також ряд універсальних (id та ін.). Усередині елемента зображення створюється з відрізків прямих, ламаних і дуг кіл. Зображення створюється шляхом виклику спеціальних методів.

Методи API Canvas:

getContext(context) – створює контекст для полотна, поки підтримується режим 2D;

fillRect(x,y,width,height) – залитий кольором прямокутник зі сторонами width і height в точці (x,y);

strokeRect(x,y,width,height) – прямокутний контур зі сторонами width і height;

clearRect(x,y,width,height) – очищає прямокутну область;

beginPath() – оголошення початку нового шляху;

closePath() – закриває шлях, створює пряму лінію між останньою позицією пера та початковою точкою шляху;

stroke() – візуалізація контуру шляху;

fill() – візуалізації шляху як фігури, залитої суцільним кольором;

moveTo(x,y) – переносить віртуальне перо в нове місце розташування;

lineTo(x,y) – додає до шляху пряму лінію, починаючи від поточної позиції пера і до точки, яка визначається атрибутами x і y;

rect(x,y,width,height) – додає до шляху прямокутник у точці (x,y) зі сторонами width і height;

arc(x,y,radius,startAngle,endAngle,direction) – додає до шляху дугу;

strokeText(text,x,y,max) – малює контурний текст, атрибут max необов'язковий, він задає максимальний розмір тексту;

fillText(text,x,y,max) – малює текст, який є залитою кольором фігурою, атрибут max необов'язковий, він задає максимальний розмір тексту;

translate(x,y) – переносить початок координат полотна в точку (x, y);

rotate(angle) – обертає полотно навколо початку координат;

scale(x,y) – змінює масштаб полотна, значення за замовчуванням (1,0, 1,0), можна також задавати негативні значення;

save() – зберігає поточний стан полотна;

restore() – відновлює стан полотна, який був збережений;

drawImage() – виводить на полотно зображення та підтримує три варіанти синтаксису: drawImage (image,x,y) – дозволяє вивести зображення на полотно в точці (x, y); drawImage(image,x,y,width,height) – дозволяє вивести зображення на полотно в точці (x,y), з новими розмірами; drawImage(image,x1,y1,width1,height1,x2,y2,width2,height2) – дозволяє вирізати з вихідного зображення частину, яка визначається параметрами x1, y1, widthl і heightl, і вивести її на полотно в точці (x2,y2) з новими розмірами width2 і height2;

getImageData(x,y,width,height) – зчитує вміст фрагмента полотна та зберігає його у вигляді об'єкта;

putImageData(imagedata,x,y) – візуалізує дані, збережені в об'єкті imagedata, як зображення на полотні.

Фрагмент коду, що створює зображення, може виглядати так:

```
...
<script>
var im = new Image (); // буферизація зображення
im.src = "sprite.png";   // для виведення на полотно
function draw1 ()
{var canvas = document.getElementById('canva');
 if (canvas.getContext) {
                        var ctx1=canvas.getContext('2d');
                        var ctx2=canvas.getContext('2d');
                        ctx1.drawImage(im,0,0,200,200);
                        ctx2.strokeRect(150,150,120,120);
                         }
}
</script>
...
<canvas id = "canva" width = "400" height = "400" style = "border: 1px solid black;"></canvas>
<button onclick = "draw1()"> draw </button>
...
```

Локальне сховище. Перед використанням локального сховища доцільно перевірити підтримку цього API в браузері:

```
if (window [ 'localStorage']! == null) {
        // робота зі сховищем
                                    }
else alert ( "Сховище не підтримується");
```

Робота зі сховищем забезпечується такими методами і властивостями об'єкта localStorage:

clear() – видалення всіх пар ключ – значення зі сховища;

getItem("ключ") – отримання поточного значення, пов'язаного з ключем;

key(i) – отримання ключа за заданим індексом у колекції;

removeItem("ключ") – видалення пари ключ – значення з колекції сховища;

setItem("ключ","значення") – задання пари ключ – значення;

length – довжина списку ключ – значення;

remainingSpace – решта обсягу пам'яті (в байтах);

подія onstorage – зініціюється в об'єкті document зі зміною змісту в локальному сховищі.

Використання фонових обчислень. Недоліком організації обчислень в браузері є виконання всіх скриптів в одному потоці без можливості розпаралелювання. Для подолання цього недоліку до складу API HTML5 включені Web workers. Сутність реалізованого підходу полягає в тому, що скрипт, розміщений в окремому файлі, завантажується в браузер і виконується в окремому потоці. Для управління цим потоком створюється спеціальний об'єкт (var myWorker=new Worker("worker.js");). Запуск потоку здійснюється автоматично після створення об'єкта, знищення виконується методом myWorker.terminate(). Взаємодія сторінки з цим потоком організовується шляхом обміну повідомленнями (подія message та метод postMessage). Для отримання та оброблення повідомлень на основній сторінці і в запущеному потоці необхідно використовувати відповідні обробники подій. У повідомленнях можна передавати як рядки, так і об'єкти.

Приклад сторінки, яка управляє фоновими обчисленнями (скрипт фонового процесу поміщений в файл task.js):

```
<! DOCTYPE HTML>
<html>
```

```
<head>
<title> Worker </title>
</head>
<body>
<fieldset>
  <legend> Управління паралельним процесом </legend>
  <button onclick = "start ();"> почати виконання </button>
  <button onclick = "stop ();"> завершити виконання </button>
</fieldset>
<script>
var w;
function start()                    // запуск фонового потоку
{w = new Worker("task.js");     // створення об'єкта, власне запуск
w.onmessage=function(event)  // обробник для отримання даних
                                    // від фонового потоку
{dat_in=event.data;};        // отримання даних від фонового потоку
                                    // тут д.б.н. обробка та підготовка даних
w.postMessage(dat_out); //відправка повідомлення з даними фонового потоку
}
function stop()                 // зупинка фонового потоку
{w.terminate();
w = undefined;
}
</script>
</body>
</html>
```

Скрипт для фонового потоку (файл task.js):

```
onmessage=function (ev) // обробник для отримання повідомлення з даними
{V=ev.data;            //отримання даних
...                    // дії фонового потоку
r="дані отримані і оброблені"; // формування відповіді
postMessage(r);     // відправка відповіді
};
```

Для скриптів, що запускаються таким способом (в окремому потоці), є ряд обмежень: відсутність доступу до дерева DOM основної сторінки й об'єктів window, document і parent. Доступні об'єкти navigator, location (тільки для читання), XMLHttpRequest і методи setTimeout(), setInterval().

Трасування роботи скриптів на сторінці можна виконати шляхом виведення часу початку та завершення робот для кожного фрагмента сценарію в консоль.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1) відпрацювати матеріал лекції;

2) обрати варіант індивідуального завдання, підготувати алгоритми та тексти сценаріїв для всіх пунктів завдання.

Під час виконання лабораторної роботи необхідно:

1) динамічне створення зображень на сторінках:

1.1) розробити сценарій динамічного створення зображень відповідно до вибраного варіанта;

1.2) доповнити сценарій функціями підтримки призначеного для користувача інтерфейсу для управління створенням зображення;

2) використання локального сховища в скриптах:

2.1) розробити інтерфейс для управління використанням даних у локальному сховищі;

2.2) розробити скрипт для підтримки призначеного для користувача інтерфейсу та створення зображення відповідно до вибраного варіанта;

3) організація паралельної роботи у скриптах:

3.1) розробити та відлагодити сторінку зі скриптом, що використовує Web workers;

3.2) з використанням засобів відлагодження переконатися в тому, що скрипт виконується в фоновому режимі.

Варіанти завдань до лабораторної роботи

Завдання для п. 1.

1. Розробіть скрипт для побудови графіка коливання курсу валют за рік.

2. Розробіть скрипт для побудови кругової діаграми з витратами на різні потреби.

3. Розробіть скрипт для побудови стовпчикової діаграми за результатами голосування.

4. Розробіть скрипт для побудови анімованого зображення, що реагує на дії користувача (наприклад, смайлик, що усміхається з наведенням курсору).

5. Розробіть скрипт для побудови точкового графіка, що позначається клацанням мишки на елементі canvas.

6. Розробіть скрипт для побудови лінії на траєкторії переміщення курсора мишки.

7. Розробіть скрипт для побудови зображення сонця, що переміщається на дузі (повинні змінюватися розташування і освітленість).

8. Розробіть скрипт, який створює на полотні художній напис (букви заповнені фоновим малюнком), що вводиться побуквенно з клавіатури.

9. Розробіть скрипт, що виводить графік поточково з ефектом анімації.

10. Розробіть скрипт, для виведення трансформованого зображення (фотографії).

Завдання для п. 2.

Скрипт, розроблений для п. 1 лабораторної роботи 10 або п. 3 лабораторної роботи 9, доповніть інтерфейсом, що забезпечує збереження та використання даних у локальній пам'яті.

Завдання для п. 3 розробіть самостійно.

Контрольні запитання

1. Назвіть формати зображень, які використовуються на WEB-сторінках.

2. Назвіть теги, які використовуються для показу зображень.

3. Що таке динамічна графіка?

4. Які атрибути може містити тег Canvas?

5. Сформулюйте дії скрипта з використання тега Canvas.

6. Назвіть основні функції створення зображення. Запишіть приклади.

7. Які анімаційні можливості зображень надає Canvas?

8. Дайте характеристику API, що дозволяють використовувати локальні сховища даних.

9. Дайте характеристику API WebStorage (можливості, особливості, обмеження).

10. Сформулюйте обмеження, які існують для Web workers.

11. Назвіть основні засоби, які можна використовувати для організації обчислень з Web workers.

Тема 8. Публікація WEB-сайта

Лабораторна робота 11
Дослідження процесу публікації сайта на хості

Мета – освоєння використання засобів, якими забезпечується передача та публікація розроблених сайтів.

Після виконання лабораторної роботи студент повинен:

знати:

порядок та особливості розміщення сайтів на платних і безкоштовних хостах;

призначення і можливості систем управління версіями;

способи та засоби поширення та передачі розроблених продуктів через Інтернет;

уміти:

розміщувати створені документи в мережі Інтернет;

вести розроблення з використанням систем управління версіями.

У результаті виконання лабораторної роботи у студента формуються **компетентності** з вибору засобів створення та місця розміщення сайтів і передачі їх замовнику.

Загальні відомості

Система контролю версій – це система, яка контролює та реєструє зміни в одному або декількох файлах і зберігає історію цих змін. Найбільшого поширення набула система Git. Для початку роботи з Git необхідно встановити на комп'ютері консольний клієнт – програму яка виконує всі дії за командою користувача. Для більшої зручності можна використовувати і графічні оболонки.

Git забезпечує роботу з локальним або віддаленим репозитарієм (сховищем версій файлів). Найбільш популярним мережевим репозитарієм є сервіс GitHub.

Для роботи в командному клієнті початок рядка введення команд відзначено знаком $; далі вводиться префікс git, ім'я команди та додаткові параметри. Наприклад, для створення сховища в поточному каталозі слід набрати: $ git init.

Основні команди git для терміналу:

git help commit – виклик довідки;

git init – створити git репозитарій в поточному каталозі;

git status – перевірка стану сховища;

git add index.php – додавання файлів у репозитарій;

git checkout – скасування змін;

git commit -m "опис" – зберегти версії;

git tag v1 – додає тег у поточні версії;

git tag – перегляд тегів;

git checkout v1 – перехід тегами, а не хешем;

git checkout <hash> – повернення до попередньої версії з використанням хешу;

git checkout master – перехід до останньої версії в гілці master;

git log – перегляд історії змін;

git log --pretty = oneline – однорядковий формат історії;

git branch – інформація про гілки;

git branch newMaster – створення нової гілки;

git branch -a – показати всі гілки;

git checkout style – перехід до гілки style;

git merge master – злиття поточної (style) та master;

git fetch origin – оновлення із загального сховища (синхронізація);

git merge origin/master – злиття змін;

git push shared master – відсилання змін до спільного репозитарію.

Сценарій організації командної роботи з використанням Git може виглядати таким чином.

Перший користувач (координатор) виконує такі дії:

створює на GitHub порожній репозитарій (origin) за допомогою інтерфейсу сайта;

під управлінням Git створює та заповнює локальний репозитарій, гілку master;

відсилає гілку master у створений на GitHub репозитарій (origin):

git remote add origin https://github.com/mvp46/rep_len.git

git push -u origin master

Другий користувач (учасник команди розробників) на іншому комп'ютері клонує репозитарій з GitHub; створює нову гілку; виконує свою частину розроблення та відсилає цю гілку на GitHub:

git clone https://github.com/mvp46/rep_len.git <їм пап>

git branch <їм гілки>
git checkout <їм гілки>
git push origin <їм гілки>

Координатор виконує злиття гілок, створених іншими на GitHub:

git branch -a
git clone https://github.com/mvp46/rep_len.git <їм пап>
git merge origin / <їм гілки>

Публікація сайту на хості. Послугу, яка полягає у наданні можливості розмістити свої файли на сервері провайдера, називають хостингом. У мережі існують як безкоштовні, так і платні хостинги.

Найбільш відомі безкоштовні вітчизняні хостинги: hostinger.com.ua; ho.ua; www.ayola.net; free.1gb.ua.

Платні вітчизняні хостинги: www.avahost.ua; www.ukraine.com.ua; www.hostlife.net; www.thehost.ua; www.s-host.com.ua.

Під час вибору хостингу слід звертати увагу на підтримувані технології (бази даних, протоколи, CMS, типи серверних скриптів і т. д.), розмір наданого дискового простору, дозволене число сайтів, нав'язування реклами, розмір оплати, швидкість доступу, якість доменних імен.

Завдання

З метою підготовки до лабораторної роботи необхідно:

1. Відпрацювати матеріал лекції.
2. Підготувати сторінки й тексти сценаріїв для всіх пунктів завдання.

Під час виконання лабораторної роботи необхідно:

1) використання системи контролю версій GIT:

1.1) встановити GIT;

1.2) освоїти роботу з консольним клієнтом, ознайомитися з графічними оболонками, виконавши розроблення скрипта з контролем версій;

1.3) створити репозитарій на GitHub і розмістити в ньому свій сайт;

1.4) організувати роботу команди через GitHub;

2) публікація сайту на хості:

2.1) знайти в Інтернеті два безкоштовних і два платних хостинга. Порівняти умови;

2.2) зареєструватися на безкоштовному хостингу та розмістити свій сайт.

Контрольні запитання

1. Сформулюйте призначення системи контролю версій.
2. Які проблеми допомагає вирішити Git?
3. Назвіть команди, що забезпечують основні можливості Git.
4. Що таке гілка у Git?
5. У чому відмінність команди git pull від git fetch?
6. Що таке хостінг? Хост?
7. Як можна переслати файли сайту на хост?
8. У чому полягає відмінність безкоштовного та платного хостингів?
9. Що таке доменне ім'я? Як його отримати?

Рекомендована література

Основна

1. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова / Д. Кирсанов. – Санкт-Петербург : Символ-Плюс, 1999. – 376 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. – Харків : Вид. ХНЕУ, 2009. – 84 с.

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

4. Нильсен Я. Дизайн WEB-страниц. Анализ удобства и простоты использования 50 узлов / Я. Нильсен, М. Тахир; пер. с англ. – Москва : Из "Вильямс", 2002. – 336 с.

Додаткова

5. Гультяев А. К. Macromedia Home Site 5. Инструмент подготовки веб-публикаций : практ. пособ. / А. К. Гультяев. – Санкт-Петербург : Учитель и ученик ; КОРОНА принт, 2002. – 1504 с.

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

7. Сакс Т. Дизайн и архитектура современного вебсайта. Опыт профессионалов / Т. Сакс, Г. Мак-Клейн; пер. с англ. – Москва : ИД "Вильяме", 2002. – 304 с.

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

Інформаційні ресурси

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

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

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

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ОСНОВИ ПРОЕКТУВАННЯ WEB-ВИДАНЬ

Методичні рекомендації до лабораторних робіт

для студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладач **Молчанов** Віктор Петрович

Відповідальний за видання *О. І. Пушкар*

Редактор *Н. І. Ганцевич*

Коректор *Т. А. Маркова*

План 2018 р. Поз. № 288 ЕВ. Обсяг 91 с.

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

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