ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»

«WEB- ДИЗАЙН»

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт
за освітньо-професійною програмою першого (бакалаврського) рівня освіти спеціальності
122 «Комп'ютерні науки»

Рекомендовано Науково-методичною радою ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»
(протокол № 6 від «04» липня 2023 р.)
Обов'язково до розміщення в репозитарії

Запоріжжя 2023

WEB-дизайн: Методичні рекомендації що до виконання лабораторних робіт за освітньо-професійною програмою першого (бакалаврського) рівня освіти спеціальності *122 «Комп'ютерні науки»* / Уклад. Кабак Л. В. Запоріжжя, ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА», 2023. 124 с.

Методичні вказівки включають тематику індивідуальних завдань, методичні пояснення щодо порядку їх виконання, критерії оцінювання виконаного індивідуального завдання, вимоги до його оформлення, включаючи зразок титульної сторінки.

Рекомендовано для студентів спеціальності 122 «Комп'ютерні науки першого (бакалаврського) рівня освіти.

Самостійне електронне текстове мережеве видання

Затверджено на засіданні кафедри
цифрових технологій та проектно-
аналітичних рішень
Протокол № 1 від «30» травня 2023р.

Узгоджено:
Секретар Редакційної ради


Малій Х. В.
«31» травня 2023 р.

Зміст

ВСТУП

Дисципліна «Web-дизайн» викладається студентам галузі знань 12 «Інформаційні технології» спеціальностей 122 «Комп'ютерні науки та інформаційні технології» та 151 «Інтелектуальні системи управління в гірничо-металургійному виробництві».

Курс «WEB-дизайн» знайомить студентів із загальними принципами створення web-документів та стандартів, що їх регламентують; структури і призначення засобів програмування для web-сайтів, стандартів їх взаємодії та основи синтаксису; дизайну просторових співвідношень, форм, кольорів, шрифтів і текстів об'єктів і їх елементів. Студент після проходження курсу повинен отримати навички роботи з комп'ютерними мережами та Інтернет; ознайомитись з HTML тегами, навчитись створювати стандартні веб-сторінки, навчитись форматувати текст за допомогою тегів HTML; навчитись працювати із кольорами, списками та посиланнями в HTML; навчитись працювати із таблицями, фреймами та формами в HTML; ознайомитись з технологією CSS, навчитись форматувати текст та зображення; навчитись форматувати CSS блоки; ознайомитись з основами веб-програмування, операторами JavaScript та PHP.

Методичні вказівки до лабораторних робіт з дисципліни «Web-дизайн» є важливою складовою частиною сукупності професійно-орієнтованих дисциплін, які формують фахові знання при підготовці бакалаврів у галузях в яких використовуються сучасні web-технології, таких як банківська справа, машинобудування, видобувна промисловість та інші.

Студенти які вивчають дисципліну " Web-дизайн " отримають знання по основним принципам ведення сучасних web-технологій, котрі використовуються у різноманітних областях виробництва, науки і техніки. У методичних вказівках описані основні принципи розробки та підтримки Web сайтів, які використовується в сучасних інформаційних системах.

Лабораторна робота № 1. Web-сервер Wampserver. Введення в HTML

Мета:
1) Навчитися встановлювати Wampserver, створювати власні домени і піддомени в програмі Wampserver.
2) Освоїти основні принципи роботи з програмою Visual Studio Code чи Notepad++;
3) Навчитися застосовувати теги HTML для структурування сторінки і форматування тексту.

Теоретичні відомості

1.1.1 Wampserver. Встановлення програмного продукту.

Програма Wampserver призначена для створення власних доменів на локальному комп'ютері, що дозволяє створювати сайти і тестувати їх працездатність, не маючи доступу до Інтернету.

Для встановлення програмного продукту Wampserver необхідно викачати дистрибутив з сайту https://www.wampserver.com/en/ і запустити установний файл. Під час встановлення необхідно вибрати наступні опції:

1. встановлення здійснити в папку C:\wamp64, а не в звичну папку C:\Program Files;
2. вибрати варіант запуску вручну, а не як службу.

Після встановлення Wampservera його слід запустити вручну і перевірити працездатність. Для цього в браузері необхідно набрати адресу тестової сторінки http://localhost/Wampserver/.

1.1.2 Створення власного домену на локальному комп'ютері

Після встановлення програми всі необхідні для її працездатності файли розташовуються в папці C:\wamp64. Завдяки цьому папку можна переносити з комп'ютера на комп'ютер, не порушуючи при цьому працездатність сервера.

Для створення власного домену в межах локального комп'ютера досить додати нову папку з ім'ям домену в каталог C:\wamp64\home. Наприклад myserver.ru (див. рис.1.1).

Файли, які повинні знаходитися на цьому сервері необхідно помістити в папку C:\wamp64\www. При створенні власного домену слід враховувати неприпустимість використання в імені домену пробілів, букв кирилиці і спеціальних символів.

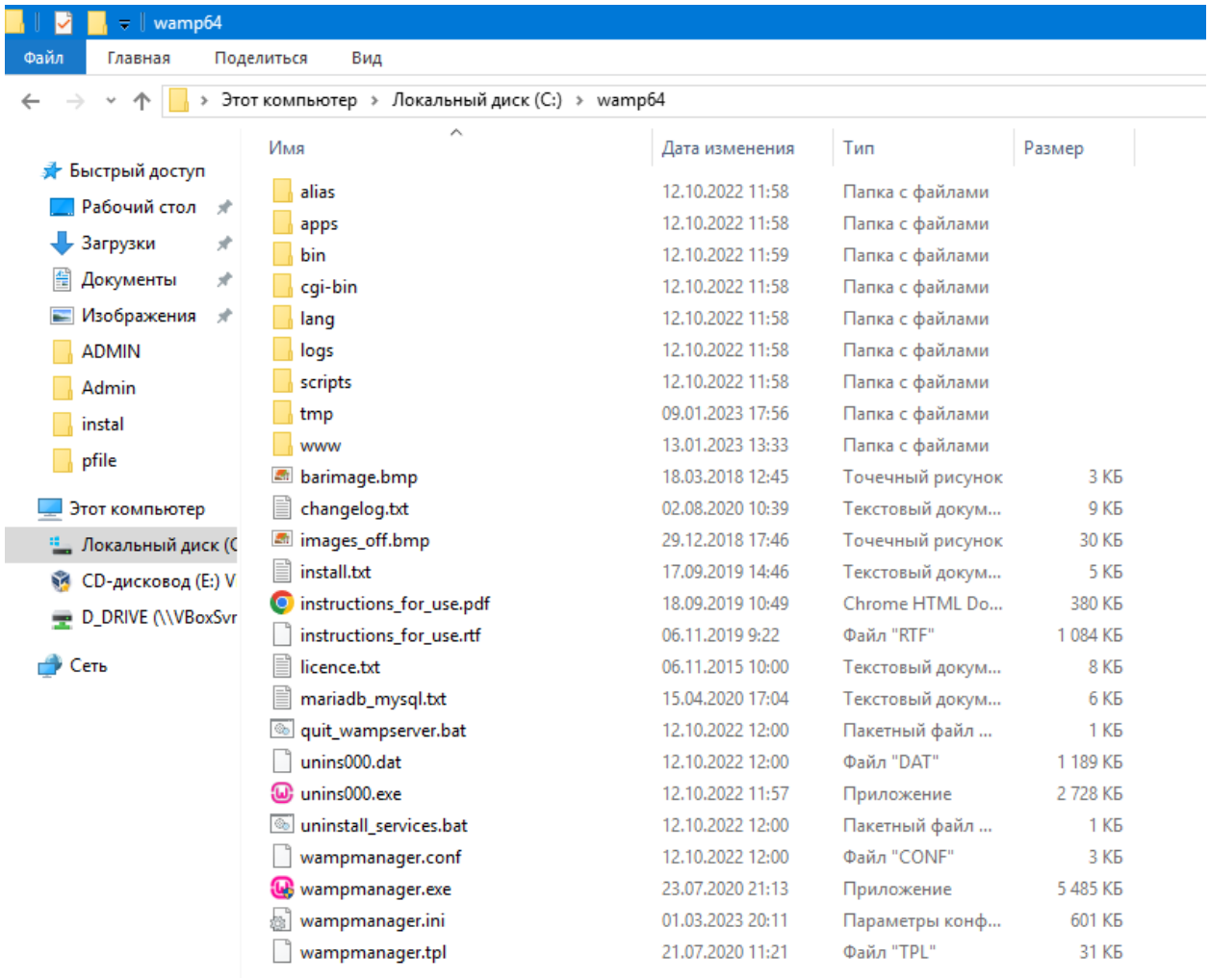


Рис.1.1. Розміщення доменів на Wampserver

Для перевірки працездатності нового домену слід:

а) перевантажити Wampserver (зробити це можна за допомогою ярлика Restart servers в меню Пуск або за допомогою файлу запуску C:\wamp64\wampmanager.exe);

б) відкрити в браузері сторінку з адресою нового домена (наприклад http://localhost/). При цьому в разі успішного запуску сервера на екрані відобразиться список файлів або зміст файлу з ім'ям index. На більшості сайтів при зазначені шляху до домену або до папки відображується файл index.html або index.php або вміст каталога (у випадку якщо індексний файл відсутній).

Варіант коли новий піддомен не відображується в браузері:

1. виконати команду ping на новий домен (ping http://localhost/). Якщо відповідь отримана, це означає що Wampserver з новими налаштуваннями успішно запущений. Якщо відповіді немає, то можливо а) Wampserver не був перезапущений з моменту додавання нового домену; б) ім'я теки відрізняється від імені пінгуємого домену; в) всередині домену не створена папка www.

1.1.2 Структура HTML-документа

Складання сторінок для сайту базується на мові HTML (від англ. Hypertext Markup Language – мова розмітки гіпертексту). Всі сторінки сайту мають розширення **.htm** або**.html.**

Основною структурною одиницею мови є тег з його атрибутами. Тег
являє собою розміщені в трикутних дужках інструкції мови, які повідомляють, якими властивостями повинен володіти той або інший фрагмент тексту на сторінці. Більшість тегів є парними: містять як відкриваючий, так і закриваючий тег (див. рис.1.2):

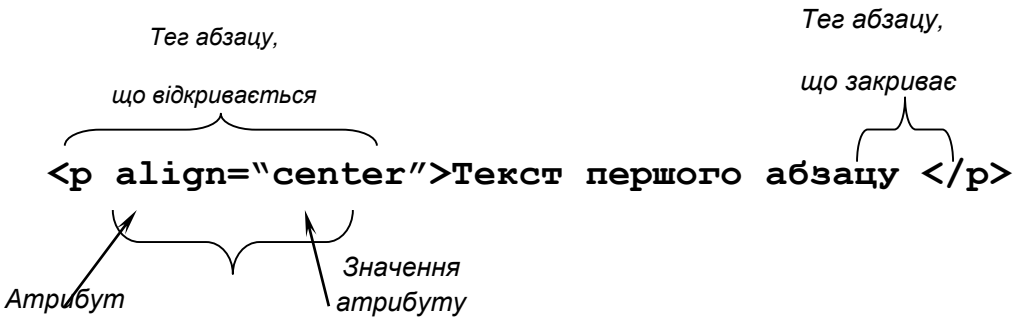


Рис.1.2. Структура тега

Слід зазначити, що написання значень атрибутів потрібно писати в подвійних лапках.

Опис всього документа будується з тегів. Весь документ має бути розміщений в головному тегу **<html>** (див. рис.1.3).

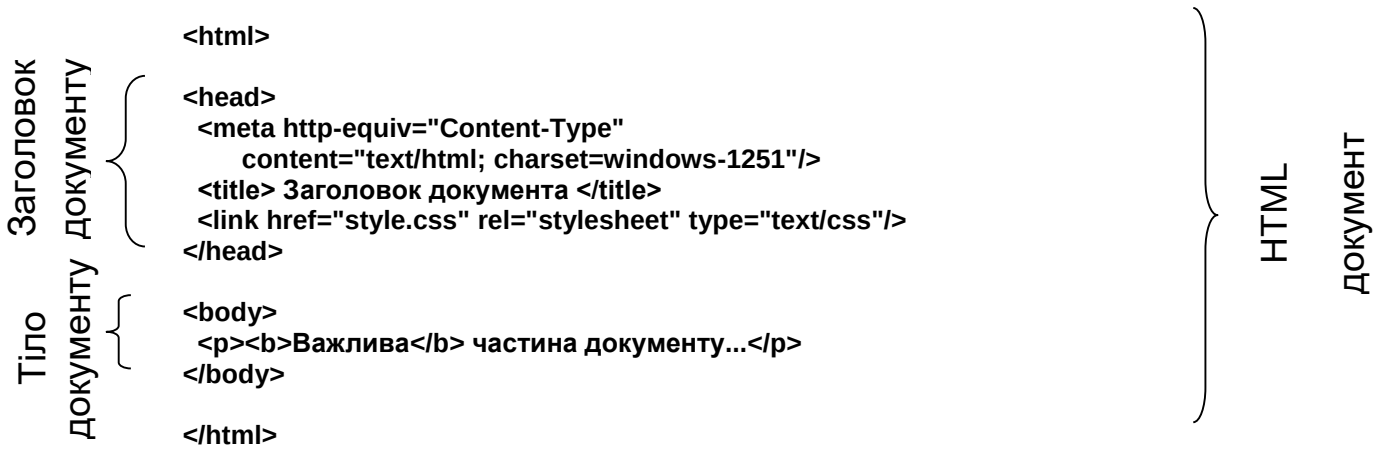


Рис.1.3. Структура html-документа

Усі теги можна умовно розділити на групи по їх функціональному призначенню: теги рівня документа, теги форматування, теги структурних елементів та ін.

1.1.3 Теги рівня документа

<html>...</html> парний тег-контейнер, який містить в собі всю веб-сторінку. Відкриваючий та закриваючий теги **<html>** в документі обов'язкові, для того щоб вказати межі документа.

<head>...</head> тег призначений для зберігання елементів, які використовуються для зберігання службової інформації, призначеною для браузерів і пошукових систем. Вміст тега не відображується на сторінці, за винятком тега **<title>.**

<title>...</title> **містить текст заголовка, який відображується в рядку заголовка вікна браузера.**

<body>...</body> призначений для зберігання вмісту веб-сторінки (контента), браузера, що відображується у вікні (форматований текст, зображення, таблиці і так далі). Тег **<body>** може містити безліч атрибутів, які впливають на відображення всього документа вцілому.

Таблиця 1.1

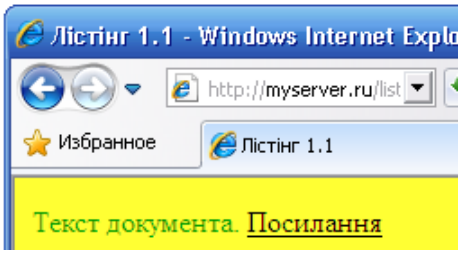
Основні атрибути тега <body>

Назва атрибуту	Призначення атрибуту
alink	колір активного посилання (колір задається шістнадцятиричним числом або константою, наприклад червоний: red або #ff0000)
background	фоновий рисунок на веб-сторінці
bgcolor	колір фону веб-сторінки
topmargin(leftmargin, bottommargin)	відступ від верхнього (лівого, нижнього) краю вікна браузера до контента
link	колір посилань на веб-сторінці
text	колір тексту в документі
vlink	колір відвіданих посилань

Наприклад, наступний фрагмент html-кода задає для всього документа колір тексту зелений, колір посилань чорний, колір фону жовтий. Лістинг 1.1.

Лістинг 1.1 – Приклад використання атрибутів тега <body>

```
1  <html>
2
3  <head>
4    <title>Лістінг 1.1</title>
5  </head>
6
7  <body  bgcolor="#FFFF33" text="#009900" link="black">
8    Текст документа. <a href="#">Посилання</a>
9  </body>
10
11 </html>
```



1.1.4 Основні теги форматування тексту

Таблиця 1.2

Теги форматування тексту

Назва тега	Призначення тега, атрибути
... або ...	встановлює жирне зображення шрифту
...	встановлює курсивне зображення шрифту
<u>...</u>	встановлює підкреслене зображення шрифту
<strike>...</strike>	встановлює закреслене зображення шрифту
< cite>...</cite>	позначає текст як цитату, зазвичай відображується курсивом
<code>...</code>	призначений для відображення тексту, який є програмним кодом, зазвичай відображується моноширинним шрифтом
~~...~~	використовується для виділення тексту, який був видалений в новій версії документа
< ins>...</ins>	використовується для виділення тексту, який був доданий в нову версію документа
<dfn>...</dfn>	застосовується для виділення термінів при їх першій появі в тексті
^{...}	відображує шрифт у вигляді верхнього індексу
_{...}	відображує шрифт у вигляді нижнього індексу

<pre>...</pre>	визначає блок тексту, в якому зберігається кількість пропусків між словами, задана на етапі створення документа
< nobr>...</nobr>	повідомляє браузер відображувати текст в один рядок без перенесення
... ...	встановлює заголовки різного рівня значущості від найкрупнішого h1 до найдрібнішого h6
...	призначений для встановлення характеристик шрифту, які задаються в атрибуті тега. Тег має наступні атрибути: **color=…** задає колір шрифту (у шістнадцятиричній формі, наприклад, #ff0000) **face=…**задає гарнітуру шрифту(Arial,Tahoma) **size=…** задає розмір шрифту в умовних одиницях (ціле число від 1 до 7 або зміни числа +1 або -2) найкрупніший шрифт 7
...	визначає межі абзацу. Тег має атрибут **align,** який відповідає за вирівнювання тексту і може набувати значень: **left, right, center**, **justify**
 	встановлює примусове перенесення рядків
< hr>	малює горизонтальну лінію. Параметри горизонтальної лінії можна задати з використанням атрибутів тега: **align=…** визначає вирівнювання лінії **color=…** колір лінії **noshade=…** малює не тривимірну лінію **size=…** товщина лінії в пікселях **width=…** ширина лінії в пікселях

При встановленні розмітки тексту теги можна писати вручну, або користуватися панелями інструментів HTML, HTML Formatting, HTML Tables і HTML Forms. Наприклад, для встановки жирного зображення тексту слід:

1. виділити фрагмент тексту, який слід зробити жирним;
2. натиснувати на панелі інструментів HTML Formatting кнопку з вказівкою жирного зображення;
3. в результаті необхідні теги будуть додані в документ;
4. якщо є необхідність задати атрибути тега, то зробити це можна за допомогою палітри Inspector. Виділити тег, якому слід змінити атрибути і встановити їх значення.

1.1.5 Спеціальні символи і коментарі

Ряд символів в мові використовується в службових цілях (знаки більше і менше), а ряд символів неможливо набрати на клавіатурі (нерозривний пробіл, копірайт). Для відображення таких символів використовують спеціальні коди, які пишуть під час верстки html-сторінок (див. Таблиця 1.3):

Таблиця 1.3

Спецсимволи мови HTML

Ім'я	Код	Вигляд	Опис
&quot;	&#34;	"	подвійна лапка
&amp;	&#38;	&	амперсанд
&lt;	&#60;	<	знак 'менший'
&gt;	&#62;	>	знак 'більший'
			нерозривний пропуск
&sect;	&#167;	§	параграф
&сміттю;	&#169;	©	знак copyright
&reg;	&#174;	®	знак зареєстрованої торгівельної марки
&deg;	&#176;	°	градус
&plusmn;	&#177;	±	плюс-мінус
&trade;	&#8482;	™	торгівельна марка
&deg;	&#176;	°	градус

Такі символи можна додавати як у вигляді імені, так і у вигляді коду. У програмі Visual Studio Code чи Notepad++ чи Notepad++для цього можна використовувати команду Insert ⇒ Character ⇒ Special Character.

Для коментарів або тимчасового виключення тексту з html-сторінки використовують пари символів: <!--Текст коментаря-->.

Порядок виконання роботи

1. Запустіть веб-сервер: Пуск ⇒ Wampserver ⇒ Start (або C:\wamp64\wampmanager.exe). Коли він запуститься в системному треї, у вас з'явиться значок у вигляді малинового пера.
2. Перевірте роботу Wampservera виконавши в командному рядку команду **ping** 127.0.0.1. Якщо відповідь походить від адреси 127.0.0.1, то це означає, що Wampserver запущений і успішно працює на локальному комп'ютері.
3. Відкрийте в браузері адресу **http://localhost/**.
4. Оновіть поточну сторінку з адресою **http://localhost/**. (натисніть F5).
5. Створіть свій домен C:\wamp64\www\lab_1 (потрібно створити папку). У цій папці вашого каталога створіть файл **index.html.** Саме з цього файлу завжди починається завантаження вмісту домену, якщо не вказанний інший файл.
6. Відкрийте файл **index.html** за допомогою програми Notepad++.
7. Запишіть в ньому наступні рядки:

```
<html>
   <head>
      <title> Заголовок вікна </title>
   </head>

   <body>
      Освоюємо web-технології  :)
   </body>
</html>
```

Заголовок вікна браузера

Те, що буде відображатися всередині вікна браузера

8. Збережіть файл. Перевірте в браузері результат, набравши в адресному рядку ім'я вашого сервера C:\wamp64\www\lab_1. У разі виникнення проблем із завантаженням сервера зверніться до теоретичних відомостей.
9. Зверніть увагу на структуру вікна в програмі Visual Studio Code чи Notepad++.
10. Встановіть курсор усередині тега **<body>** та прогляньте які атрибути є в цього тега. Змініть атрибути **text** (білий) і **bgcolor** (синій). Збережіть зміни. Перегляньте результат на вкладці Preview і в браузері. Таким чином, додаючи теги і змінюючи їх атрибути можна створити документ будь-якої складності.

11. Всі файли і зображення, які з'являтимуться на вашій веб-сервері-сторінці, повинні розташовуватися всередині папки **<ім'я домена>/www/Lab_1** і мати адреси починаючи з кореня сайту, а не з кореня диска C: або D: Виберіть фонове зображення, скопіюйте його в папку **www** і змініть фон сторінки (атрибут **background** в тезі **body**). Збережіть і прогляньте результати роботи в браузері.
12. Використовуючи інформацію текстового файлу з архіву до лабораторної роботи відформатуйте вміст сторінки таким чином, щоб вона була схожа на рис.1.6 (детальний опис починається з пункту 17). При форматуванні використовуйте теги з теоретичних відомостей, а також детальний опис тегів з домена **html.ua;**
13. Створіть новий документ з ім'ям **index.html** і збережіть його в корені свого домену.
14. Між відкриваючим та закриваючим тегом **<body>** додайте текст з файлу. Збережіть його і прогляньте в браузері що вийшло.
15. Змініть фон сторінки і колір тексту згідно рисунку.
16. Розбийте текст на абзаци використовуючи тег **<p>**.
17. Зверніть увагу на те, щоб у вас не було червоних (не закритих) тегів.
18. Встановіть для абзаців вирівнювання. Для цього встановіть курсор всередину тега абзацу, що відкривається (у тег **<p>**). Перейдіть на панель Inspector і в атрибуті **aling** виберіть потрібне значення. Якщо значення вирівнювання по ширині не вибирається із списку, то встановіть його вручну: **align="justify**".
19. Встановіть словам жирне, підкреслене і курсивне накреслення.
20. У другому абзаці використайте верхній індекс.
21. Використовуючи спеціалізовані теги виділіть цитату і програмний код. Використовуйте додатковий тег **<pre>** для збереження числа пропусків між словами в програмному коді.
22. Використовуючи тег **<font>** виділіть назви пошукових систем.
23. Змініть колір яндекса на жовтий.
24. Додайте в документ дві лінії, першу завдовжки в 400 пікселів, а заввишки в 5 пікселів. А другу і третю шириною 50%.
25. Використовуючи теги **<h1>...<h6>** виділіть заголовки.
26. Використовуючи коди спеціальних символів відобразіть на сторінці знак менше. І підпишіть виконане завдання як вказано на рисунку 1.6.
27. Перевірте, що всі файли у вас доступні через сервер з використання доменного імені.
28. Покажіть виконану роботу викладачу.

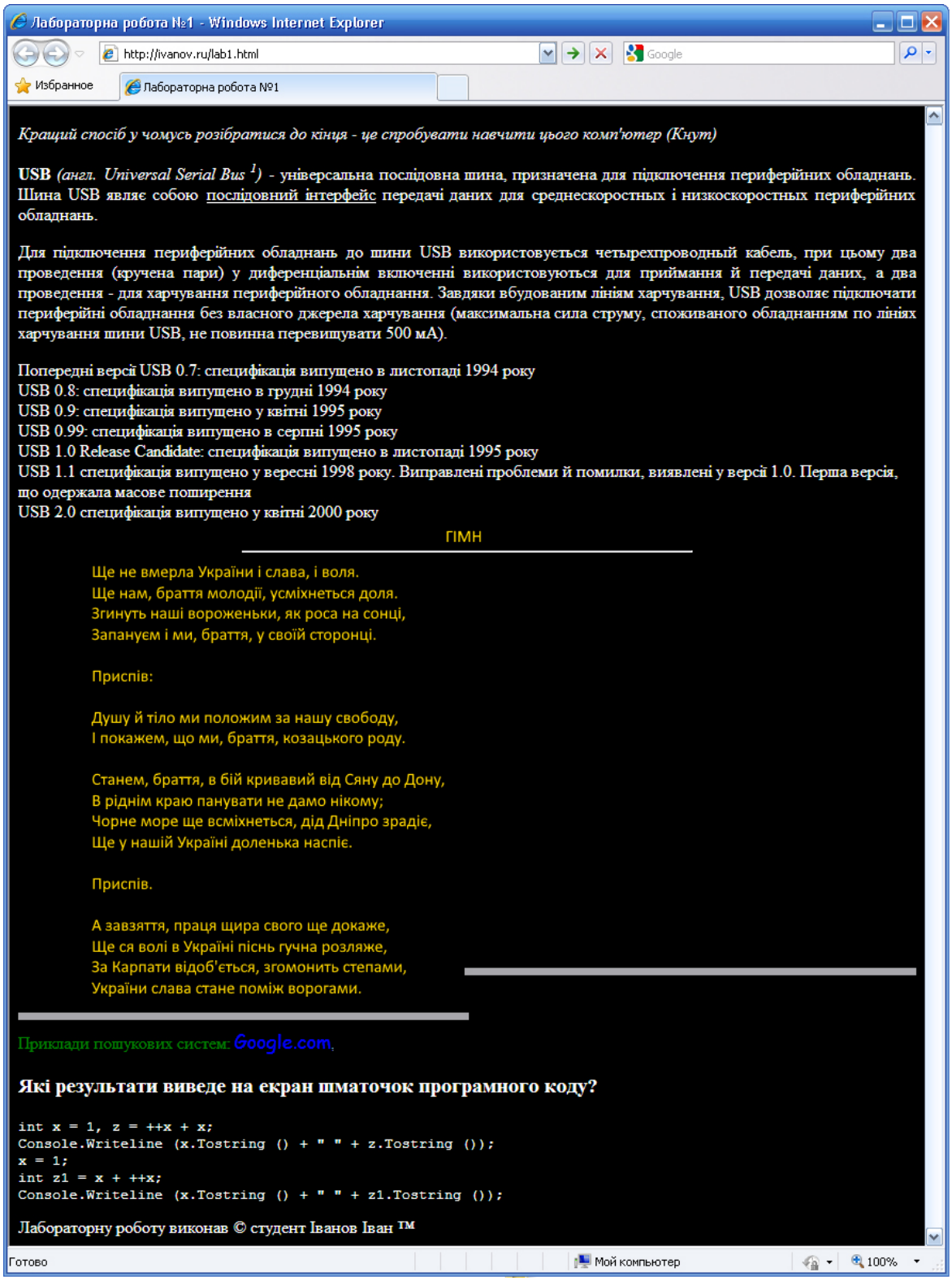


Рис.1.4. Зразок для виконання практичного завдання

Контрольні питання

1. Як додати до Wampserver свій домен?
2. Як перевірити працездатність нового домену?
3. Яку програму можна використовувати для створення html-сторінок?
4. Що таке тег?
5. Які існують теги рівня документа?
6. Якими тегами можна виділити текст жирним, курсивом, підкресленням?
7. Які атрибути використовуються тегом **<body>**?
8. Якими тегами можна відзначити верхній і нижній індекси?
9. Який тег дозволяє змінити одночасно шрифт, колір тексту і розмір тексту?
10. У чому відзнака тегів **
** і **<p>**?
11. За допомогою чого можна управляти вирівнюванням абзацу?
12. Як додати в документ горизонтальну лінію?
13. Що таке спеціальні символи і як їх можна додати в документ?
14. Між якими символами розташовується текст коментарів?

Лабораторна робота № 2 Зображення. Списки. Посилання

Мета:
1) Навчитися додавати зображення в документ і управляти атрибутами тега <img>;
2) Навчитися створювати нумеровані, маркеровані і багаторівневі списки;
3) Навчитися створювати внутрішні і зовнішні посилання.

Теоретичні відомості

2.1.1 Зображення

Вставка картинок в html-документ здійснюється з використанням тега **<img>.** Тег можна записати вручну і потім за допомогою автодоповнення коду або за допомогою панелі Inspecrot вказати його атрибути. Можна також скористатися кнопкою Image на панелі інструментів HTML програми Visual Studio Code чи Notepad++.

Тег **<img>** має наступні атрибути (див. таблицю 2.1):

Таблиця 2.1

Атрибути тега <img>

Атрибут	Призначення
src	вказує на напрямок до графічного файлу. Напрямки можуть бути відносними: photo.jpg, ../img/photo2.gif або абсолютними: http://mysite.com/pic/photo3.jpg
lowsrc	напрямок до графічного файлу гіршої якості (і меншого розміру), який вантажиться перед повнорозмірною картинкою
alt і title	альтернативний текст для зображення, використовується як спливаюча підказка або пошуковими системами
border	товщина рамки довкола зображення. Колір рамки визначається поточним кольором тексту
hspace	горизонтальний відступ від зображення до тексту
vspace	вертикальний відступ від зображення до тексту
height	висота зображення, якщо не задана, то відображується оригінальний розмір зображення, якщо вказана до зображення масштабується
width	ширина зображення, якщо при вказівці ширини одночасно з висотою не зберігаються пропорції, то зображення спотворюється
align	визначає як малюнок буде вирівнюватися і обтікатися текстом

2.1.2 Списки

У мові HTML розрізняють наступні види списків:
а) маркіровані списки;
б) нумеровані списки;
в) списки визначень.

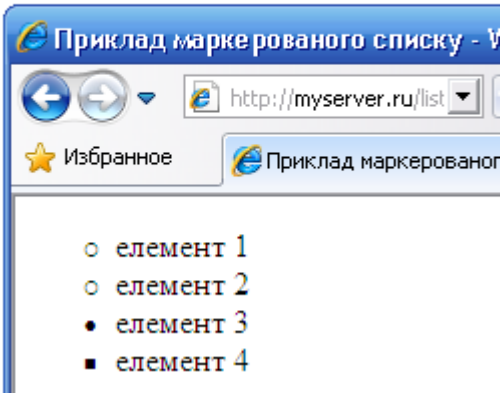
Для створення маркірованих списків застосовують теги **<ul>** і **<li>.** Тегом **<ul>** наголошується початок і закінчення всього списку. Тегом **<li>** позначають початок і кінець окремого елементу списку.

За замовчуванням елементи списку маркіруються чорним кружечком. За допомогою атрибуту **type** можна змінити стиль маркірування. Приклад використання списку наведений в Лістингу 2.1:

Лістинг 2.1 – Приклад маркірованого списку

```
<html>
<head>
  <title>Приклад маркерованого списку</title>
</head>

<body>
<ul type="circle">
    <li>елемент 1</li>
    <li>елемент 2</li>
    <li type="disc">елемент 3</li>
    <li type="square">елемент 4</li>
</ul>
</body>
</html>
```

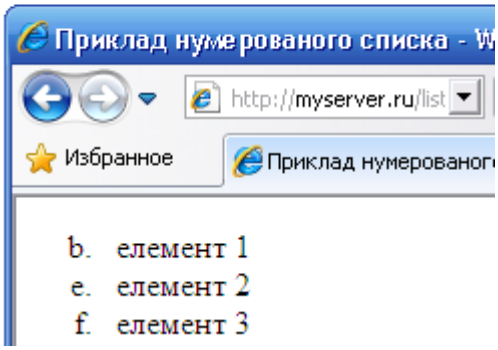


Аналогічно маркірованих списків, використовуючи теги **<ol>** і **<li>** можна створювати нумеровані списки. Для створення варіативної нумерації використовують атрибут **type** для вибору стилю нумерації і **start** для вказівки того, з якого символу слід почати нумерацію списку. Для тега **<li>** атрибут **value** дозволяє задати номер поточному елементу списку.

Лістинг 2.2 – Приклад використання нумерованого

```
<html>
<head>
  <title>Приклад нумерованого списка</title>
</head>

<body>
<ol start="2" type="a">
    <li>елемент 1</li>
    <li value="5">елемент 2</li>
    <li>елемент 3</li>
</ol>
</body>
</html>
```



Списком визначень є текст, що складається з двох взаємопов'язаних наборів – списку з термінами і списку визначень термінів. Спочатку вказується перший термін, нижче за нього йде його визначення, потім наступний термін з визначенням і так далі

Структура списку визначень наступна:

Термін 1
    Визначення терміну 1
Термін 2
    Визначення терміну 2

Сам список задається за допомогою тега **<dl>,** термін – тегом **<dt>,** а його визначення – за допомогою тега **<dd>**. Вкладення тегів для створення списку визначень продемонстроване в лістингу 2.3:

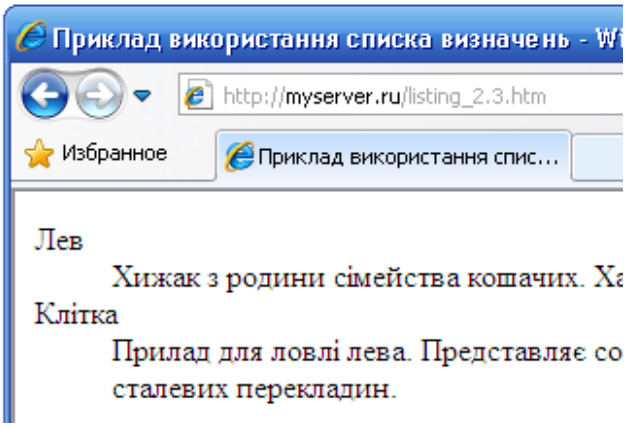
Лістинг 2.3 – Приклад використання списку визначень

```
<html>
<head>
  <title>Приклад використання списка визначень</title>
</head>

<body>
<dl>
    <dt>Лев</dt>
    <dd>Хижак з родини сімей

    <dt>Клітка</dt>
    <dd>Прилад для ловлі лев
</dl>

</body>
</html>
```



2.1.3 Посилання

Посилання або Гіперзв'язок (Link, Hyperlink) – фрагмент тексту або графіки на HTML-сторінці що посилається на іншу позицію в тому ж документі або на об'єкт в іншому документі (можливо навіть розташованому на іншому сервері).
Для створення посилань в мові HTML використовують тег **<a>** (anchor – якір). Атрибути, які використовуються в посиланні приведені в таблиці 2.2.

Таблиця 2.2

Атрибути тега <a>

Атрибут	Призначення
href	задає адресу документа, на який слід перейти. Є обов'язковим атрибутом для тега **<a>**. Може містити як відносну, так і абсолютну адресу сторінки. А також протоколи відмінні від http, наприклад ftp, mailto, ed2 та ін.
name	встановлює ім'я якоря, для визначення позиціювання посилання усередині документа.
target	встановлює ім'я вікна або фрейма, в якому буде завантажений документ. Можливі також варіанти використання: _self для відкриття посилання в тому ж вікні і _blank для відкриття посилання в новому вікні.
title	додає спливаючу підказку до тексту посилання

Лістинг 2.4 – Приклад використання посилань

```
<html>
<head>
  <title>Тег A, атрибут name</title>
</head>

<body>
  <p><a name="top"></a>
  В эту точку будет осуществлен переход при щелчке на нижнюю ссылку</p>

  <a href="http://google.com" title="" target="_blank"
            title="ссылка откроется в новом окне">Поисковая система google</a>

  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  tincidunt ut laoreet dolore magna aliguam erat volutpat.</p>

  <p><a href="#top">Наверх</a></p>
</body>
</html>
```

Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть в своєму домені каталог C:\wamp64\www\lab_2 а в нйому файл **lab_2.html**.
3. Відредагуйте файл **lab_2.html**. Вставте в документ зображення (див варіанти завдань для самостійної роботи) з урахуванням: ширини, висоти, вирівнювання, спливаючих підказок та ін.
4. Створіть список визначень згідно малюнку. Використовуйте для цього теги **<dl>**, **<dd>** і **<dt>.**
5. Створіть маркірований список використовуйте тег **<ul>**. Змініте зовнішній вигляд маркера на квадратик.
6. Створіть нумерований список. Здійсніть нумерацію римськими цифрами і змініть початок нумерації з числа XV.
7. Створіть список з багаторівневою нумерацією.

8. Позначте початок кожного розділу сьогоднішньою лабораторною як іменований якір: **<a name="image">**, **<a name="list">**.
9. На початку сторінки додайте список посилань на всі розділи лабораторної роботи. Пам'ятаєте, що при посиланні на іменований якір слід використовувати знак # перед ім'ям якоря.
10. Додайте внизу сторінки посилання на завантаження інсталяційного файлу Wampservera.
11. Додайте внизу документа посилання на свою сторінку в контакті, яка відкриватиметься в новому вікні.
12. Друге зображення в документі зробіть посиланням на пошукову систему Google
13. Створіть новий файл **index.html** у каталозі **C:\wamp64\www\lab_2**, у якому розробіть список, що містить посилання на завдання двох лабораторних робіт.
14. Створіть новий файл **my.html** і запишіть в ньому: нумерований список з відомих вам мов програмування, маркірований список з відомих вам палітр кольорів.
15. Додайте посилання на файл **my.html** у файл **index.html**.
16. Покажіть виконану роботу викладачеві.

Контрольні питання

1. Який тег служить для вставки зображення в html-документ?
2. Де в тезі **<img>** вказується шлях до графічного файлу?
3. Як задати вирівнювання картинки відносно тексту?
4. Які списки існують в HTML?
5. Який тег бере участь в створенні як маркірованого так і нумерованого списків?
6. Як змінити порядок нумерації в нумерованому списку?
7. Як змінити вигляд маркера в маркірованому списку?
8. Що таке списки визначень?
9. Як задати якір для посилань в межах одного документа?
10. За що відповідає атрибут **target** в тезі **<a>**?
11. Як перевірити працездатність нового домену?

Лабораторна робота № 3 Таблиці HTML

Мета:
1) Навчитися створювати таблиці заданого розміру;
2) Навчитися об'єднувати клітинки по горизонталі і вертикалі;
3) Навчити створювати фіксовані і динамічні таблиці;
4) Навчитися управляти відстанню між клітинками і рамкою таблиці.

Теоретичні відомості

3.1.1 Таблиці. Загальні відомості

Таблиці в HTML організовуються як набір стовпців і рядків. Елементи таблиці можуть містити будь-які HTML-елементи, такі, як заголовки, списки, абзаци, фігури, графіку, а також елементи форм. Таблиці з невидимою межою довгий час використовувалися для верстки веб-сторінок, дозволяючи розділяти документ на модульні блоки. Подібний спосіб вживання таблиць знайшов втілення на багатьох сайтах, але йому на зміну не прийшов сучасніший спосіб верстки за допомогою шарів. Для створення таблиць користуються тегом **<table>,** який визначає початок і кінець таблиці. Основні атрибути тега **<table>** приведені в таблиці 3.1.

Таблиця 3.1

Атрибути тега <table>

Атрибут	Призначення
align	визначає вирівнювання таблиці відносно всього документа
width	ширина таблиці у відсотках або в пікселях
height	висота таблиці. Часто не підтримується браузерами. У випадку якщо розмір тексту перевищує розмір таблиці – таблиця розтягується на висоту тексту
background	фоновий малюнок для всієї таблиці
bgcolor	задає колір фону таблиці
border	ширина рамки довкола таблиці, може бути значно товще ніж ширина рамки між клітинками
cellspacing	відстань між клітинками
cellpaddiing	відступ від рамки до вмісту клітинки

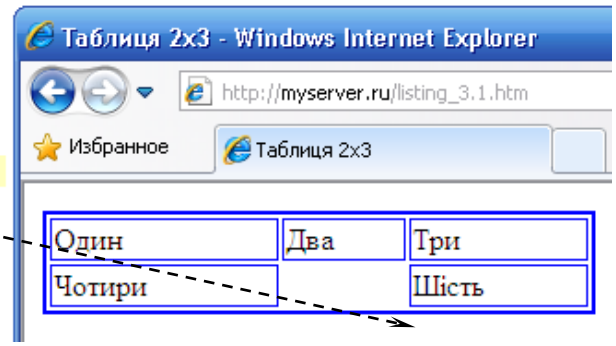
При побудові таблиця розділяється на рядки (тег **<tr>**), які у свою чергу діляться на клітинки (тег **<td>**). Приклад таблиці наведений в лістингу 3.1.

Лістинг 3.1 – Реалізації таблиці на два рядки і три стовпці

```
1  <html>
2  <head>
3    <title>Таблиця 2x3</title>
4  </head>
5  <body>
6    <table width="300" border="2" bordercolor="#0000FF">
7      <tr>
8        <td>Один</td>
         <td>Два</td>
         <td>Три</td>
       </tr>
12     <tr>
13       <td>Чотири</td>
         <td></td>
         <td>Шість</td>
       </tr>
     </table>
18 </body>
19 </html>
```

перший

другий



У наданій на лістингу таблиці, за допомогою атрибутів встановлені фіксована ширина, рамка, шириною в два пікселі і колір рамки. При заповненні клітинок, якщо між парою тегів **<td>...</td>** немає графічних символів або спеціального символу нерозривний пробіл – межа клітинки на сторінці не відображатиметься.

Замість тексту всередині клітинки може бути будь-який елемент: список, зображення, інша таблиця та ін.

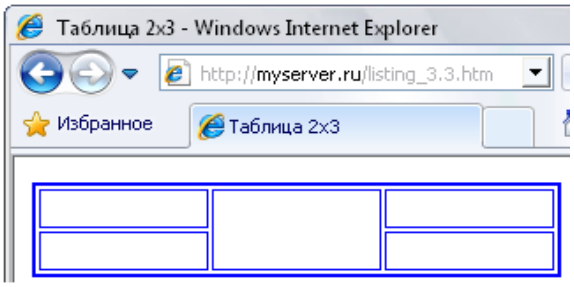
Тег **<tr>** служить контейнером для створення рядка таблиці. Кожна клітинка в межах такого рядка може задаватися за допомогою тега **<th>** або **<td>.** Можливі атрибути тега **<tr>: align, valign**, **bgcolor, bordercolor**. Можливі атрибути для тегів **<td>** і **<th>: align, background, bgcolor, bordercolor, colspan, height, nowrap, rowspan, valign, width.**

3.1.2 Об'єднання елементів таблиці

Для об'єднання елементів таблиці використовують атрибути colspan – для вказівки числа об'єднуваних клітинок по горизонталі і rowspan – для вказівки числа об'єднуваних клітинок по вертикалі. Приклад об'єднання елементів таблиці приведені в лістингах 3.2 і 3.3.

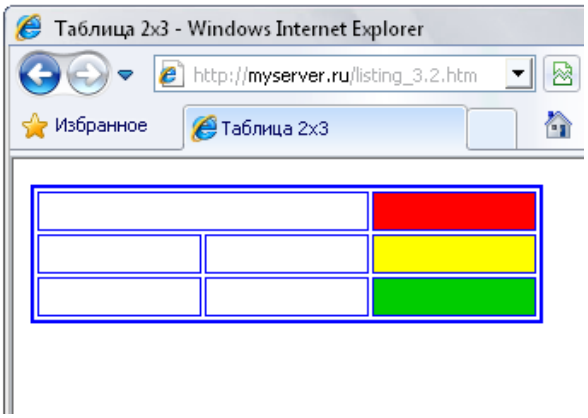
Лістинг 3.2 – Об'єднання клітинок по вертикалі

```
<table width="300" border="2" bordercolor="#0000FF">
  <tr>
    <td> </td>
    <td rowspan="2"> </td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
  </tr>
</table>
```



Листинг 3.3 – Об'єднання клітинок по горизонталі

```
<table width="300" border="2" bordercolor="#0000FF">
  <tr>
    <td colspan="2"> </td>
    <td bgcolor="#FF0000"> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td bgcolor="#FFFF00"> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td bgcolor="#00CC00"> </td>
  </tr>
</table>
```



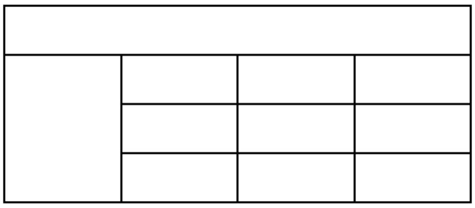
Приклад створення таблиці для створення сайту, буде розглянутий в практичній роботі

Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог C:\wamp64\www\lab_3 та файл **lab3.htm** і збережіть його в своїй папці на веб-сервері C:\wamp64\www\lab_3.
3. Створіть у файлі **lab3.htm** таблицю розміром 3x3 для гри в «хрестики-нулі». Для створення таблиці використовують тег **<table>.** Для таблиці необхідно створити рамку для відображення меж. Порожні клітинки заповніть символом нерозривного пробілу.
4. Доповнити файл таблицею з об'єднаними клітинками і жирнішою рамкою для всієї таблиці, текст вирівняти по середині клітинки (вертикально і горизонтально).

Стать	Середні покажчики	
	Зріст	Вага
Чоловік	174	78
Жінка	165	65

5. Створити таблицю, встановити їй наступні параметри: розташування на екрані – по центру, ширина 70% від ширини екрану. Колір вертикальної спільної клітинки – блакитний, спільної горизонтальної клітинки – жовтий. Колір останньої частини таблиці – зелений. Вирівнювання тексту у клітинках – по правому краю.



6. Створити в окремому файлі таблицю для шаблону сайту, яка відповідатиме наступним вимогам (див. рис.3.1).

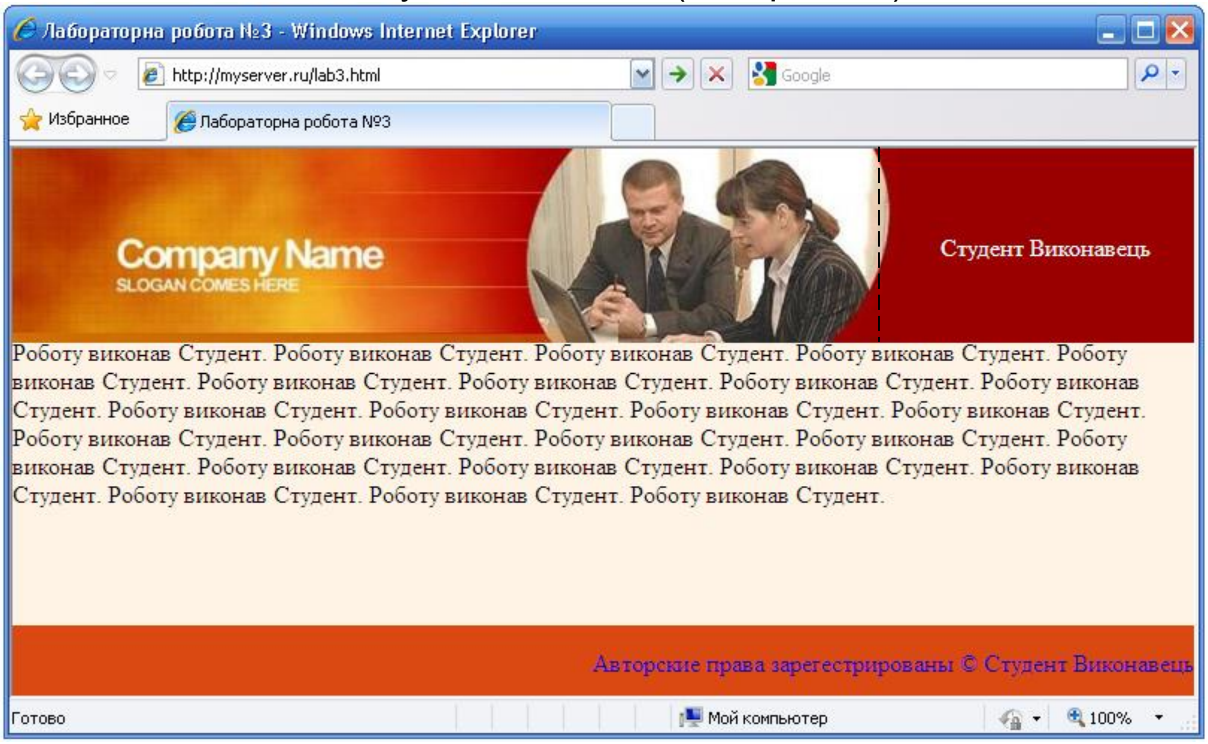
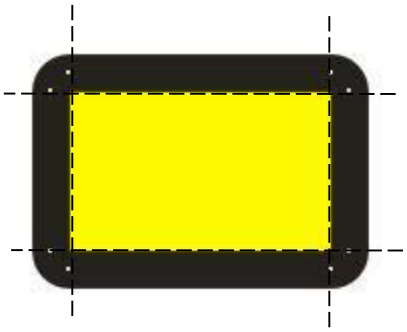


Рис.3.1. Шаблон сайту

- шапка таблиці повинна складатися з двох клітинок;
- розмір лівої верхньої клітинки задати залежно від розміру зображення;
- фоном цієї клітинки зробити зображення (атрибут **background** для тега клітинки **<td>**);
- права верхня клітинка повинна займати місце, що залишилося у вікні браузеру і мати колір фону #.... (дізнатися колір фону можна використовуючи кнопку **select** у вікні вибору кольору фону);
- для того щоб прибрати відстань між клітинками використовуйте атрибути **cellpadding** і **cellspacing** тега **<body**>;
- у верхній правій клітинці змініть колір тексту і його вирівнювання (згідно рисунку) і запишіть в ньому своє прізвище. Для вирівнювання тексту необхідно використовувати атрибути **aling** і **valign**;

- центральну частину (ту, що містить основний текст) зробити заввишки в 350 пікселів і задати колір фону **#fff3e5**;
- у центральній клітинці об'єднати два клітинки так, щоб вийшла одна спільна, встановити вирівнювання тексту по ширині і по верхньому краю клітинки;
- задати підвал таблиці заввишки в 30 пікселів, текст вирівняти по правому краю. Включити в підпис символ ©.



7. У окремому файлі створити таблицю розміром 3x3 (пунктиром вказані кордони клітинок).
8. Задати ширину і висоту таблиці і клітинок так, щоб ширина таблиці зменшувалася і збільшувалася при зміні розмірів вікна браузера, а висота була встановлена рівною висоті браузера. Якщо клітинка порожня, то вона не відображуватиметься на екрані, а якщо у клітинці знаходиться пробіл, то її висота може бути більше необхідної, для зменшення висоти вічка слід використовувати замість нерозривного пробілу прозору картинку розміром в 1 піксель (картинка є в архіві до лабораторної роботи).
9. Додайте у файл **index.htm** посилання на виконані завдання з лабораторної роботи №3.
10. Покажіть виконану роботу викладачеві.

Контрольні питання

1. Який тег служить для вставки таблиці в html-документ?
2. Які атрибути тега **<table>** дозволяють змінити фон таблиці і ширину межі таблиці?
3. Чи задається ширина межі таблиці одна для всієї таблиці або задається окремо для зовнішньої межі і внутрішніх комірок?
4. Який тег відповідає за початок рядка?
5. Який тег відповідає за початок клітинки?
6. Як змінити вирівнювання тексту у клітинці?
7. Чи може бути в одному рядку клітинки з різним вертикальним вирівнюванням?
8. Якщо для таблиці встановити колір фону і фонове зображення одночасно, що відображуватиметься на екрані?
9. Як можна об'єднати клітинки по горизонталі?
10. Як можна в таблиці об'єднати клітинки по вертикалі?

Лабораторна робота № 4 Форми

Мета:
1) Навчитися створювати форми і вказувати обробників форм;
2) Навчитися створювати елементи керування форми;
3) Закріпити навики, отримані при створенні таблиць.

Теоретичні відомості

4.1.1 Форми. Загальні відомості

Форми призначені для організації взаємодії з користувачем. Вони дозволяють вводити текст, здійснювати вибір із запропонованих значень за допомогою списків або кнопок, організовувати інтерактивний обмін інформацією між Web-сторінкою і сервером.

Як правило, форма працює спільно зі встановленим на сервері сценарним забезпеченням, що оброблятиме введену інформацію. Механізм обробки форм представлений на рис.4.1.

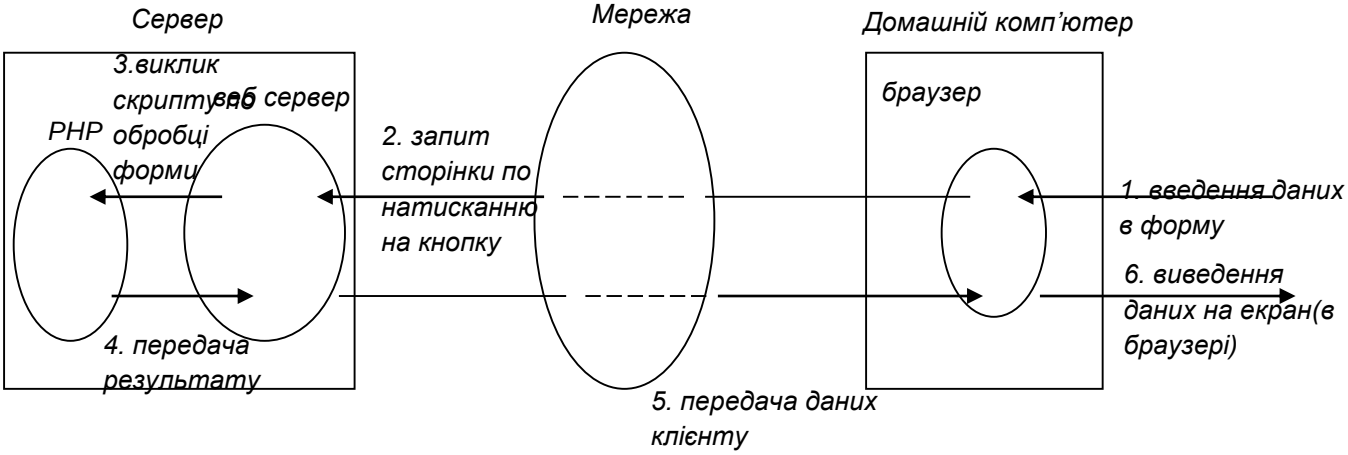


Рис.4.1. Обробка даних форми сервером

Найширше форми застосовуються для пошуку, реєстрації, заповнення анкет, тестів та ін. Форма визначається за допомогою тегів **<form>...</form>,** між якими розташовуються поля введення, кнопки, а також всі необхідні елементи оформлення форми. Для вказівки параметрів в тезі **<form>** використовуються атрибути (див. таблицю 4.1).

Таблиця 4.1

Атрибути тега <form>

Атрибут	Призначення
action	адреса програми або документа, які обробляють дані форми
method	Можливі варіанти значень методу: GET – спосіб передачі, коли дані, введені у форму додаються в рядок запиту та POST – дані пердаються у прихованому вигляді
name	ім'я форми. Використовується для звернення до форми за допомогою javscript
target	ім'я вікна або фрейма, куди обробник завантажуватиме повернений результат
enctype	тип інформації форми. Розрізняється в разі використання символьно-цифровий інформації і в разі використання файлів

4.1.2 Елементи керування форми

Сама форма служить лише обов'язковим контейнером для розміщення елементів керування. Зовнішній вигляд елементів керування залежить від встановлених значень атрибутів.

Поле введення

Однорядкове поле введення використовується, коли необхідно, щоб користувач ввів у форму дані в довільній формі але обмежені за об'ємом. Для отримання поля введення використовують тег **<input type="text">.**

Тег **<input>** є одним з різносторонніх елементів форми і дозволяє створювати різні елементи інтерфейсу і забезпечити взаємодію з користувачем. Головним чином **<input>** призначений для створення текстових полів, різних кнопок, перемикачів і прапорців.

Основний параметр тега **<input>,** що визначає вигляд елементу, – **type**. Він дозволяє задавати наступні елементи форми: текстове поле (**text**), поле з паролем (**password**), перемикач (**radio**), прапорець (**checkbox**), приховане поле (**hidden**), кнопка (**button**), кнопка для відправки форми (**submit**), кнопка для очищення форми (**reset**), поле для відправки файлу (**file**) і кнопка із зображенням (**image**). Для кожного елементу існує свій список параметрів, які визначають його вигляд і характеристики.

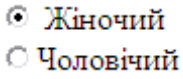
Окрім атрибуту **type** тег **<input>** має наступні атрибути (див. таблиця 4.2):

Таблиця 4.2

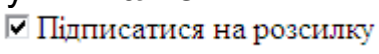
Атрибути тега <input>

Атрибут	Призначення
align	визначає вирівнювання, у випадку якщо **type="image"**
alt	визначає альтернативний текст для кнопки із зображенням (**type="image"**)
checked	заздалегідь активований перемикач або прапорець
disabled	блокує доступ і зміну елементу (поле недоступне)
maxlength	максимальна кількість символів, дозволених в тексті
name	ім'я поля, призначене для того, щоб відрізняти один елемент керування від іншого. Ім'я може бути вказане по всім правилам іменування змінних
readonly	дозволяє лише читання з текстового елементу керування
size	ширина текстового поля в символах
value	значення елементу. Значення змінної вказаної в атрибуті **name**

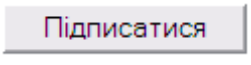
Перемикач



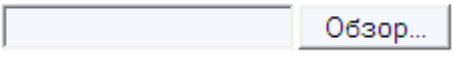
Перемикачі визначають поля вибору одного значення з декількох доступних, для кожної позиції перемикача створюється свій тег **<input type="radio">.** Групуються перемикачі за допомогою однакового імені, що задається атрибутом **name.**

Прапорець 

Прапорці використовуються, коли необхідно, щоб користувач вибрав один або декілька варіантів з обмеженого числа варіантів вибору. Прапорці у формі не залежать один від одного, їх можна встановити або скинути в будь-якій комбінації. Для кожного прапорця необхідно задати своє унікальне ім'я за допомогою атрибуту **name.** Створюються прапорці тегом **<input type="checkbox">.** Для встановки прапорця при завантаженні сторінки необхідно вказати атрибут **checked="checked".**

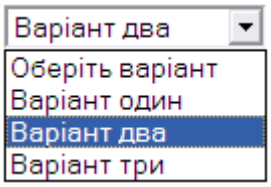
Командна кнопка 

Командна кнопка відправки **(type="submit")** використовується для виконання пересилки даних форми на сервер. Командна кнопка скидання **(type="reset")** повертає форму до вихідного достатку (очищає форму).

Поле вибору файлу 

Поле вибору файлу **(type="file")** створює поле для вибору файлу, який буде завантажений на сервер разом з інформацією форми. Поряд з полем введення відображується командна кнопка "Огляд...", що відкриває стандартне діалогове вікно вибору файлу. Якщо форма має на увазі завантаження файлів на сервер, то в атрибутах форми мають бути встановлені наступні атрибути **method="post"** і **enctype="multipart/form-data".**

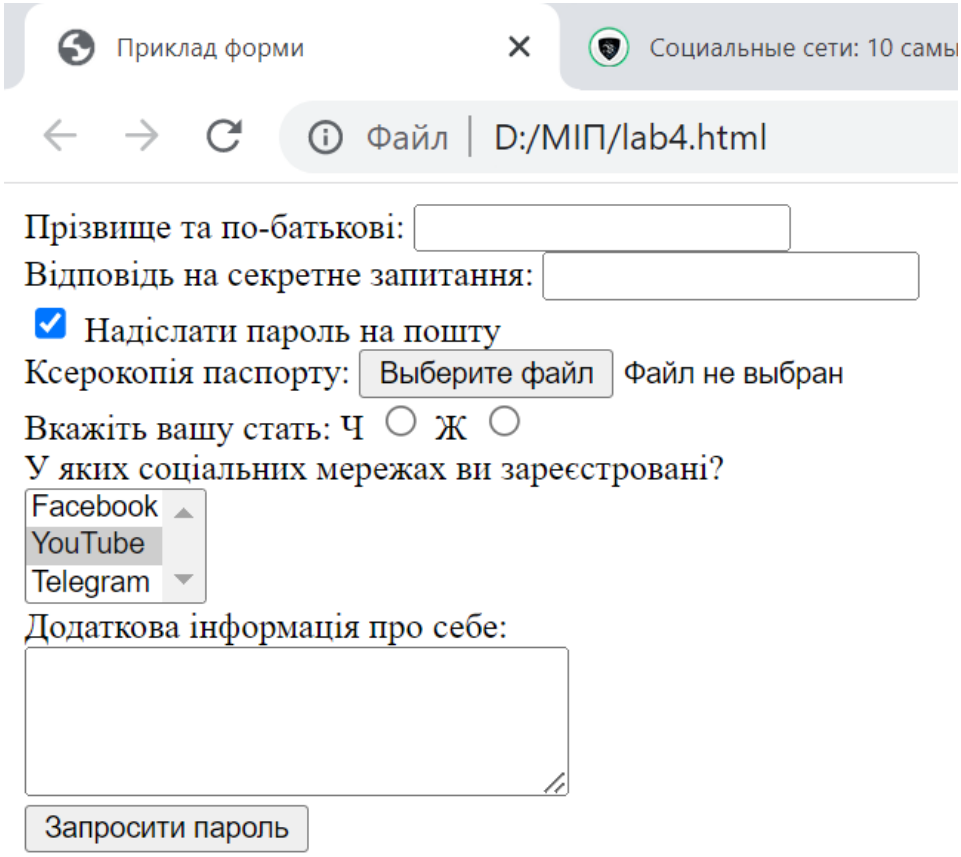
Списки вибору



Списки вибору бувають двох типів: списки, що розкриваються (випадні меню), і списки з множинним вибором. Незалежно від типів списків описуються вони однаково за допомогою пари тегів **<select> </select>.** Окремі елементи списку задаються з використанням тега **<option>.** Тип списку визначається за допомогою атрибуту **multiple** тега **<select>.**Приклад використання форми наведений в лістингу 4.1:

Лістинг 4.3 – Використання форми і елементів керування

```
<!DOCTYPE html>
<html>
<head>
    <title>Приклад форми</title>
       </head>
<body>
<body>
<form name="info" method="post" enctype="multipat/data">
Прізвище та по-батькові:
<input name="name" type="text" size="20" /> <br />
Відповідь на секретне запитання:
<input name="passwors" type="password" /> <br />
<input type="checkbox" name="zapros" checked="checked" /> Надіслати пароль на пошту<br />
Ксерокопія паспорту:
<input name="      " type="file" /> <br />
Вкажіть вашу стать: Ч <input type="radio" name="variant" />
Ж <input type="radio" name="variant" /><br />
У яких соціальних мережах ви зареєстровані?<br />
<select name="seti" size="3" multiple="multiple"> <option>Facebook</option>
<option>YouTube</option>
<option>Telegram</option>
</select><br />
Додаткова інформація про себе:<br />
<textarea name="about" rows="4" cols="30"></textarea><br />
<input type="submit" name="send" value="Запросити пароль"/>
</form>
```



Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог **C:\wamp64\www\lab_4** та файл **anketa.htm** і збережіть його в своїй папці на веб-сервері **C:\wamp64\www\lab_4.**
3. Створити форму для заповнення анкети (див. архів до лабораторної роботи). Зверніть увагу що сторінка з анкетою зверстана у вигляді таблиці з невидимою рамкою (див. рис.4.2).
4. Збережіть анкету у файлі **anketa.htm** на своєму домені. Зверніть увагу, що вся форма повинна розташовуватися у тезі **<form>**.
5. Створити поля для введення імені користувача і пароля користувача (символи вводяться в поле Пароль повинні відображуватися зірочками). Вставка полів здійснюється тегом **<input>** з різними значеннями атрибуту **type**.
6. Створити поля для введення дати народження. День і місяць народження вибирається з випадного списку, а рік вводиться в ручну в полі введення, причому кількість символів, що вводяться, не повинна перевищувати чотири. Для формування списку використовувати тег **<select>,** а для завдання обмеження на кількість символів, що вводяться, атрибут **maxlength** тега **<input>.**
7. Створити перемикачі для вибору статі. Реалізувати можливість вибору лише одного варіанту. Використовувати тег **<input>.**

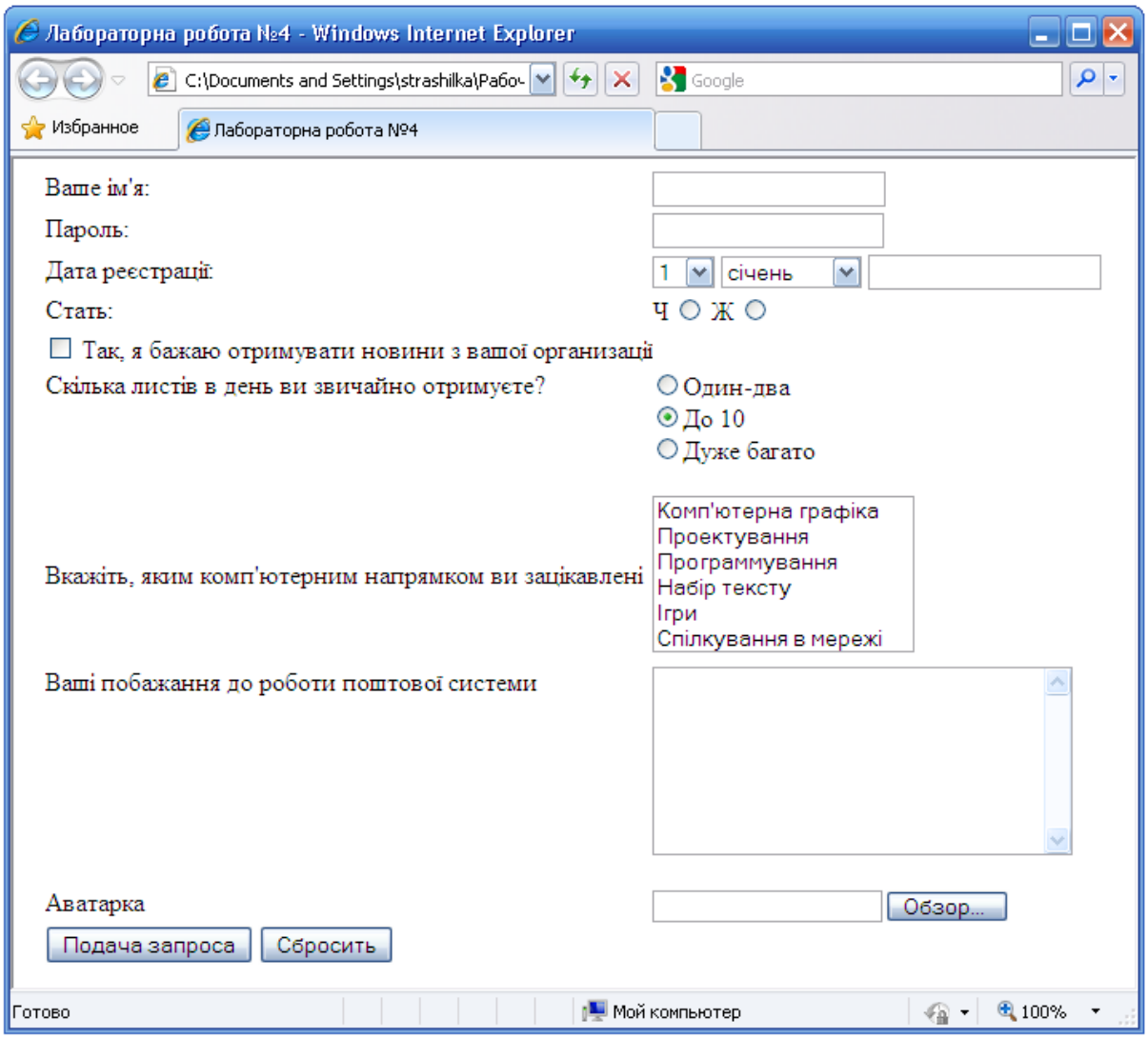


Рис.4.2. Форма заповнення анкети

8. Створити прапорець, з текстом «Так, я бажаю отримувати спам» з прапорцем, встановленим за умовчанням. Використовувати тег **<input>.**
9. Створити перемикачі для вибору кількості листів отримуваних в день, перевірити щоб перемикачі із завдання 6 і 8 формували дві різні групи.
10. Створити список інтересів з можливістю вибору декількох елементів.
11. Додати поле для введення декількох рядків тексту з додатковою інформацією про користувача. Поле повинне мати розміри 7 рядків і 30 стовпців. Для вставки такого поля використовуйте тег **<textarea>**.
12. Створити поле з кнопкою «Огляд» для завантаження файлу зображення. Використовувати тег **<input>**.
13. Створити дві кнопки. Першу для передачі даних форми на сервер, а другу для очищення форми. Використовувати тег **<input>**.

14. Додайте у файл **index.htm** посилання на виконані завдання з лабораторної роботи №4.
15. Покажіть виконану роботу викладачеві.

Контрольні питання

1. У якому тезі розташовані всі елементи керування?
2. Який атрибут тега **<form>** вказує на файл, який оброблятиме дані форми?
3. Які існують значення атрибуту **type** в тезі **<input>**?
4. У чому відзнаки тега **<input type="reset">** і **<input type="submit">**?
5. Як додати на форму перемикач?
6. Для чого призначений тег **<textarea>**?
7. Скільки тегів необхідно щоб вивести на екран поле введення для вказівки шляху до файлу і кнопки огляд?
8. Які існують методи передачі даних форми на сервер?
9. Як здійснюється обробка форм?
10. За рахунок чого перемикачі можна розбивати на групи?
11. Який атрибут тега **<input>** дозволяє вказати значення, яке буде заповнено при першому відображенні форми?
12. Для чого призначений атрибут **checked**?

Лабораторна робота № 5 Фрейми. Карти зображень

Мета:
1) Навчитися створювати фрейми;
2) Навчитися управляти властивостями фреймів;
3) Навчитися створювати і застосовувати карти зображень для непрямокутних посилань.

Теоретичні відомості

5.1.1 Фрейми

Фрейми дозволяють розбивати веб-сторінки на скроллюємі підвікна з метою поліпшення зовнішнього вигляду і функціональності інформаційних систем і веб-додатків. Кожен фрейм має наступні властивості:

– фрейм має свій URL, що дозволяє завантажувати його незалежно від інших фреймів;
– фрейм має власне ім'я (атрибут **name**), що дозволяє переходити до нього з іншого фрейма (атрибут **target** в тезі посилання **<a>**);
– розмір фрейма може бути змінений користувачем прямо на екрані за допомогою миші (якщо це не заборонено вказівкою спеціального параметра).

Ці властивості фреймів дозволяють створювати інтерфейсні рішення, які можуть поєднувати статичну інформацію в одному фреймі (це може бути зміст, графічний логотип фірми або набір кнопок, що управляють) і динамічну інформацію в іншому фреймі. Таким чином в одному фреймі знаходиться власне запит, а в іншому результати запиту.

Формат документа, використовуючого фрейми, зовні дуже нагадує формат звичайного документа, лише замість тега **<body>** використовується контейнер **<frameset>,** що містить опис внутрішніх html-документов, містить власне інформацію, що розміщується у фреймах. Контейнер **<frameset>,** у свою чергу складається з самих фреймів – тег **<frame>** і **<noframe>** дозволяє будувати подвійні документи для браузерів, підтримуючих фрейми і не підтримуючих фрейми.

Тег **<frameset>** має два взаємовиключні атрибути: **rows** (визначає число і висоту рядків в наборі фреймів) і **cols** (визначає число і ширину стовпців в наборі фреймів). Ширина (або висота) фрейма може задаватися як в пікселях, так і у відсотках від спільної ширини (або висоти) вікна браузера.

Наприклад:

<frameset cols=”50,*,50“> – описує три фрейма, два по 50 пікселів справа і зліва, та один усередині цих смужок.

<frameset rows=”20%,3*,*“> – описує три фрейма, перший з яких займає 20% площі зверху екрану, другий 3/4 що залишився від першого фрейма (тобто 60% всієї площі вікна), а останній 1/4 (тобто 20% всієї площі вікна).

Тег **<frame>** має свої атрибути, які детально описують вміст і поведінку кожного фрейма (див. таблиця 5.1):

Таблиця 5.1

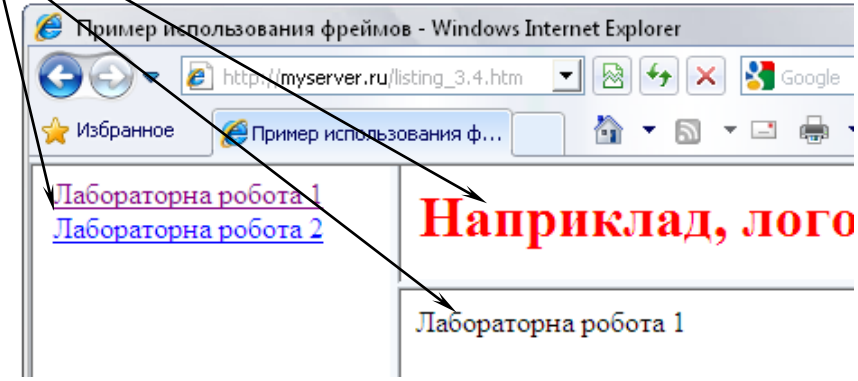
Атрибути тега <frame>

Атрибут	Призначення
src	задає шлях до файлу, призначеного для завантаження у фрейм
name	задає унікальне ім'я фрейма
scrolling	спосіб відображення смуги прокрутки у фреймі. Можливі варіанти значень: yes, no, auto
bordercolor	колір лінії межі
frameborder	визначає чи слід відображати рамку довкола фрейма
norsize	визначає чи можна змінювати розмір фрейма користувачеві чи ні

Приклад використання фреймів розглянутий в лістингу 5.1:

Лістинг 5.1 – Приклад використання фреймів

```
<html>
<head>
  <title> Приклад використання фреймів </title>
</head>
<noframes> Ваш браузер не підтримує фрейми </noframes>
<frameset cols="200,*">
  <frame src="menu.htm" frameborder="1"/>
  <frameset rows="30%,*">
    <frame src="head.htm" noresize="noresize"/>
    <frame src="content.htm" name="content" scrolling="yes"/>
  </frameset>
</frameset>
</html>
```



Коментарій до лістингу.

1. в документі відсутній тег **<body>;**
2. спочатку фреймова структура розбита на два стовпці: перший шириною в 200 пікселів, а другою займає простір, що залишився;
3. у лівому фреймі відкритий файл menu.htm і посилання цьому файлу організовані з вказівкою цільового фрейма, в якому повинні відкривати нові файли;
4. замість правого фрейма організована нова фреймова структура з двох рядків. Висота першого рядка 30% від спільної висоти вікна, а нижня частина займає ті 70%, що залишилися;
5. у верхньому файлі відкритий файл header.htm;
6. у нижньому файлі відкритий файл content.htm або той, який буде вибраний в лівому фреймі. Саме на ім'я тега **content** (див. лістинг 5.1) і по атрибуту **target** в тезі **<a>** файлу menu.htm (див. лістинг 5.2) визначається де має бути відкрите посилання.

Лістинг 5.2 – Атрибут target в тезі <a>

```
<html>
<head>
  <title></title>
</head>
<body>
  <a href="lab1.htm" target="content">Лабораторна робота 1</a><br />
  <a href="lab2.htm" target="content">Лабораторна робота 2</a>
</body>
</html>
```

5.1.2 Карти зображень

Карти-зображення дозволяють прив'язувати посилання до різних областей одного зображення. Реалізуються карти зображень в двох різних варіантах – серверному і клієнтському.

В разі використання серверного варіанту браузер посилає запит на сервер з вказівкою координат курсора миші над зображенням для здобуття адреси вибраного посилання і чекає відповіді з необхідною інформацією. Такий підхід вимагає додаткового часу на чекання результату і окремі файли для кожної карти-зображення.

У клієнтському варіанті карта розташовується в тому ж HTML-документі, що і посилання на зображення. Клієнтський варіант є більш частіше використовуваним. Для вказівки браузеру, що зображення є картою, використовується атрибут **usemap** в тезі **<img>.** Він є посиланням на опис конфігурації карти, яка задається тегом **<map>.** Значення атрибуту **name** даного тега повинне відповідати імені в **usemap**. Для завдання активної області, що є посиланням на HTML-документ, використовується тег **<area>** з атрибутами (див. таблицю 5.2):

Таблиця 5.2

Атрибути тега <area>

Атрибут	Призначення
shape	визначає форму активної області. Може набувати значень **circle**, **rect**, **poly**
alt	додає альтернативний текст підпису посилання для кожної області
cords	задає координати активної області. Координати відлічуються в пікселях від лівого верхнього кута зображення. Перше число є координатою по горизонталі, друге – по вертикалі.

Список координат переданих в атрибут **coords** залежить від форми області.

Для кола задаються три числа – координати центру кола і радіус: **<area shape="circle" coords="230,340, 100" href="circle.html">**

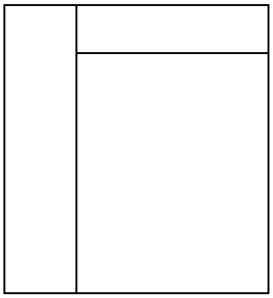
Для прямокутника – координати лівого верхнього і правого нижнього кута: **<area shape="rect" coords="24,18, 210,56" href="rect.html">**

Для полігону задаються координати його вершин.

В разі використання посилань непрямокутної форми альтернативою картам зображення може бути впровадження flash.

Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог **C:\wamp64\www\lab_5** та файл **frame.htm** і збережіть його в своїй папці на веб-сервері **C:\wamp64\www\lab_5.**
3. Створіть html-файл (frame.htm), що складається з трьох фреймів (див. рисунок справа). Для створення фрейма необхідно після розділу заголовка документа вказати структуру фреймів: два фрейми вертикальні, а в другому вертикальному ще два горизонтальні фрейми.



```
<frameset cols="20 *">
        <frame src="file.htm">
        <frameset rows="100 *">
   <frame src="file.htm">
   <frame src="file.htm">
</frameset>
</frameset>
```

4. Розташуйте файл на своєму веб-сервер і дайте кожному фрейму унікальне ім'я: значення атрибуту **name** для тега **<frame>.**
5. Задайте лівому фрейму фіксовану ширину в 234 пікселі із забороною змінювати ширину фрейма (атрибут **noresize**).

6. У лівий фрейм помістіть файл меню (**menu.htm**), який складатиметься із списку лабораторних робіт. Меню розмістіть в таблиці, що складається з трьох рядків (у кожен рядок фоном помістіть рисунок).

Лабораторна робота №1
Лабораторна робота №2
Лабораторна робота №3

7. Задайте для кожної лабораторної роботи посилання на відповідний документ, який буде відкритий в правому нижньому фреймі. Для відкриття посилання в іншому фреймі використовуйте атрибут **target** тега **<a>,** як значення атрибуту **target** вкажіть ім'я фрейма в якому необхідно відкрити файл.
8. У верхній фрейм помістіть, пошукову систему google.com. Значення атрибуту **src** для другого тега **<frame>.**
9. Створіть в папці свого домену файл **map.html**, що містить карту зображень.
10. Помістіть у файл map.html зображення-карту з геометричними фігурами:
    <img src="figure_1.png" border="0" usemap="#navigation">
11. Доповніть файл обробкою карти зображень

```
<map name="navigation">
<area shape="circle" coords="72,93,40"
href="krug.html" title="Коло">
<area shape="rect" coords="129,129,268,194"
href="pryam.html" title="Прямокутник">
<area shape="poly" coords="168,108,221,15,279,108"
href="triangle.html" title="Трикутник">
</map>
```

12. Перевірте роботу карти зображень. Зверніть увагу на спливаючі підказки і посилання.
13. Створіть в окремому файлі аналогічну карту зображень для завдання областей України. У карті зображень виділіть три області, на київську область призначте посилання на сайт **meta.ua** на дві інші області будь-які посилання. Щоб узнати координати точок зображення можна використовувати програму Paint .
14. Додайте у файл, що знаходиться в лівому фреймі посилання на обидві карти зображень.
15. Додайте у файл **index.htm** посилання на виконані завдання з лабораторної роботи № 5.
16. Покажіть виконану роботу викладачеві.

Контрольні питання

1. Для чого використовуються фрейми?
2. За що відповідає атрибут **target** тега **<a>**?
3. Як вказати посиланню в якому фреймі треба відкритися?
4. У яких одиницях можна задати ширину або висоту фрейма?
5. Чи можна змінювати межі між фреймами?
6. Для чого використовуються карти зображень?
7. Які форми фігур можуть використовуватися в картах зображень?
8. Як можна підписати фігури карти зображень?
9. Які види карт зображень бувають?
10. Що може служити альтернативою використання карт зображень?

Лабораторна робота № 6 Введення в каскадні таблиці стилів (CSS).

Мета: 1) Вивчити підключення таблиць стилів до документа;
2) Навчитися змінювати css-файли

Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог **C:\wamp64\www\lab_6**

3. Запустіть Visual Studio Code чи Notepad++ створіть в ньому два файли: **lab.html** та **lab6style.css**. Обидва файли збережіть на своєму домені **C:\wamp64\www\lab_6**.

4. Створіть в html файлі п'ять абзаців довжиною в три рядки.
5. Збережіть html сторінку та проглянете її на своєму домені.
6. В css файлі напишіть стиль для селектору тегу body, в якому вкажіть кольори тексту, розмір букв і назву шрифту:

```
body{
  color: #009933;
  font-size: 14px;
  font-family: Arial;
}
```

7. Включіть попередній перегляд і подивіться як виглядає стил усього документа. Для того, щоб html документ «підняв» ці стилі необхідно в html-файлі підключити css файл:

```
<head>
  <title></title>
  <link rel="stylesheet" type="text/css" href="lab6style.css"/>
</head>
```

8. Збережіть зміни. Подивіться html-файл у браузері. Зверніть увагу на шрифт.
9. В css файлі створіть шрифт абзацу, у якому встановіть наступний інтервал тексту між буквами:

```
p{
  letter-spacing: 10px ;
}
```

10. Збережіть css файл. Оновіть html файл у браузері. Використовуючи доповнення **FireBug** перегляньте які стилі використовуються в другому абзаці.
11. Для того, щоб один з абзаців зробити курсивом та підкреслити, необхідно створити наступний клас

```
.kursiv{
  font-style: italic;
  text-decoration: underline;
}
```

17. Підключіть даний клас до другого абзацу

```
<body>
  <p>Текст абзаца. Текст абзаца. Текст абзаца. Текст абзаца.
  <p  class="kursiv">Текст абзаца. Текст абзаца. Текст абзац
  <p>Текст абзаца. Текст абзаца. Текст абзаца. Текст абзаца.
  <p>Текст абзаца. Текст абзаца. Текст абзаца. Текст абзаца.
```

12. Збережіть зміни та перевірте результат.
13. Створіть клас для форматування третього абзацу з наступними властивостями:
    - шрифт абзацу Comic Sans; (font-family)
    - розмір шрифту 25 пт; (font-size)
    - відступ нового рядка в 50 пикселів (text-indent).
14. Підключіть стиль до третього абзацу (використовуючи атрибут class).
15. Створіть класс .rich для відображення п'ятого абзацу, що містить наступні властивості тексту:
    - Шрифт Arial, 15 пт;
    - Накреслення жирне + курсив (font-weight; font-style);
    - Колір букв жовтогарячий (color);
    - Висота рядка 25 пунктів (line-height).
16. Створіть клас perenos і забороніть в ньому перенесення слів в межах одного абзацу (white-space). Призначте цю властивість другому та четвертому абзацу. Для цього в атрибуті class тегу абзацу через пробіл вкажіть два класи (той що був раніше + клас perenos). Збережіть зміни.
17. Для того що б виділити слова в межах одного абзацу можна використати тег (він дозволяє додавати форматування до будь-якого текстового елемента). Створюємо клас, що містить заголовні червоні букви:

```
.red{
color: #FF0000;
text-transform: uppercase;
}
```

і вказуємо новий клас у тегу

<p c1ass="kuriv perenos">Текст абзацу.кзрап c1ass="red"> Увага! Текст абзацу. Текст абзацу.

18. Створіть клас з наступними властивостями:
    - Шрифт Courier 12 пт;
    - всі букви заголовні;

- міжбуквений інтервал виряджений на 200%;
- текст підкреслений;
- міжрядковий інтервал подвійний.

19. Додайте ще пару абзаців і призначте їм властивості цього класу.
20. Створіть свої 2-3 класи стилів тексту, які ви будете використовувати в практичній роботі (використовуючи властивості таблиці 6.1). Кожен стиль повинен включати по 3-4 властивості форматування тексту і бути підключеним до абзаців або до заголовків першого, другого та третього рівнів.
21. Додайте у файл **index.htm** посилання на виконані завдання з лабораторної роботи № 6.

Таблиця 6.1

Властивості тексту

№	Властивість (можливі значення)	Призначення властивості
1.	font-family (Arial | Times New Roman | Courier | Tahoma)	назва шрифту
2.	font-weight (normal | bold | lighter)	насиченість шрифту (ширина букв)
3.	font-size (small | large | medium | 120% | 14px)	розмір шрифту
4.	font-style (normal | italic)	стиль шрифту (звичайно курсив)
5.	font-stretch (normal | width | condensed | expanded)	розтягнуте накреслення шрифту
6.	text-indent (число px | %)	відступ першого рядка, при негативному значенні виступ
7.	text-align (left | center | right | justify)	горизонтальне вирівнювання
8.	line-height (normal | …px | …% | …em)	висота рядка (міжрядковий інтервал)
9.	vertical-align (sub | sup | top | middle | bottom)	вертикальне вирівнювання тексту в контейнері
10.	word-spacing (…px|normal)	відстань між словами
11.	letter-spacing (…px|normal)	відстань між буквами
12.	text-transform (uppercase | lowercase | capitalize | none)	регістр букв
13.	text-decoration (none | underline | overline | line-through)	оформлення тексту

14.	text-shadow (кольори довжин довжина довжина)	тінь до тексту
15.	white-space (normal | nowrap | pre)	обробка пробілів
16.	color	кольори

Контрольні питання

1. Для чого використовують доповнення **Firebug**?
2. Наведіть синтаксис правила в CSS?
3. Назвіть способи застосування таблиці стилів до HTML документа?
4. Наведіть властивості тексту?
5. Наведіть властивості шрифту?

Лабораторна робота № 7 Властивості блокових об'єктів.

Мета: 1) Вивчити можливості CSS по роботі з блоковими об'єктами;
2) Вивчити основні властивості блоків;
3) Навчитися застосовувати теги ** ** та **.**

Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог **C:\wamp64\www\lab_7**

1. Створіть у каталозі html-файл lab7.htm, що містить блочний елемент DIV.

```
<body>
<div>Це блоковий елемент</div>
</body>
```

2. Створіть css-файл і підключіть його до html-документа.

3. Створіть клас k1, в якому визначте розмір блоку 200x200 пикселів, тло (ясно-зелений колір) і рамку (суцільна, темно-зелена, шириною в 3 пикселя).

```
.k1{
  width:200px;
  height: 200px;
  background-color: #99FF99;
  border-style: solid;
  border-color: #009933;
  border-width: 3px;
}
```

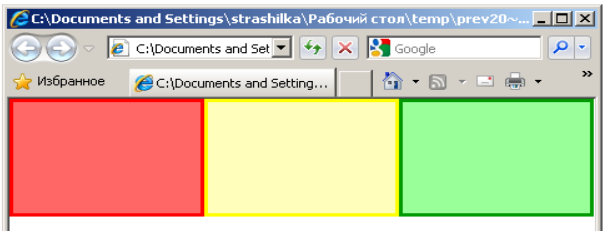
4. Підключіть клас k1 до блоку в html-документі. Зверніть увагу на те, як зміниться блок.

5. Додайте ще п'ять таких самих блоків розміром 200x200 пикселів. Блоки розташовуються один під одним.

6. Для того, щоб блоки розташовувалися один за одним по горизонталі необхідно додати в клас k1 оголошення `float: left;` . Оновіть і подивиться на зміни при зміні розмірів вікна.

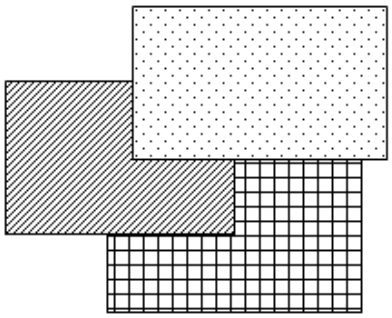
7. Тепер блоки «липнуть» один до одного, між ними немає відстані, для того щоб додати зовнішній відступ ліворуч додаємо оголошення `margin-left: 15px;` . Збережіть, перегляньте зміни. У браузері Mozilla виділіть будь-який **div** (за допомогою **FireBug**) і перейдіть на вкладку **Layout**. Поекспериментуйте із властивостями: **padding, margin, border** та **offset**. Зверніть увагу, що при зміні всіх цих властивостей, вони дописуються в html-код.

8. Створіть в окремому файлі горизонтальний світлофор із трьох блоків. Ширину блоків зробіть динамічної, що б вони змінювалася залежно від ширини вікна, але всі блоки повинні мати одну ширину. Задайте кожному блоку свої кольори. Відступи для **body** заберіть за допомогою css.



9. Задайте першому блоку оголошення **display:none**, а іншому **visibility: hidden;** В чому відмінності цих двох оголошень?

10. Створіть файл, а в ньому три блоки. У першому блоці зробіть рамку з різним типом лінії для кожної сторони блоку. В другому блоці задайте фонове зображення (**background-image**). В третьому блоці, задайте тло…Додавайте рядки по одному та дивіться на зміни в браузері після кожного рядка.



```
.blok{
  height:400px;
  background-image: url(fon.jpg);
  background-repeat: repeat-x;
  background-position: bottom;
  background-color: #FEFE00;
}
```

11. Розташуйте блоки один над одним. Для цього встановіть кожному блоку оголошення **position: absolute;** і порядок розташування блоків зверху донизу: оголошення **z-index: 5;** (у тих блоків що нижче, значення індексу повинне бути меншим, наприклад 10, 20 та 30). Для того, щоб блоки починалися не в одній точці, їм необхідно задати властивості **left** і **top**.

12. Створимо округлені кути. Для цього створюємо чотири вкладених один в одний блоки.

```
<body>
<div><div><div><div>Это блок со скругленными углами</div></div></div></div>
</body>
```

опишемо для зовнішнього блоку клас ugol і призначимо цей клас блоку

```
.ugol {
  background: url(panel_top_left.gif);
  background-position:  top left;
  background-repeat: no-repeat;
  background-color: #74C440 ;
}
```

опишим клас для правого верхнього кута

```
.ugol div {
    background: url(panel_top_right.gif) top right no-repeat;
}
```

і самостійно для лівого і правого нижнього кутів.

13. Створити новий html файл у який помістити зображення ягоди.
14. В css-файл підключений до html-файлу запишіть стиль для зміни прозорості будь-якого зображення при наведенні на нього курсору миші.

```
img{opacity: 0.3;}
img:hover{opacity: 1; cursor:pointer;}
```

Для Internet Explorer рядки повинні бути наступними:

```
img{opacity: 0.3;filter: alpha(opacity=30);}
img:hover{opacity: 1; cursor:pointer; filter: alpha(opacity=100);}
```

15. Використовуючи тільки блоки, створіть поле для гри в хрестики-нуліки.
16. Використовуючи блоки, їхню вкладеність, тло, вирівнювання, властивість **float**, шрифт **Georgia** створіть наступну шапку. Кути виріжте самостійно.

17. Додайте у файл **index.htm** посилання на виконані завдання з лабораторної роботи № 7.

Контрольні питання

1. Як розташувати блоки один під одним, та по горизонталі?
2. Як розташувати блоки один над одним?
3. Яким чином регулюється порядок розташування блоків зверху донизу?
4. Пропишіть стиль для зміни прозорості будь-якого зображення при наведенні на нього курсору миші?

Лабораторна робота № 8 Додаткові властивості CSS

Мета: 1) Детально вивчити структуру таблиці для прискорення створення стилів CSS;

2) Вивчити основні можливості роботи зі списками за допомогою CSS.

Порядок виконання роботи

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог C:\wamp64\www\lab_8
3. Створіть html- файл lab_8.html і вставте у всередину тега <body> текст

```
<table>
  <caption>Безоплатні ресурси для програмування</caption>
  <tr>
    <th>Сайти</th>
    <th>Ютуб канали</th>
    <th>Мобільні додатки</th>
  </tr>
  <tr>
    <td>Freecode Camp</td>
    <td>Freecode Camp</td>
    <td>Enki</td>
  </tr>
  <tr>
    <td>W3Schools</td>
    <td>Academind</td>
    <td>Programming Hero</td>
  </tr>
  <tr>
    <td>Khan Academy</td>
    <td>The Coding Train</td>
    <td>Solo learn</td>
  </tr>
</table>
```

4. Прогляньте і проаналізуйте структуру таблиці. Зверніть увагу на те, які теги можна використати в таблиці.
5. Створіть css- файл в якому вкажіть те, що в таблиці та осередках (table, td, th{.}) буде використовуватися рамка червоного кольору, шириною в 1 піксель. Під'єднайте цей css- файл до html- файлу.
6. Зверніть увагу, що рамка в таблиці виходить подвійна. Що б уникнути подвійної рамки, необхідно додати новий стиль.

```
*{
        border-collapse: collapse;
        }
```

7. Призначте таблиці окремий клас. Опишіть властивості окремих елементів таблиці використовуючи наступні стилі:

```
.frmtbl { /* завдання стилю таблиці загалом */ }
.frmtbl thead { /* завдання стилю для заголовка таблиці */ }
.frmtbl tbody { /* завдання стилю для основної частини таблиці */ }
.frmtbl tfoot { /* завдання стилю для нижньої частини таблиці */ }
```

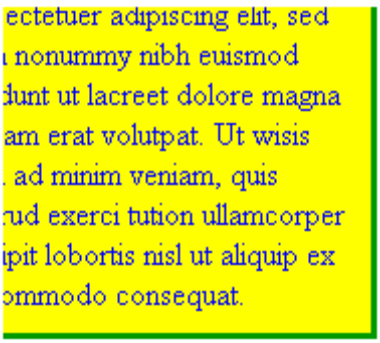
8. Використовуючи властивості **list - style - type**, **list - style - position**, **list - style - image** розмістіть у блоці маркірований список. В якості маркера використайте зображення, маркер має бути всередині списку. Відступ від лівого краю блоку до списку повинен складати 50 пікселів.
9. Створіть новий документ і розмістіть в ньому блок з ідентифікатором **layer.**

```
<div id="layer">
  Lorem ipsum dolor sit amet, consectetuer
  nonummy nibh euismod tincidunt ut lacree
  Ut wisis enim ad minim veniam, quis nost
  lobortis nisl ut aliquip ex ea commodo c
</div>
```

10. Опишіть ідентифікатор **layer**, який міститиме: жовтий фон, зелену рамочку, відступ від тексту до краю блоку до тексту, сині букви, ширину в 200 пікселів, абсолютне позиціонування і область позиціонування елементу.

clip: rect(40px, auto, auto, 40px);

Якщо все зробили правильно, то на екрані повинна з'явитися тільки кадрована частина тексту.



11. Створіть блок з рамкою розміром 200x200 пікселів. Помістіть в цей блок зображення свідомо більшого розміру. Прогляньте в браузері. Поекспериментуйте з властивістю **overflow** таким чином, що у тому випадку, якщо у блок не вміщується вміст, з'являлися смуги прокрутки (чи тільки одна)?
12. В браузері Internet Explorer є можливість змінити колір смуг прокруток додавши наступні стилі. Наприклад:

```
html, body {
   scrollbar-3dlight-color:red;
   scrollbar-arrow-color:yellow;
   scrollbar-highlight-color: aqua;
   scrollbar-face-color:green;
   scrollbar-shadow-color:fuchsia;
   scrollbar-darkshadow-color:blue;
   scrollbar-track-color: maroon;
}
```

13. Додайте у каталог файл index.htm та додайте в цей файл посилання на виконані завдання з лабораторної роботи № 8.

Контрольні питання

1. Як призначити таблиці окремий клас, опишіть властивості окремих елементів таблиці?
2. Як отримати кадровану частину тексту?
3. Опишіть властивість **overflow**?

Лабораторна робота № 9 Приклади верстання веб-сторінок

Мета: 1) Отримання практичних навичок верстання веб-сторінок.

Порядок виконання роботи

На даному лабораторному занятті необхідно створити веб-сайт, що має наступний вигляд (див. додаток Д) згідно варіанту. Для реалізації даного завдання необхідно виконати наступну послідовність дій.

1. Запустіть веб-сервер Wampserver.
2. Створіть каталог **C:\wamp64\www\lab_9**
3. Скопіюйте в цю папку картинки з **додатку Г** згідно варіанту.
4. Створіть в програмі Visual Studio Code чи Notepad++ html-файл і збережіть його з ім'ям lab9.htm в папці **C:\wamp64\www\lab_9**\.
5. Створіть в програмі Visual Studio Code чи Notepad++ css-файл і збережіть його з ім'ям styles_9.css в папці **C:\wamp64\www\lab_9**\.
6. Зв'яжіть html файл з css файлом. В тезі <head> пропишіть

```
<link type="text/css" rel="stylesheet" media="all" href="styles_9.css">
```

7. Створіть заголовок web-сторінці "Лабораторная работа 9".

```
<title>Лабораторна робота 9</title>
```

8. Опишіть в css файлі параметри всього html документу

```
body {
  background-color:#DFF5FF;
  font-size:13px;
  margin:0;
}
```

9. Збережіть обидва файли (css і html) і прогляньте сторінку у браузері, вона має бути ніжно блакитного кольору і мати заголовок вікна.
10. Всередині тега <body> створіть таблицю з наступними параметрами: 1 стовпець, 4 рядки, ширина таблиці 100%, рамка 0 пікселів (скористайтеся для цього кнопкою таблиця на панелі інструментів) :

```
<table width="100%" cellpadding="0" cellspacing="0" border="0">
 …
```

</table>

11. Перед початком кожного рядка додайте коментар з інформацією про номер рядка

    <!--Перша строка-->.

Перший рядок головної таблиці призначений для шапки

12. Всередині першого рядка (всередині тега <td>) розташуйте таблицю, що складається з одного рядка і двох комірок всередині даного рядка.

```
<table width="100%" cellspacing="0" cellpadding="0" border="0">
…
</table>
```

13. В css файлі створіть клас для лівої комірки (файл css) і призначте його першій комірці <td class="top_left"> (файл html)

```
.top_left{
 text-align: left;
 vertical-align: top;
 background-image: url(images/bg_top.gif);
 height: 318px;
}
```

14. В css файлі створіть клас для правої комірки і призначте його першій комірці <td class="top_right">

```
.top_right{
 width: 713px;
 background-image: url(images/big_picture.jpg);
}
```

15. Розташуйте в першій (лівій) комірці логотип і посилання на ваш сайт (для цього використайте вставку зображення, а посилання в ньому можна буде вказати)

```
<a href="http://my.net">
<img src="images/logo.jpg" width="277" height="318" alt="" /></a>
```

16. Для того, щоб посилання не мали синьої контурної лінії добавте в зображення (тег <img>) атрибут border="0".
17. Подивіться на результат (верхня частина сторінки повинна цілком відповідати рис.9.1).

Другий рядок головної таблиці (призначений для розділення шапки і меню)

18. Для комірки другого рядка головної таблиці створіть і підключіть клас separator.

19. Клас повинен вміщувати наступні властивості: висота 19 пікселів, ширина 100 відсотків, фон зображення su_bg.gif.
20. Збережіть, оновіть, подивіться на результат.

Третій рядок головної таблиці призначений для верхнього горизонтального меню

21. Створіть для комірки з рядком меню клас menu_line з наступними параметрами: висота 32 пікселя, внутрішній відступ 60 пікселів, фонове зображення menu_bg.gif.
22. Призначте комірці третього рядка класс menu_line.
23. Створіть всередині комірки третього рядка блоковий об'єкт, в якій розташуйте зображення і посилання (створіть 5-6 копій блоків)

```
<div class="menu_item">
   <a href="#">перше посилання</a>
</div>
```

24. Опишіть клас menu_item в якому необхідно врахувати: зображення тла (трикутник menu_list.gif), скасувати повторення зображення тла, вирівняти тло по левому краю, внутрішній відступ ліворуч 15 пікселів, а зовнішній відступ праворуч 40 пікселів, блоки вкладати в ряд таким чином, щоб вони своїм лівим боком приєднувалися до попереднього блоку.

25. Збережіть, оновіть, подивіться на результат.
26. Опишіть клас ".menu_item a", який буде описувати стиль посилань, що розміщуються всередині тега, що має клас menu_item. В класі повинні бути налаштовані наступні параметри: шрифт - Verdana не жирний, текст - не підкреслений, колір посилань #B7F7FF, розмір шрифта 13 пікселів.
27. Створіть клас, який буде при наведенні на посилання змінювати його стан на підкреслене.

Четвертий рядок головної таблиці призначений для лівого меню і контентної частини документу

28. Створіть для комірки четвертого рядка клас content_td, який буде вміщувати тло, що повторюється по осі X (з файлу bottom.jpg).

В цу комірку розташуйте таблиці і призначте коміркам класи:

<table>
<tr>
<td>class="content_left"

<table>
<tr><td>class="menu_table_top"</td></tr>
<tr><td>class="menu_left"</td></tr>
<tr><td>class="menu_table_bottom"</td></tr>
</table>
</td>
<td>class="content_right"</td>
</tr>
</table>

29. Опишіть кожен клас. Для класа content_left

```
.content_left{
background-image:url(images/palmOS.jpg);
background-position:left bottom;
background-repeat:no-repeat;
padding:8px 8px 290px;
width: 220px;
vertical-align: top;}
```

30. Помістіть всередину комірку з класом menu_left наступний фрагмент сторінки

```
Перший рівень<br />
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
Другий рівень<br/>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
Третій рівень<br />
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
<a href="#">Посилання</a>
```

31. Для класів menu_table_top, menu_left, menu_table_bottom опишіть стили так.

menu_table_top: висота, ширина, тіло;
menu_left: тіло, відступи від границі до тексту, колір тексту, подкреслення тексту, жирність тексту, назва шрифту, розмір шрифту;
menu_table_bottom: висота, ширина, тіло.

32. Створіть стиль для посилань всередині комірки з класом menu_left. В назві стиля необхідно вказати назву класа і назву тега. Для цього стиля опишіть властивості згідно з вашим варіантом.
33. Розташуйте в праву контентну частину фрагмент кода з текстового файла (з архіву). Створіть для неї стилі згідно з вашим варіантом.

Лабораторна робота №10 Уведення в JavaScript

Мета:
4) Знайомство з мовою розробки клієнтських веб – сценаріїв Javascript.
5) Вивчення основ мови і його застосування для автоматизації процесу розмітки й додавання інтерактивних можливостей веб – сторінок.

Теоретичні відомості

10.1 Елементи мови Javascript

Javascript дозволяє "пожвавити" веб-сторінку. Це реалізується шляхом додавання до статичного опису фрагмент коду, що виконується. Javascript-сценарій може взаємодіяти з будь-якими компонентами Html-Документа й реагувати на зміну їх стану.

Javascript не є строго типізованим мовою, у змінні можуть зберігатися практично будь-які типи даних.

Як і програма мовою Java, сценарій Javascript виконується під управлінням інтерпретатора. Інтерпретатор, як правило є вбудованим у браузер. В налаштуваннях браузера є можливість заборонити або дозволити виконання Javascript–сценаріїв на сторінці.

10.2.Структура сценарію

Сценарієм Javascript уважається фрагмент коду, розташований між дескрипторами <SCRIPT> і </SCRIPT>:

```
Текст Html-Документа
<SCRIPT>
Код сценарію
</SCRIPT>
Текст Html-Документа
```

Сценарій Javascript може бути у вигляді окремого файлу з розширенням **js**, на який треба із сторінки зробити посилання.

```
<html>
 <head>
       <script type="text/javascript" src="xxx.js"></script>
 </head>
```

```
<body>
</body>
</html>
```

Зауваження: Пам’ятайте, що розташовувати посилання на зовнішній файл скриптів потрібно там , де писали б ці скрипти, не використовуючи зовнішній файл! Javascript чутливий до регістру.

Скрипти можливо розташовувати у секціях <head>, <body>. Кількість скриптів на сторінці може бути в будь якій кількості.

10.3.Змінні

У сценаріях Javascript змінні можуть зберігати дані будь-яких типів: числа, рядок тексту, логічні значення, посилання на об'єкти, а також спеціальні величини, наприклад "нульове" значення *null* або значення *NaN*, яке повідомляє про неприпустимість операції.

Змінна в мові Javascript оголошується за допомогою ключового слова **var**. Так, наприклад, вираження

```
var selected = "first item";
```

створює змінну з іменем select і привласнює їй у якості значення рядок символів "*first item*". Змінні можуть оголошуватися також автоматично. Це відбувається при присвоєнні значення змінних, що не зустрічалися раніше в даному сценарії. Так, у наступному прикладі створюється змінна з іменем *rating*, якій привласнюється числове значення, рівне *512.5*:

```
rating = 512.5;
```

Кожний рядок (оператор) закінчується символом **;**

10.4.Об'єкти

У мові Javascript не передбачені засоби для роботи із класами в тому виді, у якому вони реалізовані в C++ або Java. Розроблювач сценарію не може створити підклас на основі існуючого класу, перевизначити метод або виконати яку-небудь іншу операцію із класом. Сценарію, написаному мовою Javascript, в основному доступні лише готові об'єкти. Побудова нового об'єкта доводиться виконувати лише в рідких випадках.

Об'єкти містять *властивості* (властивості об'єктів можна зрівняти зі змінними) і *методи*. Об'єкти, а також їх властивості й методи ідентифікуються *іменами*. Об'єктами є форми, зображення, гіпертекстові посилання й інші компоненти веб-сторінки, Html-Документ, відображуваний у вікні браузера, вікно браузера й навіть сам браузер. У процесі роботи Javascript – сценарій звертається до цих об'єктів, одержує інформацію й управляє ними.

Крім того, розроблювачеві сценарію мовою Javascript доступні об'єкти, не зв'язані безпосередньо з Html-Документом. Їх називають *визначеними,* або *незалежними* об'єктами. За допомогою цих об'єктів можна реалізувати масив, описати дату й час, виконати математичні обчислення й розв'язати деякі інші завдання.

Перший об'єкт, з яким необхідно познайомитися, щоб написати найпростіший сценарій, - це об'єкт *document*, який описує HTML документ, відображуваний у вікні браузера. Нижче наведений вихідний текст веб-сторінки, що містить сценарій, дії якого зводяться до виводу рядка тексту у вікні браузера.

Приклад 1.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript</title>
</head>
<body>
   <h2>Перша програма на JavaScript</h2>
   <script>
document.writeln ("<H1> Мова програмування JavaScript!</H1>");
   alert('Привіт світ!');
   </script>
</body>
</html>
```

Імена чутливі до регістрів символів, і якщо ви спробуєте звернутися до поточного документа з великої літери, інтерпретатор Javascript відобразить повідомлення про помилку.

Основне призначення сценаріїв Javascript - створювати динамічно мінливі об'єкти, коректувати вміст Html-документів залежно від особливостей оточення, здійснювати взаємодія з користувачем і т.ї.

10.5 Операції

Набір операторів в Javascript, їхнє призначення й правила використання в основному збігаються із прийнятими в мові C++. Виключенням є операція, що задається символом "+". В Javascript символ "+" визначає як *підсумовування* числових значень, так і *конкатенацію* (об'єднання) рядків.

Так, наприклад, у результаті обчислення вираження

```
sum = 47 + 21;
```

змінної *sum* буде привласнене значення *68*, а після виконання операції

```
sum = "рядок 1 " + "рядок 2";
```

у змінну *sum* буде записана послідовність символів "*рядок 1 рядок 2*".

Розглянемо ще один приклад:

Приклад 2.

Перший рядок document.write відображає результат підсумовування двох числових значень, друга й третя - результат конкатенації рядка й символьного значення числа. Якщо операція підсумовування чисел передує конкатенації, Javascript обчислює суму чисел, представляє її в символьному виді, потім робить конкатенацію двох рядків. Якщо ж першої у вираженні зазначена операція конкатенації, то Javascript спочатку перетворить числові значення в символьний вид, а потім виконує конкатенацію рядків.

```
<HTML>
 <BODY>
 <h2>Числа й рядки </h2><br>
 <script>
    var per = prompt('Скільки вам папуг?', 18);
    var a = 3; var b = 8; var c = " папуг ";
 document.write("a+b="); document.write(a + b); document.write("<br>");
 document.write( "a +c = "); document.write(a+c); document.write("<br>");
 document.write("c + a = "); document.write(c + a); document.write("<br>");
 document.write("a + b + c = "); document.write(a + b + c); document.write("<br>");
 document.write("c + a + b = "); document.write(c + a + b); document.write("<br>");
 document.write("Кількість папуг= "); document.write(per); document.write("<br>");
function perrot(p) {
  p = per;
      document.write("Кількість папуг= "); document.write(p); document.write("<br>");
} ;
</script>
<INPUT TYPE="button" value="Функція" onClick="perrot()">
</BODY>
 </HTML>
```

10.6 Керуючі конструкції

Керуючі конструкції, використовувані в мові C++, в основному застосовні й у сценаріях Javascript. В Javascript додатково визначені язикові конструкції, відсутні в C++, а саме: оператори *for…in* і *with*. У прикладі 3., за допомогою оператора циклу, на веб – сторінці формується таблиця множення чисел.

Приклад 3.

```
<html>
<body>
<table>

 <script>
  document.write("<tr><td>finbsp;<td>");
  for (i = 1; i < 10; i++) document.write("<td>"+i+"finbsp;<ftd>");
      d ocum ent .write (" </tr>");
   for (i = l;i < 10; i++)
{
document .write("<tr><td>" + i + '&enbsp;<Vtd>"); forG = l;j<10;j++)
{
document.write("<td bgcoloror=#00ffa0'>" + "cfcnbsp;<td>");      }
document.write("</tr>");
          }
 </script>
<table>
</body>
</html>
```

Окремої уваги заслуговує оператор *new*. Незважаючи на те, що більшість об'єктів уже створена браузером і доступні сценарію, у деяких випадках доводиться створювати об'єкти в процесі роботи. Це ставиться до визначених об'єктів і об'єктам, обумовленим розроблювачем сценарію. Для створення об'єкта використовується оператор *new*, який викликається в такий спосіб:

*змінна = **new** тип_ об'єкта (параметри)*

10.7 Функції

Формат оголошення функції виглядає в такий спосіб:

***function** ім'я функції ([параметри]) тіло функції*

Оголошення функції починається із ключового слова **function**. Так само, як і в мові C++ для ідентифікації функції використовується ім'я, при виклику функції можуть передаватися параметри, а по закінченню виконання вертатися значення. Однак, на відміну від C++, тип значення,

що вертається, і типи параметрів не задаються. Нижче показано два способи виклику функції

- *ім'я_функції ([параметри]);*
- *змінна = ім'я функції ([параметри]);*

У другому випадку значення, що вертається функцією, привласнюється зазначеної змінної.

10.8 Область видимості змінних

Робота зі змінними в тілі функції підкоряється наступним правилам.

- Якщо змінна оголошена за допомогою ключового слова *var*, доступ до неї здійснюється за правилами, подібним тим, які використовуються в мові *C.*
- Змінна, оголошена усередині функції, уважається *локальною*. Область видимості такий змінної обмежується тілом функції, у якій вона оголошена.
- Змінна, оголошена поза функцією, уважається *глобальною*. До неї можна звертатися з будь-якого місця сценарію.
- Якщо локальна й глобальна змінні мають однакові імена, то в тілі функції локальна змінна "маскує" глобальну.
- Якщо змінна створюється автоматично, тобто якщо вона не оголошена за допомогою ключового слова *var*, але є присутня у лівій частині оператора прямого присвоювання, то вона вважається глобальної й стає доступною з будь-якого місця сценарію.

Порядок виконання роботи

1. Використовуючи один з HTML- редакторів створити HTML- файли із сценаріями прикладів 1–3 наведених у теоретичній частині та подивитися їх роботу у вікні браузера. У разі наявності помилок провести налагодження сценаріїв шляхом відповідного редагування початкових кодів файлів.
2. Проаналізувати роботу скрипта з прикладу 2. Надати пояснення, що до отриманих результатів та навести їх у звіті.
3. Із прикладу 3 створити скрипт, у вигляді окремого файлу, що викликається з довільної HTML – сторінки. Перевірити взаємодію HTML- файлу з скриптом.
4. Відредагувати сценарій прикладу 2., з можливістю вводу числових змінних через діалогові вікна, використовуючи **метод prompt**, для виводу результатів використовувати **метод alert.**
5. Перевірити роботу відредагованого скрипта, порівнявши результати роботи зі скриптом прикладу 2.

Лабораторна робота № 11 Складання та налагодження простих сценаріїв на мові JavaScript (Jscript)

Мета роботи: Навчитися створювати та налагоджувати нескладні сценарії на мові JavaScript (Jscript) у HTML- файлах із використанням діалогових вікон, арифметичних операцій та операцій порівняння та відношення.

Теоретичні відомості

Уведення в техніку складання сценаріїв.

11.1 Вивід рядка тексту

Почнемо вивчення з простого приклада сценарію (або програми), у якому в текст документа HTML виводиться рядок "Ласкаво просимо у світ програмування на JavaScript!". Оператори сценарію, написаного на JavaScript, обробляються інтерпретатором JavaScript, убудованим у браузер Internet Explorer.

```
1<!DOCTYPE html>
2<html>
3<head>
4     <title>JavaScript</title>

5</head>
6   <script>
7   function hello(){
8          document.writeln ("<H1> Мова програмування JavaScript!</H1>");
9      alert('Привіт світ!');}
10   </script>
11<body>
12  <h2>Перша програма на JavaScript</h2>
13   <INPUT TYPE="button" value="Hello" onClick="hello()">
14</body>
15 </html>
```

Увага! Мова JavaScript чуттєва до регістра символів.

Для зручності всі рядки розглянутих прикладів сторінок і сценаріїв пронумеровуються, (ці номери не є частиною сторінок HTML або сценаріїв на JavaScript).

Перші 5 рядків HTML - сторінки в коментарях не мають потреби.

У розглянутому нами випадку сценарій на JavaScript міститься у розділі заголовка. Оскільки браузер інтерпретує зміст заголовної секції, у першу чергу,, то програми на JavaScript, включені в цей розділ будуть виконуватися до того, як буде оброблений розділ <BODY> HTML-документа.

Тег <SCRIPT>, що вказує браузерові на те, що далі буде текст, що є частиною сценарію. Атрибут LANGUAGE служить для визначення мови, на якому написаний сценарій, у даному випадку — JavaScript.

У рядках 7 і 9

```
document .writeln (
"<H1> Мова програмування JavaScript! </H1>"   ) ;
```

указується інструкція інтерпретаторові мови сценаріїв браузера виконати якісь дії, а саме — вивести в сторінку Web рядок, зазначений у цьому операторі між символами подвійних лапок ("). Дозволяється використання одиночних лапок (‘). Символ - крапка з комою є ознакою завершення поточного оператора сценарію.

У рядках 9 і 10 використовується об'єкт браузера з ім’ям document, що представляє поточний документ, який переглядається у браузері. Використовуючи об'єкт document, програмісти можуть уводити текст мовою HTML у документ HTML. Крім цього, браузер містить повний комплект об'єктів, що представляють всі елементи документа HTML, за допомогою яких сценарії можуть маніпулювати елементами документа.

У розглянутому операторі ми викликаємо метод writeln об'єкта document, щоб вивести в документ HTML рядок. У дужках, що випливають за ім'ям методу writeln, вказуються аргументи (параметри), використовувані методом для виконання своєї задачі (або своєї дії). У даному випадку метод writeln дає вказівка браузерові вивести в документ рядок, зазначений в аргументі. Якщо цей рядок містить елементи HTML, то браузер інтерпретує їх і відображає отриманий результат на екрані. У нашому прикладі браузер виводить на екран фразу Ласкаво просимо у

світ програмування на JavaScript! як заголовок рівня H1, оскільки ця фраза, включена в елемент H1.

У рядку 11 указується тег </SCRIPT>, що позначає кінець сценарію.

У рядку 13 тег </HEAD> указує на закінчення розділу <HEAD> документа HTML. У цьому ж рядку, один за одним, указуються теги <BODY> і </BODY>, між якими текст HTML не задається. Таким чином, даний документ містить порожній розділ тіла документа. Тег у рядку 14 закриває документ HTML.

Діалогові вікна попереджень, звичайно, використовуються для виводу користувачеві якихось важливих повідомлень під час перегляду їм сторінки Web. За допомогою JavaScript задача виводу повідомлення у вікно діалогу вирішується дуже просто.

```
1<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
2 <HTML>
3  <!-- welcome.html -->
4  <!-- Вивід рядка у діалогове вікно
5 попереджень -->
6<HEAD>
7
8<SCRIPT>
9 window.alert("Ласкаво просимо у світ програмування на
JavaScript!");
10 </SCRIPT>
11
12</HEAD>
13
14 <BODY>
15 <P> Клацніть на кнопці «Обновити» для запуску сценарію </P>
16 </BODY>
17 </HTML>
```

11.2 Арифметичні операції в JavaScript

Для виводу діалогового вікна попередження використовується убудований у браузер об'єкт window. Через параметр методу alert цього об'єкта передається виведені у вікні рядки тексту. Для виводу повідомлень у вигляді декількох рядків можна використовувати по тексту сполучення символів \n.

Таблиця 11.1

Арифметичні операції в JavaScript

Операція JScript	Символ операції	Приклад алгебраїчного вираження	Приклад вираження на JavaScript
Додавання	+	f + 7	f + 7
Віднімання	-	p – c	p - c
Множення	*	bm	b * m
Ділення	/	x/y, — або x:y	x/y
Ділення по модулю	%	r mod s	r % s

У нашому наступному сценарії користувач буде вводити з клавіатури два цілих числа, а у відповідь одержувати суму цих чисел, розраховану сценарієм.

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2 <HTML>
3   <!--  Addition.html -->
4 <HEAD>
5 <TITLE> Програма обчислення суми двох цілих чисел </T1TLE>
6 <SCRIPT>
7    var firstNumber,   // перший рядок, уведений користувачем
8        secondNumber,  // другий рядок, уведений користувачем
9 number1,             // перший доданок
10 number2,            // другий доданок
11 sum;                // сума чисел numberl і number2
12 // читання першого, введеного користувачем числа у формі рядка
13  firstNumber = window.prompt("Уведіть перше ціле число", "0");
14// читання другого, введеного користувачем числа у формі рядка
15     secondNumber = window.prompt( "Уведіть друге ціле число ", "0" );
16// перетворення даних із строкового типу до цілого типу,
17    numberl = parseInt ( firstNumber ) ;
18     number2 = parseInt( secondNumber );
19// обчислення суми двох чисел
20      sum = numberl + number2;
21// виведення  отриманого результату
22 document.writeln( "<H1>Сума чисел дорівнює "+sum+"</H1>" );
23</SCRIPT>
24 </HEAD>
25<BODY>
26 <P> Клацніть на кнопці  (Обновити) для пере запуску сценарію.
27 </P>
28 </BODY>
29 </HTML>
```

у рядках 7-11 приклада оголошуються перемінні.

```
var   firstNumber,              // перший  рядок,   уведений  користувачем
secondNumber,       // другий  рядок,   уведений  користувачем
numberl,              // перший  доданок
number2,              // другий  доданок
sum;                 // сума  чисел  numberl  і number2

// - однорядковий коментар
 /* Приклад
     багаторядкового
               коментарю. */
```

У рядку 13

```
firstNumber  = window.prompt(   "Уведіть  перше  ціле  число",   "0"   );
```

Викликуваний у цьому операторі метод prompt об'єкта window виводить на екран вікно діалогу. Перший параметр методу prompt задає рядок тексту, що пояснює користувачеві, що він повинний увести в поле введення. Це повідомлення називається запрошенням, тому що в ньому користувача запрошують виконати деяку дію. У необов'язковому другому параметрі можна задати значення за замовчуванням, що буде виводитися в поле введення; якщо другий параметр не заданий, то значення за замовчуванням у поле уведення виводитися не буде.

Результат, що повертається методом document.prompt — уведений користувачем текстовий рядок, привласнюється перемінної firstNumber за допомогою операції присвоювання (=).

Рядки 14 і 15

```
//  читання  другого введеного користувачем числа у  формі  рядків

secondNumber = window.prompt(   "Уведіть друге ціле число  ",   "0");
```

У цьому операторі викликається друге вікно діалогу prompt, у якому користувач повинний увести інше ціле число, що бере участь в обчисленні суми.

Потім у рядках 17 і 18

```
numberl  = parseInt(firstNumber);
number2  = parseInt(secondNumber);
```

виконується перетворення введених користувачем рядків до значень цілого типу (int), що можуть брати участь в арифметичних обчисленнях.

Функція parseInt виконує перетворення свого строкового аргументу в число цілого типу. Повертається в рядку 17 функцією parseInt ціле значення,що привласнюється перемінній number1.

В операторі присвоювання - рядок 20

sum = numberl + number2;

обчислюється сума двох перемінних numberl і number2, а отримане значення привласнюється за допомогою операції присвоювання третій перемінній – sum.

11.3. Операції порівняння й відносин

У мові JavaScript використовується керуюча структура if. Дана структура використовується в програмах для прийняття рішень на основі того, чи є умова цієї структури істиною або неправдою. Умовне вираження структури if може бути складене з операцій порівняння й відносини перерахованих нижче в таблиці.

Таблиця 11.2

Оператори порівняння в JavaScript

Алгебраїчна операція порівняння або відносини	Операція порівняння або відносини JavaScript	Приклад умовного вираження JavaScript	Опис приклада
Операції порівняння			
=	= =	x = = y	x дорівнює в
<>	! =	x != y	x не дорівнює в
Операції відносини			
>	>	x > y	x більше в
<	<	x < y	x менше в
≥	>=	x >= y	x більше або дорівнює в
≤	<=	x <= y	x менше або дорівнює в

Порядок виконання роботи.

1. Використовуючи один з HTML- редакторів створити HTML- файли із сценаріями прикладів наведених у теоретичній частині та подивитися їх роботу у вікні браузера. У разі наявності помилок провести налагодження сценаріїв шляхом відповідного редагування початкових кодів файлів.
2. Виконати індивідуальне завдання 1-3. Номер варіанта відповідає порядковому номеру студента у журналі.

Завдання 1 Розробити алгоритми та скласти сценарії на мові JavaScript .

1. Дано два ненульові числа. Знайти суму, різницю, добуток та приватне їх модулів.
2. Дано два кола із загальним центром та радіусами R_1 та R_2 ($R_1 > R_2$). Знайти площі цих кіл S_1 та S_2, а також площа S_3 кільця, зовнішній радіус якого дорівнює R_1 а внутрішній радіус дорівнює R_2:

   $S_1 = \pi(R_1)^2$, $S_2 = \pi (R_2)^2$, $S_3 = S_1 - S_2$. _ _
3. Дана довжина *L* кола. Знайти її радіус *R* та площа *S* кола, обмеженого цим колом, враховуючи, що $L = 2\pi R$, $S = \pi R^2$. Як значення π використовувати Math.PI.
4. Дана площа S кола. Знайти його діаметр D та довжину L кола, що обмежує це коло, враховуючи, що L = 2 πR , S = πR 2 . Як значення π використовувати Math.PI.
5. Дано три точки *A, B, C* на числовій осі. Знайти довжини відрізків *AC та BC та їх* суму.
6. Дано три точки *A, B, C* на числовій осі. Точка *C* розташована між точками *A і B.* Знайти добуток довжин відрізків *AC* та *BC .*
7. Дані координати двох протилежних вершин прямокутника: (x_1 , y_1), (x_2 , y_2). Сторони прямокутника паралельні до осей координат. Знайти периметр та площу даного прямокутника.
8. Знайти відстань між двома точками із заданими координатами (x_1 , y_1) і (x_2 , y_2) на площині. Відстань обчислюється за формулою : $(x_2 - x_1)^2 + (y_2 - y_1)^2$.
9. Дано змінні *A, B, C.* Змінити їх значення, перемістивши вміст *A* **в** *B , B* **в** *C , C* **в** *A* ,і вивести нові значення змінних *A, B, C.*
10. Дано число *A* . Обчислити A^8 , використовуючи допоміжну змінну та три операції множення. Для цього послідовно знаходити A_2 , A_4 , A_8 . Вивести всі знайдені ступені числа *A.*
11. Дано число *A* . Обчислити A_{15} , використовуючи дві допоміжні змінні та п'ять операцій множення. Для цього послідовно знаходити A_2 , A_3 , A_5 , A_{10} , A_{15} . – Вивести всі знайдені ступені числа *A.*
12. Дано значення кута ***a*** **в** градусах (0 < ***a*** < 360). Визначити значення цього ж кута в радіанах з огляду на те, що 180° = π радіан.

13. Дано значення кута *a в* радіанах (0 < *a* < 2 *π*). Визначити значення цього ж кута в градусах з огляду на те, що 180° = *π* радіанів . Як значення *π* використовувати 3.14.

14. Швидкість першого автомобіля *V 1* км/год, другого - *V 2* км/год, відстань між ними *S* км. Визначити відстань між ними через *T* годин, якщо автомобілі віддаляються один від одного. Дана відстань дорівнює сумі початкової відстані та загального шляху, виконаного автомобілями; загальний шлях = час сумарна швидкість.

15. Розв'язати лінійне рівняння *A x* + *B* = 0, задане своїми коефіцієнтами A *i B* (коефіцієнт *A* не дорівнює 0).

16. Наведено розмір файлу в байтах. Використовуючи операцію поділу націло, знайти кількість повних кілобайтів, які займає даний файл (1 кілобайт = 1024 байти).

17. Дано цілі позитивні числа *A* та *B* (*A* > *B*). На відрізку довжини *A* розміщено максимально можливу кількість відрізків довжини *B* (Без накладень). Використовуючи операцію поділу націло, визначити кількість відрізків *B* , розміщених на відрізку *A* .

18. Дано двозначне число. Вивести спочатку його ліву цифру (- десятки), а потім його праву цифру (одиниці). Для знаходження десятків використовувати операцію поділу націло на 10, для знаходження одиниць - операцію взяття залишку від поділу.

19. Дано двозначне число. Знайти суму та добуток його цифр.

20. Дано двозначне число. Вивести число, отримане при перестановці цифр вихідного числа.

21. Дано тризначне число. Знайти суму та добуток його цифр.

22. З початку доби пройшло *N* секунд (*N* - ціле). Знайти кількість повного годинника, що пройшов з початку доби.

23. Дано ціле число *A.* Перевірити істинність висловлювання: «Число A *є* позитивним».

24. З початку доби пройшло *N* секунд (*N* - ціле). Знайти кількість повних хвилин, що минули з початку останньої години.

25. Дано ціле число, більше 999. Використовуючи одну операцію поділу націло і одну операцію взяття залишку від поділу, знайти цифру, що відповідає розряду сотень у записі цього числа.

Завдання 2

1. Обчислити коріння квадратного рівняння загального виду a< >0 ax^2 + bx + c = 0 у ділянці дійсних чисел.
2. Визначити чи знаходиться точка M (x,y)всередині області на її межі поза області обмеженого кола радіусом R з центром на початку координат.
3. Мінімальний елемент з x_1 , x_2 , x_3 .
4. Розташуйте числа A, B , C у порядку зростання (зменшення).
5. Дано три дійсних числа A,B ,C. Визначте, скільки серед них негативних (позитивних).
6. Складіть програму введення 4 символьних змінних та визначте скільки
серед них цифр та виведіть їх на екран.
7. Знайти залишок від ділення значення цілого виразу C = K * (A + B) на 4 і вивести повідомлення про величину залишку. Якщо залишок "0" то вираз C залишити без зміни, якщо " 1" або "3" - зменшити на величину залишку, якщо "2" збільшимо на величину залишку. Нове значення "C" вивести на екран. Програму скласти із застосуванням оператора CASE .
8. Дано ціле число. Якщо воно є позитивним, додати до нього 1; інакше не змінювати його. Вивести отримане число.
9. Дано ціле число. Якщо воно є позитивним, додати до нього 1; якщо негативним, то відняти від нього 2; якщо нульовим, то замінити його на отримане число.
10. Дано три цілих числа. Знайти кількість позитивних та кількість негативних чисел у вихідному наборі.
11. Дано дві змінні цілого типу: *A* та *B* . Якщо їх значення не рівні, то присвоїти кожній змінній суму цих значень, а якщо рівні, то присвоїти змінним нульові значення. Вивести нові значення змінних *A* та *B* .
12. Дано три числа. Знайти середнє з них (тобто число, розташоване між найменшим та найбільшим).
13. Дано три числа. Знайти суму двох найбільших із них.
14. Дано три змінні речовинного типу : *A* , *B* , *C.* Якщо їх значення упорядковано за зростанням, то подвоїти їх; в іншому випадку замінити значення кожної змінної на протилежне. Вивести нові значення змінних *A* , *B* , *C.*
15. Дано три цілих числа, одне з яких відмінне від двох інших, рівних між собою. Визначити порядковий номер числа, відмінного від інших.

16. На числовій осі розташовані три точки : *A* , *B* , *C.* Визначити, яка з двох останніх точок (*B* або *C*) розташована ближче до *A* , і вивести цю точку та її відстань від точки *A* .
17. Дано координати точки, що не лежить на координатних осях *OX* та *OY* . Визначити номер координатної чверті, де знаходиться дана точка.
18. Дано три дійсні числа. Вибрати їх ті, які належать інтервалу (1, 3).
19. Дано дійсні числа x, y. Якщо x, y негативні, кожне значення замінити його модулем; якщо негативне лише одне їх, то обидва значення збільшити на 0.5; якщо обидва значення не є негативними і жодне з них не належить відрізку [0.5, 2.0], то обидва значення зменшити в 10 разів; в решті випадків x, y залишити без зміни.
20. Визначити та вивести на друк номер квадранта, в якому розташована точка M(x, y), x та y задані речові числа.
21. З величин, що визначаються виразами a = sinx , b = cosx , c = ln | x | при заданому x, визначити та вивести на екран дисплея мінімальне значення.
22. Визначити, яка з двох точок – $M_1(x_1,y_1)$ або $M_2(x_2, y_2)$ – розташована ближче до початку координат. Вивести на екран дисплея координати цієї точки.
23. Визначити, яка з двох фігур (коло чи квадрат) має велику площу. Відомо, що сторона квадрата дорівнює a, радіус кола r. Вивести на екран назву та значення площі більшої фігури.
24. Визначити, чи точка M(x,y) потрапляє в коло радіусом r з центром у точці (x_0,y_0)
25. Визначте, чи серед заданих цілих чисел A , B , C хоча б одне парне.
26. Задані площі кола та квадрата. Визначте, чи поміститься квадрат у колі.

Завдання 3

1. Напишіть _ сценарій , що виводити числа від 1 до 4 в одну строчку, відокремлюючи їх одне вид іншого одним перепусткою. Складіть два варіанти програми , в одному документі, у яких будуть використовуватися наступні методи :

а) один оператор **document .writeln ;**

b) чотири оператори **document .write .**

2. Напишіть сценарій , у якому користувача виводиться запрошення ввести два числа. Після введення чисел розраховуються і виводяться в текст HTML сума , добуток , різниця вид ділення цих чисел упорядковані за зростанням .

3. Напишіть сценарій, у якого користувачеві виводиться запрошення «увести два цілі числа», а після введення цих чисел визначається більше з цих чисел і у вікні діалогу виводиться текст HTML , у якому вказується значення більшого з цих чисел, за яким випливає текст " **більше з двох чисел".** Якщо введено два рівні числа, сценарій повинне бачити повідомлення **"Введено два рівні числа".**

4. Напишіть сценарій, що отримує від користувача три цілі числа і виводитиме через діалогове вікно **alert** суму, середнє значення, добуток, а також найбільше щонайменше з трьох чисел.

5. Напишіть сценарій , у якому _ користувач вводить значення радіуса кола в основі якого розраховуються й виводяться у текст документа HTML діаметр , довжина кола та площа окружності за вибором користувача . Як числоπ Використовуйте число 3.14159 . Зауваження : ви можете використовувати визначену константу **Math.PI ,** у якої зберігається значення числа π. У цій константі число πнаводиться з більшою точністю , ніж 3.14159. Об'єкт **Math** є вбудованим об'єктом JavaScript , що використовується при проведенні математичних обчислити . Використовуйте наступні формули : діаметр = 2 r , довжина кола = 2π r , площа кола = π r^2 , де r — радіус кола .

6. Напишіть сценарій у якому користувач виводить 4 числа . Найбільше та найменше чисел виводяться у документ HTML .

7. Користувач виводить із клавіатури число. Якщо воно більше 10, то виводиться повідомлення “ Ви ввели число > 10”. Інакше вивести “Ваше число не перевищує10”

8. На запит сценарію вводитиметься 6 чисел. Вивести в порядку спадання суму пар чисел (1,6) (2,5) (3,4).

9. На запит сценарію користувач вводить змінну A. Якщо A=1, то вивести суму чисел від 1 до 20. Якщо A=2, то вивести добуток чисел від 1 до 8. В інших випадках вивести значення введеного параметра як текст HTML . A – цілі, натуральні.

10. Напишіть сценарій, у якому користувач виводитиме 5 цілих чисел, серед яких сценарій знаходить максимальне та мінімальне число і виводитиме ці значення як текст HTML .

11. Напишіть сценарій , у якому користувач виводити ціле число, а сценарій виводити текст HTML, що пов ідомляє , чи введене число є парним чи непарний ім .

12. Напишіть сценарій , у якому зчитуються два введені користувачем числа і визначається , чи є друге число дільником першого числа. Результат аналізу виводиться у вигляді тексту HTML . _

13. Користувач вести 3 числа A, B , C. Напишіть сценарій , що виводити у документ HTML результат аналізу умов :
Якщо (A > B) & (B < C) вивести A^2 +BC .

Якщо (A < B) or (B > C) вивести $C^2 + B^3 A$.

14. Напишіть сценарій, що запрошує від користувача п'ять цілих чисел і виводитиме текст HTML , в якому повідомляється кількість введених позитивних, від'ємних і нульових чисел.

15. Напишіть сценарій , у якому користувач виводити своє _ ім'я й прізвище як дві незалежні рядки. Потім сценарій поєднує та виводити ім'я та прізвище , розділивши їх про білий .
Якщо перша строчка більша за іншу, то виводитися в документ HTML отриманий рядок як текст HTML у тегах < B > та </ B > , інакше використовувати теги < I > , </ I >

16. Напишіть сценарій, у якому введене користувачем число, що складається з п'яти цифр, розділяється на складові цифри числа, а потім виводиться текст HTML , у якому виводяться ці цифри, відділені один від одного трьома перепустками. Наприклад , якщо користувач увів число **42339,** сценарій винен він бачити результат **4 2 3 3 9 .**

17. Напишіть сценарій, у якому розраховуються квадратні або кубічні значення цілих чисел від 1 до 10 залежно від запиту сценарію. Отримані результати повинні виводитися у столбик як нумерований список HTML.

18. Напишіть сценарій , у якому користувач виводити своє _ ім'я , прізвище та професію як незалежні рядки. Потім сценарій поєднує ім'я та прізвище , розділивши їх про пуском , і виводити у документ HTML отримана строка повідомлення , вказуючи ваше _ ім'я , фамилия та спеціальність , в залежності від професії :

Професія	Спеціальність
Будівельник	Штукатур
Лікар	Терапевт
Викладач	Філософ

19. Напишіть сценарій, який отримує від користувача два цілі числа і виводитиме через діалогове вікно **alert** , за вибором користувача, суму квадратів чисел, добуток, а також найбільше з двох чисел.
20.
21. Напишіть сценарій , у якому користувач виводити чотири числа і виводити максимальне з них та добуток чисел без мінімального .

22. Напишіть сценарій , у якому користувач виводити два числа. П і драхувати їх є середнє та вив є ці три числа в порядку зростання .

23. Напишіть _ сценарій , у якому користувач виводити три цілі позитивних числа і визначає , чи можуть ці числа бути сторонами трикутника . Результат вивчити у документ HTML .

24. Напишіть сценарій , у якому користувач виводити три числа і виводити у вікно повідомлення середнє геометричні двох найменших чисел.

25. Напишіть сценарій , у якому користувач вводити 5 чисел і вив о дити середнє , суму й добуток найменшого та найбільшого серед уведених .

26. Напишіть сценарій , у якому користувач виводити текст , а його вив ід здійснюється в зворотному порядку .

Лабораторна робота № 12 Клієнтські сценарії. Використання регулярних виразів

Мета роботи:

1) Одержати уявлення про загальні принципи обробки в JavaScript подій, пов'язаних з вікном веб – браузера, веб – сторінкою, що розташована у браузері й елементами документа.
2) Навчитися використати найпростіші елементи регулярних виражень для пошуку підрядка, структура й зміст яких описується нетривіальним шаблоном.

Теоретичний матеріал

12.1 Обробка подій в JavaScript

Популярність JavaScript багато в чому обумовлена саме тим, що написаний на ньому сценарій може реагувати на дії користувача й інші зовнішні події. Кожна з подій пов'язана з тим або іншим об'єктом: формою, гіпертекстовим посиланням або навіть із вікном, що містить поточний документ.

Як приклади зовнішніх подій, на які можуть реагувати об'єкти JavaScript, можна привести наступні.

- закінчення завантаження документа у вікно (або закінчення завантаження документів в усі фрейми вікна). Ця подія пов'язане з об'єктом *window*;
- щиглик мишею на об'єкті. Ця подія може бути пов'язане з інтерактивним елементом форми або з гіпертекстовим посиланням;
- одержання об'єктом фокуса уведення. Ця подія може бути пов'язана з об'єктами типу Text, Password і з іншими інтерактивними елементами;
- передача на сервер даних, уведених користувачем за допомогою інтерактивних елементів. Зв'язується з формою.

Обробка події здійснюється за допомогою спеціально призначеного для цього фрагмента коду, називаного *оброблювачем події*. Для кожної події JavaScript надає свій оброблювач. Однак при побудові сценарію

можна створювати власний оброблювач події й використати його замість оброблювача, заданого за замовчуванням.

Ім'я оброблювача визначає, яку подію він повинен обробляти. Так, для того щоб сценарій потрібним чином відреагував на щиглик мишею, використається оброблювач із ім'ям *onClick*, для обробки події, що полягає в одержанні фокуса уведення, - оброблювач *onFocus*.

Для того щоб указати інтерпретаторові JavaScript на те, що обробкою події повинен займатися оброблювач, необхідно включити в HTML-дескриптор наступне вираження:

ім'я_оброблювача="команди_оброблювача"

Це вираження включається в тег, що описує об'єкт, з яким зв'язана подія.

Наприклад, якщо необхідно обробити подію, що полягає в одержанні фокуса полем уведення, дескриптор, що описує цей інтерактивний елемент, повинен мати приблизно наступний вид:

<input type="text" name="Inform" onFocus="handleFocus();">

Ім'я оброблювача є одним з атрибутів HTML-дескриптора, а команди, призначені для обробки події, виступають у ролі значення цього атрибута. У цьому випадку обробка події виробляється в тілі функції *handleFocus*(). У принципі, оброблювачем може бути не тільки функція, але й будь-яка послідовність команд JavaScript у вигляді складеного оператора.

Приклад 1.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Событие onfocus</title>
  <style>
   input {
    border: 1px solid #ccc; /* Параметри рамки */
    padding: 3px; /* Поля навколо текста */
    color: #666; /* Колір текста */
    width: 50%; /* Ширина поля */
   }
  </style>
 </head>
 <body>
```

```
<form>
 <p><input type="text" class="text" value="Введіть текст"
  onfocus="this.value=' '" onblur="this.value='Новий текст'"></p>
 </form>
</body>
</html>
```

У даному прикладі при одержанні фокуса в текстовому полі ховається текст (подія onfocus) , а при втраті фокуса (подія onblur), навпаки, у полі додається текстовий рядок.

Наступний приклад демонструє обробку події, пов'язаного з наведенням курсору миші на гіперпосилання:

```
<a href = ”http://www.myhp.edu” onmouseover="alert('An onMouseOver event'); return
false">
 <img src="my.gif" width="50" height="50">
 </a>
```

Нижче приводиться повний текст HTML документа із JavaScript сценарієм, у якому обробляється подія натискання кнопки миші, і визначається, яка саме з них була натиснута:

Приклад 2.

```
<html>
<head>
<script>
function whichButton(event)
{
if (event.button == 2)
  {
  alert("Ви клацнули правою кнопкою миші!");
  }
else
  {
  alert("Ви клацнули лівою кнопкою миші!");
  }
}
</script>
</head>
<body onmousedown="whichButton(event)">
<p>Клацніть будь-якою кнопкою миші в будь-якому місці документа</p>
</body>
</html>
```

Таким чином, для того щоб обробити яку-небудь стандартну подію в браузері, необхідно додати в підходящий HTML тег атрибут, що відповідає цій події, указавши як значення атрибута ім'я JavaScript функції. Список атрибутів, які визначені для HTML тегів приводиться нижче:

Таблиця 12.1

IE: Internet Explorer, **F**: Firefox, **O**: Opera, **W3C**: стандарт

Атрибут	Опис	Номер версії браузера			W3C
		IE	F	O	
onabort	Перерване завантаження зображення	4	1	9	Так
onblur	втрата фокуса елементом	3	1	9	Так
onchange	Зміна вмісту в поле уведення	3	1	9	Так
onclick	Щиглик миші на об'єкті	3	1	9	Так
ondblclick	Подвійний щиглик миші на об'єкті	4	1	9	Так
onerror	Помилка при завантаженні зображення або документа	4	1	9	Так
onfocus	Одержання фокуса елементом	3	1	9	Так
onkeydown	Натискання клавіші	3	1	Немає	Так
onkeypress	Клавіша натиснута	3	1	9	Так
onkeyup	Отжатие клавіші	3	1	9	Так
onload	Завершення завантаження сторінки або зображення	3	1	9	Так
onmousedown	Натискання кнопки миші	4	1	9	Так
onmousemove	Переміщення курсору миші	3	1	9	Так
onmouseout	Зсув курсору миші з об'єкта	4	1	9	Так
onmouseover	Наведення курсору миші на об'єкт	3	1	9	Так
onmouseup	Отжатие кнопки миші	4	1	9	Так
onreset	Кнопка "Reset" натиснута	4	1	9	Так
onresize	Зміна розміру вікна	4	1	9	Так
onselect	Виділення тексту	3	1	9	Так
onsubmit	Кнопка "Submit" натиснута	3	1	9	Так
onunload	Відхід з веб-страницы	3	1	9	Так

12.2 Регулярні вирази

Регулярні вирази - система пошуку текстових фрагментів в електронних документах, заснована на спеціальній системі запису зразків для пошуку.

Зразок, що задає правило пошуку, називається «*шаблоном*». Застосування регулярних виражень принципово перетворило технології електронної обробки текстів.

За допомогою регулярних виражень можна задавати структуру шуканого шаблону і його позицію усередині рядка (наприклад, на початку або наприкінці рядка, на границі або не на границі слова).

При описі структури шаблона використаються:

- гнучка система квантифікаторов (операторів повторення);
- оператори опису наборів символів і їхнього типу (числові, нечислові, спеціальні).

Для того, щоб задати положення шуканого фрагмента усередині рядка, можна використати один з наступних операторів:

Оператори шаблону Таблиця 12.2

Подання	Позиція
^	Початок рядка
$	Кінець рядка
\b	Границя слова
\B	Не границя слова
(?=шаблон)	Шуканий рядок випливає після зазначеного рядка (з переглядом уперед)
(?!=шаблон)	Шуканий рядок не випливає після зазначеного рядка (з переглядом уперед)
(?<=шаблон)	Шуканий рядок випливає після зазначеного рядка (з переглядом назад)
(?<!шаблон)	Шуканий рядок не випливає після зазначеного рядка (з переглядом назад)

Крім того, мова регулярних виражень надає набір квантифікаторов, що дозволяють указати число повторень шаблона:

Квантифікатори Таблиця 12.2

Подання	**Число повторень**
{*n*}	Рівно *n*
{*m*, *n*}	Від *m* до *n* включно
{*m*,}	Не менш *m*
{,*n*}	Не більше *n*

Є й більше прості квантифікатори:

Прості квантифікатори Таблиця 12.3

Подання	**Число повторень**	**Еквівалент**
*	Нуль або більше	{*0*,}
+	Одне або більше	{*1*,}
?	Нуль або одне	{*0*,*1*}

Для завдання усередині шаблона групи символом можна використати наступні оператори:

Оператори Таблиця 12.4

Оператор	**Опис**
[xyz]	Будь-який символ із зазначеної безлічі
[^xyz]	Будь-який символ,що не входить у зазначену безліч
[x-z]	Будь-який символ із зазначеного діапазону
[^x-z]	Будь-який символ, що не входить у зазначений діапазон
. (*крапка*)	Будь-який символ крім символів розриву або переносу рядка
\w	Будь-який буквено-цифровий символ, включаючи символ підкреслення
\W	Будь-який не буквений символ
\d	Будь-яка цифра

\D	Будь-який нецифровий символ
\s	Любий не відображуваний символ
\S	Будь-який символ (крім не відображуваних символів)

Для угруповання окремих частин шаблона можна використати наступні оператори:

Оператори шаблону Таблиця 12.3

Оператор	**Опис**
( )	Пошук групи символів усередині дужок і збереження знайденої відповідності
(?:)	Пошук групи символів усередині дужок без збереження знайденої відповідності
|	Комбінування частин в одне вираження з наступним пошуком будь – якої із частин окремо. Аналогічно операторові «АБО».

Якщо шаблон пошуку включає спеціальні (як правило не відображувані) символи, для їхнього опису можна використати наступні позначення:

Спеціфльні символи Таблиця 12.5

Позначення	**Опис**
\0	Символ з нульовим кодом
\n	Символ нового рядка
\r	Символ початку рядка
\t	Символ табуляції
\v	Символ вертикальної табуляції
\xxx	Символ, що має заданий восьмеричний ASCII код *xxx*
\xdd	Символ, що має заданий шістнадцятковий ASCII код *dd*
\uxxxx	Символ, що має ASCII код виражений ЮНІКОДОМ *xxxx*

Квантіфікаторам у регулярних вираженнях відповідає максимально довгий рядок з можливих (тобто квантіфікатори є «жадібними»). Це може приводити до деяких проблем. Наприклад, шаблон (**<.*>**), що описує на

перший погляд теги HTML насправді буде виділяти більші фрагменти в документі.

Наприклад, рядок виду

<i>Регулярні вираження<i> - зручний інструмент для пошуку в рядках

формально відповідає зазначеному вище шаблону

Для рішення даної проблеми можна використати два підходи.

1. У регулярному вираженні враховуються символи, що не відповідають бажаному зразку (наприклад, **<[^>]*>** для вищеописаного випадку).
2. Визначення квантифікатора як *нежадібного* (ледачого) - більшість реалізацій дозволяють це зробити, додавши після нього знак питання.

Наприклад, по шаблоні (<.*?>) будуть знайдені всі теги з розглянутого рядка.

Таким чином, виходять наступні «нежадібні» модифікації квантіфікаторів:

«Нежадібні» модифікації квантіфікаторів Таблиця 12.4

Квантіфікатор	Опис
*?	«не жадібний» еквівалент *
+?	«не жадібний» еквівалент +
{n,}?	«не жадібний» еквівалент {n,}

Треба, однак, мати на увазі, що використання «ледачих» квантіфікаторів може привести до ситуації, коли вираженню відповідає занадто короткий, зокрема, порожній рядок.

12.3 регулярних виражень в JavaScript

При пошуку по тексту можна використати шаблон, що описує підрядок. В JavaScript такий шаблон може бути описаний за допомогою об'єкта ***RegExp.*** У найпростішому випадку такий шаблон описує окремий символ, однак має сенс його використати для регулярних виражень.

Наступний нижче код описує *RegExp* об'єкт із ім'ям *pttn*, що містить регулярне вираження, що описує ціле десяткове число:

var *pattn* = new RegExp("/[0-9]+/");

Об'єкт *RegExp* має три убудованих методи: ***test*(), *exec*() і *compile*()**.

- Метод ***test*()** виконує пошук по шаблону:

```
var pattn = new RegExp("[0-9]+");
document.write(pattn.test("38 папуг"));
```

Результат:

true

- Метод ***exec*()** виконує пошук підрядка по шаблону й повертає знайдені відповідності; якщо відповідностей немає, повертається значення *null*:

```
var pattn=new RegExp("[0-9]+");
document.write(pattn.exec("38 папуг"));
```

Результат:

38

Якщо необхідно знайти всі відповідності, то при виклику конструктора ***RegExp*** варто вказати додатковий параметр “g”, що вказує на необхідність глобального пошуку.

Приклад 3.

```
var pattn = new RegExp("[0-9]+", "g");
do
{
result  = pattn.exec("1 папуга, 2 папуги,..., 38 папуг");
document.write(" " + result);
}
while (result != null)
```

Результат: 1 2 38 null

- Метод *compile*() застосовується для зміни раніше створеного шаблона.

Приклад 4.

```
var pattn = new RegExp("[0-5]+");
document.write(pattn.exec("38 папуг"));
```

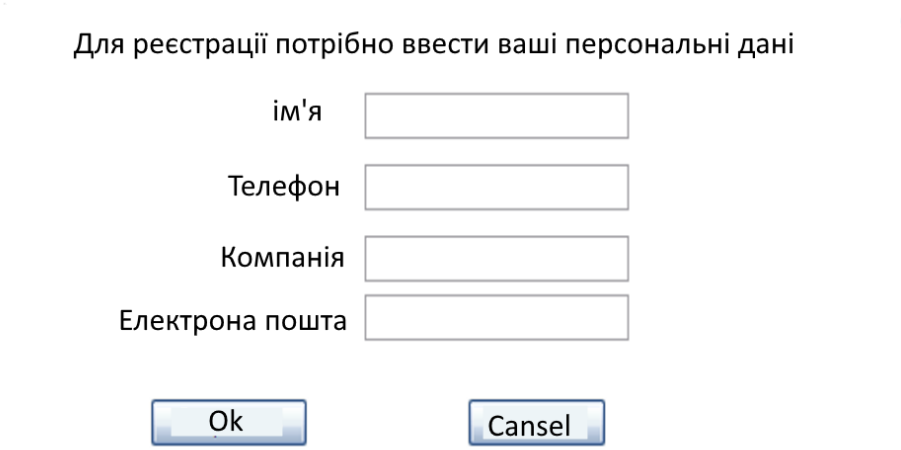
```
pattn.compile("[6-9]+");
document.write(";" + pattn.exec("38 папуг"));
```

Результат:3;8

Порядок виконання роботи

Перевірка значень, уведених користувачем у поля форми для реєстрації.

1. Для виконання лабораторної роботи необхідно створити веб-сторінку, що містить форму з полями, та має вигляд:



2. У тезі <form> додайте оброблювач події відправлення даних виду:
   onSubmit = "*CheckData*(); return false;"

У даному випадку зазначена функція оброблювач *CheckData()*. Оператор *return false*; запобігає автоматичному відправленню даних після виконання функції-оброблювача. Відправлення даних буде виконуватися з оброблювача.

3. Додайте на сторінці секцію JavaScript коду, що описує функцію-оброблювач:

```
function CheckData()
{
  var ans;
ans = confirm("Ви впевнені, що хочете        відправити уведені дані ?");
if (ans) submit();
}
```

Метод ***confirm*** виводить повідомлення у вікні із двома кнопками: "ОК" и "ОТМЕНА".

Як це видно з коду, функція *CheckData()* у випадку підтвердження з боку користувача самостійно викликає метод *submit()* для передачі даних з форми.

4. Додати перевірку значень, уведених у поля форми користувачем.
   - Насамперед, необхідно переконатися в тім, що заповнено всі поля, обов'язкові для уведення. Для цього можна використати перевірку на рівність нулю довжини рядка (властивість *length*), що є значенням вузла дерева документа, що відповідає полю уведення, наприклад: *document.getElementById("uname").value.length*.
   - Наступна перевірка повинна контролювати структуру й уміст полів. Для цього можна використати об'єкт *RegExp*, наприклад для перевірки адреси електронної пошти :

```
var validEMail, pattn;

    pattn = new RegExp("^[\.\-_A-Za-z0-9]+?@[\.\-A-Za-z0-9]+?\.[A-Za-z0]{2,6}$");
validEMail = pattn.test(document.getElementById("email").value));
```

У даному фрагменті описана перевірка структури електронної адреси з поля форми з ідентифікатором "*email*". Для перевірки був використаний шаблон на основі регулярного вираження.

- Побудуйте регулярне вираження, що описує шаблон для перевірки «Телефону»(тільки цифри, кількість цифр у номері 10), і внесіть всі необхідні зміни й доповнення у функцію *CheckData()*.
- Побудуйте регулярне вираження, що описує шаблон для перевірки «Ім’я» (містить тільки кирилицю), і внесіть всі необхідні зміни й доповнення у функцію *CheckData()*.
- Побудуйте регулярне вираження, що описує шаблон для перевірки «Компанія» (може містить як латинь так і кирилицю), і внесіть всі необхідні зміни й доповнення у функцію *CheckData()*.

5. Якщо заповнені поля форми не відповідають заданим вимогам шаблонів повідомити користувача про помилку у відповідному полі (полях). У разі відсутності помилок – повідомлення про успішне відправлення.

Лабораторна робота № 13 Java Script. Робота з Html – сторінкою за допомогою об'єктної моделі документа DOM

Мета роботи: Навчитись працювати з Html – сторінкою за допомогою об'єктної моделі документа DOM

Теоретичні відомості

13.1 DOM: робота з Html-Сторінкою

Об'єктна модель документа не є частиною мови JavaScript. DOM (Document Object Model) – це інтерфейс прикладного програмування для подання документа (наприклад, документа HTML, а також інших) і забезпечення доступу до його елементів і інтерактивної зміни їх властивостей. Більше того, DOM надає механізми для зміни самої структури документа (додавання й видалення елементів, зміна їх умісту). Але це окремий стандарт, у цей час, що розбудовується під егідою W3C.

Об'єктна модель документа (Document Object Model – DOM) є стандартом, запропонованим веб-консорціумом, і регламентує спосіб представлення вмісту документа (зокрема веб – сторінки) у вигляді набору об'єктів. Під умістом розуміється усе, що може розташовуватись на веб –сторінці: малюнки, посилання, абзаци, текст і навіть пробіли.

13.2 Глобальна структура об'єктів браузера

На рисунку схематично відображена структура основних об'єктів браузера.

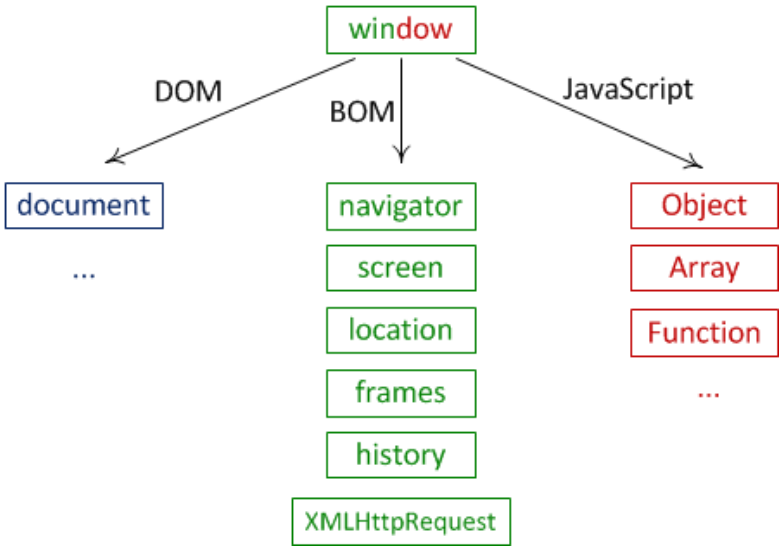


Рис. 13.1 Глобальна структура об'єктів браузера

DOM (**Document Object Model**, об'єктна модель документа) – набір об'єктів, який надає доступ до вмісту сторінки.

BOM (**Browser Object Model**, об'єктна модель браузера) – доступ до об'єктів, що не відносяться до веб – сторінки – фреймам, запитам до сервера, функцій alert/confirm/prompt.

Глобальний об'єкт **window** має дві ролі: 1) вікно браузера. У нього є методи window.focus(), window.open() і інші; 2) глобальний об'єкт Javascript.

13.3 Дерево Dom – об'єктів

Основним інструментом роботи й динамічних змін на сторінці є DOM (Document Object Model) - об'єктна модель, що використовується для XML/ Html – документів.

Згідно Dom – моделі, документ є ієрархією. Кожний Html – тег утворює окремий елемент – вузол, кожний фрагмент тексту - текстовий елемент, і т.ї. DOM - це представлення документа у вигляді дерева тегів. Це дерево утворюється за рахунок вкладеної структури тегів плюс текстові фрагменти сторінки, кожний з яких утворює окремий вузол.

Нижче наведений приклад Html – документу й зображення дерева Dom – об'єктів для нього:

```
<html>
  <head>
    <title>Документ</title>
```

```
  </head>
  <body>
    <div id="datakeeper">Заголовок</div>
   <ul>
      <li>Текст 1</li>
      <li>Текст 2</li>
   </ul>
      <div id="footer">Кінець документа</div>
  </body>
</html>
```

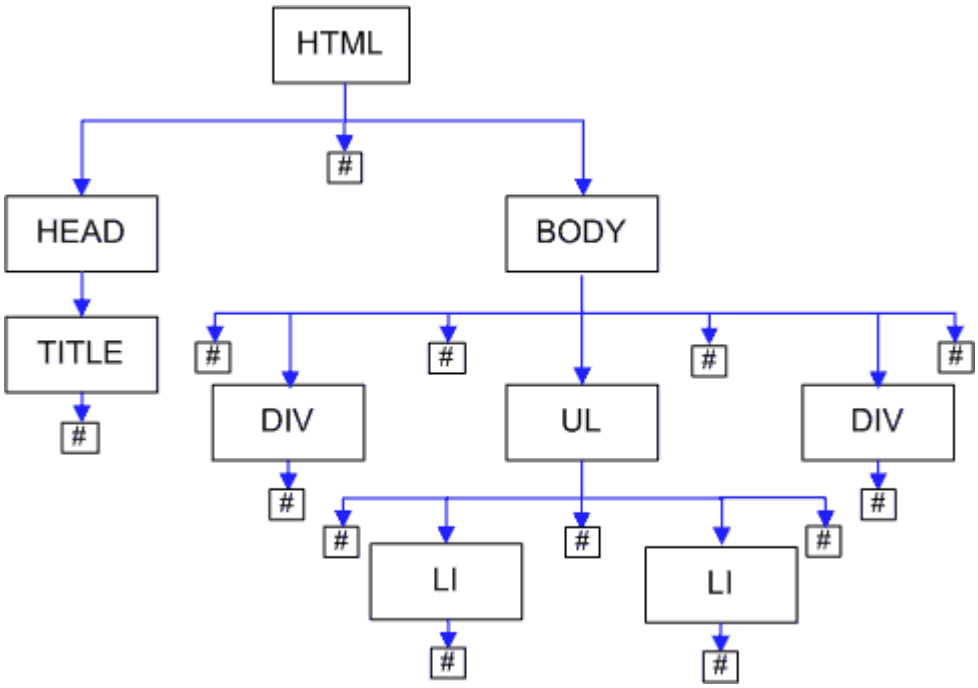


Рис. 13.2 Dom дерево

На малюнку для стислості текстові вузли позначені «#». Кожний Dom –елемент є об'єктом і надає властивості для маніпуляції своїм умістом, для доступу до батьків і нащадкам.

13.4 Доступ до елементів DOM і навігація

Для маніпуляцій з DOM використовується об'єкт document. Використовуючи document, можна одержувати потрібний елемент дерева й міняти його зміст. Будь – який доступ і зміни DOM беруть свій початок від об'єкта document.

Існують наступні способи доступу до елементом DOM:

document.documentElement – Самий верхній тег. У випадку коректної Html – сторінки, це буде тег <html>;

document.body – тег <body>, якщо є у документі (повинен бути);

document.getElementsByTagName(“тег”) – доступ до елементу за ім’ям тегу;

document.getElementById(“ID”) – доступ до елемента по атрибуту id;

document.getElementByName(“ім'я”) – доступ до елемента по атрибуту name.

У моделі DOM найпростіший спосіб отримати доступ до елементу – звернутися до нього безпосередньо по його ідентифікатору id, скориставшись методом ***getElementById*** об'єкта document:

Приклад 1:

```
<html>
<head>
<title>Основи DOM</title>
</head>
<body>
<h1 id = "head">Основи DOM</h1>
<p>A Text</p>
<script type = "text/javascript">
var a = document.getElementById("head");
// зверніть увагу – скрипт пишеться після оброблюваного елемента
alert(a);
</script>
</body>
</html>
```

Метод elem.getElementsByTagName("tag") шукає всі елементи із заданим тегом tag усередині елемента elem і повертає їх у вигляді списку.

Приклад 2:

```
<table id="Table1" >
  <tr><td>Курс навчання:</td>
            <td><label><input type="radio" name="course"     value="first" checked>
  перший</label>
              <label><input type="radio" name="course" value="second"> другий </label>
                <label> <input type="radio" name="course" value="third"> третій </label>
                <label><input type="radio" name="course" value="fourth"> четвертий
      </label>
</td>  </tr> </table>
  <script>
  var tableElem = document.getElementById('Table1');
  var elements = tableElem.getElementsByTagName('input');
  for (var i=0; i<elements.length; i++) {
    var input = elements[i];
          alert(input.value + ': ' + input.checked);
              }
</script>
```

Примітка.

Тег <label> встановлює зв'язок між певною міткою, у якості якої звичайно виступає текст, і елементом форми (<input>, <select>, <textarea>). Такий зв'язок необхідний, щоб змінювати значення елементів форми при натисканні курсором миші на текст. Крім того, за

допомогою <label> можна встановлювати гарячі клавіші на клавіатурі й переходити на активний елемент подібно посиланням.

Існує два способи зв'язування об'єкта й мітки. Перший полягає у використанні ідентифікатора id усередині елемента форми й вказівці його імені як атрибута for тегу <label>. При другому способі елемент форми міститься усередину контейнера <label>.

Синтаксис

```
<input id="идентификатор"><label for="идентификатор">Текст</label>
<label><input type="..."> Текст</label>
```

Для навігації по дереву існую поняття дочірніх, батьківських елементів, а також «братів». Для навігації по дереву використовуються наступні правила:

1. Усі дочірні елементи, включаючи текстові, перебувають у масиві *childNodes[].*
2. Властивості *firstChild* і *lastChild* показують на перший і останній дочірні елементи й рівні null, якщо дітей немає.
3. Властивість *parentNode* Вказує на батька.
4. Властивості *previousSibling* і *nextSibling* вказують на лівого й правого братів вузла.

Властивості елементів DOM

Деякі властивості елементів, корисні при роботі з DOM:

1. *tagName* – містить ім'я тегу у верхньому регістрі, тільки для читання;
2. *style* – управляє стилем. Аналогічно встановленню стилю в CSS;
3. *innerHtml* – містить увесь Html – Код у середині вузла, і застосовується, в основному, для динамічної зміни змісту сторінки;
4. *className* – задає клас елемента, аналогічно html – атрибуту «class»;
5. *onclick, onkeypress, onfocus* та ін. - зберігають функції – оброблювачі відповідних подій.

У наступному прикладі 3., у циклі перебираються всі дочірні елементи елемента body і формується текстовий рядок, що містить послідовно номера елементів, імена їх тегів та Html – код вузла. З

результатів роботи програми видно, що текстові вузли (текст поза елементами HTML) не мають властивості innerHTML.

Приклад 3.

```
<html>
  <head>
  <script>
    function go() {
     var S = "";
     for(var i=0; i<document.body.childNodes.length; i++) {
       var child = document.body.childNodes[i];
       //alert(child.tagName);
           S=S+i+"-"+child.tagName+"="+child.innerHTML+"   ";
       }
      alert(S);
    }
    </script>
  </head>
  <body>
     <div id="dataKeeper">Заголовок</div>
     <ul>
        <li>Текст 1</li>
        <li>Текст 2 </li>
     </ul>
     <div id="footer">Конец документа</div>
     <input type="button" onclick="go()" value="Go"/>
     Текст
  </body>
</html>
```

13.5 Атрибути елементів DOM

Тег HTML (Dom – елемент) може мати будь-яку кількість атрибутів.

У наступному прикладі елемент div має атрибути id, class та нестандартний атрибут alpha:

```
<div id="Myelement" class="big" alpha="omega"></div>
```

Атрибути можна додавати, видаляти й змінювати. Для цього є спеціальні методи:

setattribute(name, value) – встановлює значення атрибута;
getattribute(name) – отримання значення атрибута;
hasattribute(name) – перевірити, чи є такий атрибут;
removeattribute(name) – вилучити атрибут.
Ім'я атрибута не залежить від регістру.

У деяких випадках між поняттям «властивість» і «атрибут» є зв'язок – браузер синхронізує значення *стандартних* властивостей з

атрибутами. Якщо змінюється атрибут, то змінюється й властивість із цим іменем, і навпаки. Така синхронізація гарантується для всіх основних стандартних атрибутів. При цьому атрибуту з менем class відповідає властивість className, тому що ключове слово class зарезервоване в javascript.

Наступний приклад показує ідентичність атрибута й властивості з іменем id:

Приклад 4.

```
<html>
   <head>
   <script>
     function go() {
       document.body.setAttribute('id', 'Newid'); //встановлюємо атрибут
       alert(document.body.id) //читаємо його значення як властивість
       }
      </script>
   </head>
   <body>
      <input type="button" onclick="go()" value="Go"/>
   </body>
</html>
```

13.6 Додавання й видалення елементів DOM

Щоб створити новий елемент – використовується метод document.createElement(тип). Для того, щоб елемент побачити на сторінці, його необхідно додати в дерево DOM. Це можна зробити методом appendChild, який в DOM є у будь – якого елемента.

У наступному прикладі, в кінець дерева DOM (у кінець сторінки) додається рядок тексту:

Приклад 5.

```
<html>
   <head>
   <script>
     function go() {
       var newDiv = document.createElement("div"); //створення елемента
       newDiv.innerHTML = "Привіт, мир!"; //встановлення властивостей нового
                                                    //елементу
       newDiv.style.backgroundColor = "red";
       document.body.appendChild(newDiv) //додавання нового елемента у дерево
      }
      </script>
   </head>
   <body>
      Текст
      <input type="button" onclick="go()" value="Go"/>
  </body>
</html>
```

Новий елемент можна додати не в кінець дітей, а перед потрібним елементом. Для цього використовується метод insertBefore батьківського елемента. Він працює так само, як і appendChild, але приймає другим параметром елемент, перед яким потрібно вставляти.

Приклад 6.

```
<html>
   <head>
   <script>
      function go() {
       var newdiv = document.createElement("div"); //створення нового елемента
       var olddiv = document.getElementById("id1"); //одержання існуючого елемента
       newdiv.innerHTML = "Привіт, мир!"; //Встановлення властивостей нового елемента
       newdiv.style.backgroundColor = "blue";
       document.body.insertBefore(newdiv,olddiv) //додавання нового елемента перед
                    існуючим
      }
       </script>
   </head>
   <body>
      <div id="id1">Текст</div>
      <input type="button" onclick="go()" value="Go"/>
  </body>
</html>
```

Щоб видалити вузол з документа - достатньо викликати метод removeChild з його батька. Якщо батько елемента невідомий, то його легко отримати за допомогою функції parentNode. Наступний приклад реалізує додавання елемента та його видалення двома способами.

Приклад 7.

```
<html>
   <head>
   <script>
      function add() {
       var newdiv = document.createElement("div"); //створення нового елемента
       newdiv.innerHTML = "Привіт, мир!"; //встановлення властивостей нового елементу
       newdiv.style.backgroundColor = "green";
       newdiv.id="id1";
       document.body.appendChild(newdiv) //додавання нового елемента
      }
      function del1() {
         var todel = document.getelEmentById("id1"); //одержання елемента для видалення
         document.body.removeChild(todel) //видалення елемента як нащадка body
      }
      function del2() {
         var todel = document.getElementById("id1"); //одержання елемента для видалення
         todel.parentnode.removeChild(todel) //видалення елемента, як нащадка свого батька
```

```
    }
    </script>
  </head>
  <body>
    Текст
    <input type="button" onclick="add()" value="Додати"/>
    <input type="button" onclick="del1()" value="Вилучити1"/>
    <input type="button" onclick="del2()" value="Вилучити2"/>
 </body>
</html>
```

У наступному прикладі продемонстроване додавання елемента маркірованого списку й видалення довільного елемента списку:

Приклад 8.

```
<html>
  <head>
  <script>
    function add(form) {
     var newdiv = document.createElement("li"); //створення нового елемента списку <li>
     newdiv.innerHTML = form.about.value+"<input type=\"button\" onclick=\"del(this)\"
                        value=\"Вилучити\"/>";
     newdiv.style.backgroundColor = "#3FD3A7";//встановлення властивостей нового
                        //елемента
     var list=document.getElementById("list1"); //одержання елемента-списку по id
     list.appendChild(newdiv) //додавання нового елемента
    }
    function del(todel) { //todel - переданий елемент <input>
       var todel_parent=todel.parentNode; //одержання дочірнього елемента - <li>, куди
                        //входить переданий <input>
       todel_parent.parentNode.removeChild(todel_parent)  //видалення елемента <li> як
                        //нащадка свого батька
    }
    </script>
  </head>
  <body>
    Список:
    <ul id="list1">
    </ul>
    <form>
    <input type="text" name="about">
    <input type="button" onclick="add(this.form)" value="Додати"/>
    </form>
  </body>
</html>
```

13.7 Робота з таблицями в DOM

Таблиця та її елементи (рядки, стовпці) також є об'єктами дерева DOM і підтримують усі методи роботи з ним. Крім того, в таблицях (об'єкт HTMLTableElement) є додаткові засоби навігації й доступу до елементів.

table.rows[] – список рядків таблиці;

tr.cells[] – список осередків рядка;

tr.rowindex – номер рядка;

td.cellindex – номер осередку в рядку.

Можна отримати кількість рядків таблиці та кількість осередків у рядку:

table.rows.length – кількість рядків таблиці;

tr.cells.length – кількість осередків у рядку .

Для додавання й видалення елементів таблиці можна скористатися функціями:

table.insertRow(індекс_рядка) – вставка рядка;

table.deleteRow(індекс_рядка) – видалення рядку;

tr.insertCell(індекс_рядка) – вставка осередку у рядок.

tr.deleteCell(індекс_осередку) – видалення осередку із рядку.

У наступному прикладі реалізований сценарій додавання й видалення даних у таблицю:

Приклад 9.

```
<html>
   <head>
   <script>
   function add(form) {
       table1 = document.getElementById('mytable'); //одержання таблиці за ім'ям
       row1 = table1.insertRow(table1.rows.length); //вставка рядка в таблицю
       cell1 = row1.insertCell(row1.cells.length); //вставка 1-го осередку у рядок
       cell1.innerHTML = row1.rowIndex;     //додавання номеру  рядку в перший осередок
       cell1 = row1.insertCell(row1.cells.length); //вставка 2-го осередку  в рядок
       if (form.info.value != "") cell1.innerHTML = form.info.value //вставка даних в осередок
       else cell1.innerHTML = "default";
       cell1 = row1.insertCell(row1.cells.length); //вставка 3-го осередку у рядок
       cell1.innerHTML = "<input type=\"checkbox\" name=\"check\">"; //додавання
                                                                    // "чекбоксу" в осередок
      }
   function del(form) {
       table1 = document.getElementById('mytable'); //одержання таблиці за ім'ям
       var i = table1.rows.length;                  //одержання кіл-сті рядків  таблиці
         while (i--) {                          //цикл перебору рядків
         var row1 = table1.rows[i]                  //одержання поточного рядка
         if (row1.cells[2].childnodes[0].checked){   //якщо відмічений "чекбокс" поточного
                                                     //рядку
            table1.deleteRow(row1.rowIndex);        //вилучити поточний рядок
            }
         }
      }
      </script>
```

```
  </head>
  <body>
    Таблиця:
    <table id="mytable" border="1">
    <tr>
      <th>Номер</th>
      <th>Значення</th>
      <th>Відзначити</th>
    </tr>
    <table>
    <br>
    <form>
    <input type="text" name="info">
    <input type="button" onclick="add(this.form)" value="Додати"/><br>
    <input type="button" onclick="del(this.form)" value="Вилучити відзначені"/>
    </form>
  </body>
</html>
```

13.7 Робота зі стилями за допомогою Javascript

13.7.1 Робота із класом елемента

Властивість classname відповідає Html – атрибуту class. За допомогою цієї властивості можна привласнити елементу будь-який клас:

```
document.body.className = ”myclass”;
```

Оскільки елемент може одночасно мати кілька класів, перерахованих через пробіл то можна додати до вже існуючих класів новий:

```
document.body.className += ” myclass”;
```

Пошук і видалення класу у списку класів, розділених пробілами, вимагає більш складного програмного коду, пов'язаного з розбором текстового рядка – властивості className.

Для того щоб вилучити всі класи елемента, досить привласнити властивості className порожнє значення:

```
document.body.className = ” ”;
```

13.7.2 Робота з CSS – властивостями

За допомогою властивості style можна змінювати більшість CSS – властивостей, наприклад:

```
element.style.color=”red”
```

Для властивостей, назви яких складаються із декількох слів, використовується такий запис:

```
background-color  => backgroundcolor
```

border-left-width => borderleftwidth

Для видалення CSS – властивості необхідно надати йому порожнє значення.

Існує властивість style.cssText, який дозволяє задати весь перелік css-властивостей елементу через крапку із комою, аналогічно тому, як це робиться при опису властивостей в css, за допомогою селекторів:

element.style.cssText="color: red !important; background-color: yellow;”

При встановленні style.cssText усі існуючі властивості style перезаписуються, тому рекомендується працювати з конкретними значеннями css-властивостей, а не із cssText.

Приклад 10. Реалізації вибору різних зовнішніх виглядів таблиці:

```
<html>
   <head>
   <style type="text/css">
      .mytable {text-align: center; font-family: Verdana; color: #D02090; background: #00FFFF;
                font-weight: bold; font-size: 120%}
   </style>
   <script>
   function setstyle(form) {
      table1 = document.getElementById('mytable');
      switch (form.set_style.value) {
            case "1": {
               table1.style.cssText="";    //Скидання css-властивостей
               table1.classname="mytable"  //Встановлення  класу
            }
            ; break;
            case "2": {
               table1.className=""         //Скидання класу
               table1.style.cssText="";    //Скидання css-властивостей
               table1.style.cssText="color: red; background-color: yellow;" //Встановлення
                  переліку css-атрибутів
               table1.style.width="60%" //Встановлення css-атрибута
            }
            ; break;
            case "3": {
               table1.setAttribute("border", "0") // Встановлення атрибута елемента table
            }
            ; break;
            case "4": {
               table1.setAttribute("border", "1") // Встановлення атрибута елемента table
            }
            ; break;
         }
   }
            </script>

            </head>
```

```
<body>
   Таблиця:
   <table id="mytable" border="1">
   <tr>
      <th>Стовпець1</th>
      <th>Стовпець2</th>
      <th>Стовпець3</th>
   </tr>
   <tr>
      <td>Стовпець1</td>
      <td>Стовпець2</td>
      <td>Стовпець3</td>
   </tr>
   <table>
   <br>
   Стиль таблиці:
   <form>
   <select name="set_style" size="1" onchange="setstyle(this.form)">
      <option value="0">
      <option value="1">Мій стиль
      <option value="2">Кольоровий
      <option value="3"> Без сітки
      <option value="4">Із сіткою
   </form>
</body>
</html>
```

Порядок виконання роботи.

1. Реалізувати та проаналізувати приклади 1 – 10, що наведені в лабораторній.
2. Розробити скрипт, що перебирає усі дочірні елементи вузлу HTML файлу «Титульний лист» (Лабораторна робота JS), та побудувати об’єктну модель документу.
3. Розробити скрипт, що дозволяє керувати елементами таблиці (видаляти та змінювати вміст осередків, добавляти рядки у таблицю) із переліком лабораторних робіт.
4. Привести початкові HTML – коди до файлів (приклади 1–10 інструкції до лабораторної роботи), із необхідними коментарями та поясненями.
5. Привести початкові коди до скриптів (п.п 2 – 3 порядку виконання роботи) з необхідними поясненнями.
6. Привести структурну схему об’єктної моделі документу (п. 2 порядку виконання роботи).

Примітка. Скрипти можна розташовувати у вигляді окремих файлів, або безпосередньо у HTML – файлах.

Лабораторна робота № 14 Створення динамічних гіпертекстових сторінок

Мета роботи: Навчитись створювати динамічні гіпертекстові сторінки за допомогою PHP

Теоретичні відомості

Файли з розширенням PHP є звичайними текстовими файлами, які можна створювати навіть в стандартній програмі Блокнот. Але набагато зручніше скористатись інтегрованим середовищем розробки, наприклад Macromedia Dreamweaver.

PHP-файл – це звичайний HTML-файл, в якому присутні теги <? ?> з кодом на PHP.

Приклад найпростішої програми на PHP:

```
<html>
<body>
<?
        echo 'abc';
?>
</body>
</html>
```

Для запуску сценарію в NuSphere PhpED натисніть Ctrl+F9, для запуску в режимі налагодження – F9. Під час налагодження для перегляду вмісту змінної достатньо навести на неї мишею.

Код, в тезі <? ?> буде переданий інтерпретатору, який поверне рядок abc і буде виведений у вікно переглядача. Для досягнення таких результатів, взагалі кажучи, використовувати PHP необов’язково, оскільки така програма завжди генерує однаковий код.

У мові PHP для пришвидшення інтерпретації назвам змінних завжди передує символ $. Змінні описувати не потрібно. При ініціалізації змінної, інтерпретатор сам визначить її тип, керуючись типом значення, яким вона ініціалізується. Під час виконання програми змінна може змінювати свій тип, може бути знищеною, а потім створеною знову.

14.1 Мова PHP має С-подібний синтаксис

Приклад розгалуження та 2-х способів виведення у вікно переглядача:

```
<?
   $a=5;
//  $a=6;

   if ($a%2)
   {
?>
      <p>a-непарне</p>
<?
      echo $a;
   }
   else
      echo "<p>a-парне</p>$a";
?>
```

Використання подвійних лапок для формування рядка, на відміну від одинарних, як в попередньому прикладі, дозволить створити рядок, у який буде підставлено значення змінної $a.

Функції в PHP не мають жорстко заданого типу. Одна й та ж функція з параметрами таких самих типів може повертати значення різних типів, в залежності від значень переданих параметрів та алгоритму її роботи. Наприклад, ціле число при успішному виконанні або NULL при помилці.

PHP (офіційна назва "PHP: Гіпертекстовий препроцесор") - це мова сценаріїв на стороні сервера, вбудована в HTML документ.
Для того щоб зрозуміти, що це означає, давайте розглянемо приклад простого сценарію:

```
<html>
   <head>
      <title>Приклад</title>
   </head>
   <body>

      <?php
      echo "Вітаю, це PHP-скріпт!";
      ?>

   </body>
</html>
```

Створіть цю HTML-сторінку та подивіться, як вона працює.

14.2 Запуск PHP-програм

Для запуску скрипта скопіюйте його до теки:

- C:\wampserver64\www\lab_14
- Запустіть wampserver:
- C:\wampserver64\wampserver64.exe
- Для запуску програми та перегляду результату її роботи відкрийте вікно переглядача тенет (наприклад Internet Explorer), введіть у рядок адреси: http://localhost/lab_14 та натисніть ENTER.

Зверніть увагу, що жоден програмний код не відображає теги HTML документа, сам гіпертекстовий документ містить невеликий вбудований код. Іншими словами, веб-майстер все ж може створювати гіпертекстові сторінки за допомогою звичайного редактора, а потім при необхідності вбудовувати елементи динамічної обробки - невеликі скрипти. У даному прикладі вся програма складається з одного рядка:

"Привіт, це PHP сценарій!";

Сервер "дізнається", що це програма яка виконується на спеціальних тегах:

<? php ... ? >

Різниця між PHP і клієнтськими мовами програмування (наприклад, JavaScript) полягає в тому, що код виконується на стороні сервера, тобто ще до того, як сторінка отримає браузер користувача.

Всі сторінки, що містять код, повинні мати розширення .HTML або .PHP,але якщо у вас є доступ до адміністрування сервера, виможете налаштувати препроцесор PHP, щоб всі сторінки, включаючи .HTML, перевірялися на вміст сценарію.

Порядок виконання роботи.

Виведення на екран зміних в *PHP.*

1. За допомогою текстового редактора Блокнота *створіть файл testphp1.php* і розмістить наступний код у верхній частині сторінки:

```
<html>
<head>
	<title>Вывод на экран и переменные в PHP</title>
	<?php
		echo "Привет, мир!";
	?>
	</head>
<body>
</body>
</html>
```

2. Створіть скрипт з використанням змінних кожного з цих типів:

```
<html>
<head>
<title>Вывод на экран и переменные в PHP</title>
</head>
<body>
<?php
$i = 6; // ціле
$d = 4.89; // дробне
$str = "PHP для усіх!"; // строка
echo ($i + $d);
echo "<br> Hello! ".$str;
?>
</body>
</html>
```

3. Зверніть увагу, що всі змінні в PHP повинні починатися з символу долара ($),тип змінної не встановлюється явно, *вона розраховується* препроцесором PHP в залежності від контексту. і значення змінної $str.
4. Підтримуються всі арифметичні операції та функції, багаторівневі дужки, логічні операції, операції збільшення або зменшення на одиницю тощо. Крім того, дуже просто і природно організувати порівняння, якщо - *то* - *інакше*.

```
if (вимога)
	{
	 Код для запуску N1
	}
Else
	{
	 Код для запуску N2
	}
```

Розглянемо простий приклад:

```
<html>
<head>
<title> Виведення на екран і змінних в PHP</title>
```

```
</head>
<body>
<?php
$i = 6; // ціле
$v = 7;
$d = 4.89; // дробне
$str = "PHP для усіх!"; // строка
echo ($i + $d);
echo "<br>Привіт, світ!".$str;

if($i == $v)
{
        echo $i + $v;
}
else
{
        echo $i.$v;
}
?>
</body>
</html>
```

5. При порівнянні з істиною застосовуються дві ознаки рівності, щоб інтерпретатор міг легко відрізнити порівняння від асигнування. Результат скрипту - *67, оскільки $i* не дорівнює *$v, а команда*echo *$i.$v;* об'єднує два рядки: *"6"* і *"7"*. Нерівність (брехня) позначаєтьсясимволами !, всіма іншими арифметичними і логічними символами і операторами (наприклад, або, і, >, < і т.д.).
6. PHP має засоби для швидкої зміни змінної на одиницю в напрямку збільшення або зменшення. Для цього потрібно вказати ім'я змінної і за нею, без ознак рівності - поспіль два плюси або мінуси відповідно. Наприклад, *$a.* Змінні *$a* будуть збільшені на один.
   Підтримка одночасного присвоєння одного значення декільком змінним - $a $b q
7. Введіть наступний код на *сторінці testphp1.php:*

```
<html>
<head>
<title>Виведення на екран змінних PHP</title>
</head>
<body>
<?php
$b = $a = 5;
echo "<br>змінна a=$a, b=$b";
$c = $a++;
echo "<br> змінна a=$a, c=$c";
$e = $d = ++$b;
echo "<br> змінна e=$e, d=$d, b=$b";
$g = 10;
$h = $g += 10;
echo "<br> змінна g=$g, h=$h";
?>
```

```
</body>
</html>
```

8. Операція присвоювання також дає свій *результат, тому операція $a =5 дала* результат *5,*тому змінна *b* була ініційована в 5.
   В операції *$c $a;* спочатку було зроблено присвоєння, а потім змінну *$a* збільшено на 1.
   В операції *$d $b;* спочатку $b збільшено на 1 змінну, а потім присвоюється отримане значення.

Передача параметрів за посиланням, передача параметрів з форми (GET і POST - запити).

9. За допомогою текстового редактора "Блокнот" (Notepad) створіть файл з розширенням *php* і помістіть наступний код:

```
<html>
<head>
<title> Виведення на екран параметрів що входять </title>
</head>
<body>
<?php
echo $_GET['message']."., ".$_GET['name']
?>
</body>
</html>
```

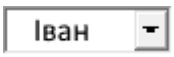
Зверніть увагу, що змінні message і name передаються в адресному рядку, відокремлюються від адреси сторінки знаком?, Між собою розділені амперсант (&). Змінити значення змінних прямо в адресному рядку, натиснути Enter і отримати інший результат роботи скрипта.

10. Створіть сторінку з розширенням html. Створіть форму:

```
<form name="myform" method="get" action="testphp2.php">
<br>Повідомлення:<br>
<input type="text" name="message">
<br>Имя:<br>
<input type="text" name="name">
<br><input type="submit" value=" ОБРОБИТИ ДАНІ ">
</form>
```

Перевірте працездатність скрипта. Зверніть увагу, що на сторінці з формою елементів призначені імена, відповідні іменам змінних в приймаючому скрипті. Спробуйте змінити метод GET на метод POST, тепер значення змінних не повинні передаватися у відкритому вигляді.

Замість елемента <input type = "text" name = "name"> визначте на сторінці меню, що випадає, таке, як наведено нижче:



11. Змініть сценарій, щоб користувач зміг вибрати ім'я зі списку, введіть привітання, *натисніть "Обробити дані"* і отримайте привітання для браного імені.
12. Змініть сторінку форми, щоб користувач оминав привітання, натиснувши кнопку залежно від *фіксації (перемикач):*

- Привіт
- Добрий день
- Радий Вас знову бачити
- Доброго вечора

Змініть сторінку форми, щоб користувач зміг вибрати фон сторінки (засіб вибору на ваш розсуд).

Динамічне формування сторінок.

Створіть 3 файли з іменами: *testphp3_inc1.html, testphp3_inc2.html* та *testphp3_inc3.html.*

13. Створіть *файл testphp3.php,*вставте в нього такий код:

```
<html>
<head>
<title>Динамічне формування сторінок</title>
</head>
<body bgcolor=silver>
<?php
$file = "";
if ($link == 1) { $file = "testphp3_inc1.html"; }
if ($link == 2) { $file = "testphp3_inc2.html"; }
if ($link == 3) { $file = "testphp3_inc3.html"; }
if ($file == "") { ?>
<h3> Будь ласка, виберіть вірш:</h3>
<a href="testphp3.php?link=1"> текст N1</a><br>
<a href="testphp3.php?link=2"> текст N2</a><br>
<a href="testphp3.php?link=3"> текст N3</a>
<? } else {
        include($file);
}
?>
</body>
</html>
```

14. Збережіть створені файли, потім запустіть файл testphp3.php.
15. При виборі посилання скрипту передається як параметр ідентифікатор сторінки, зміст якої має бути включено в результуючий файл.
16. Змініть скрипт таким чином, щоб тексти включалися в таблицю з жовтим фоном; посилання на сторінку, яка імпортована в файл зараз, була неактивна.
17. Створено масив $titles, який буде містити назви текстів. З цього масиву має формуватися вміст тега <title> ... </ title> в залежності від імпортованої сторінки.
18. Внизу сторінки помістіть Попереднє посилання, до змісту та наступне. Посилання Попереднє повинно вести до попереднього (за порядковим номером) вірша, посилання Наступне повинна вести до наступного (за порядковим номером) вірша. Посилання повинні зникати зі сторінки, якщо попередні/наступні вірші вичерпані. Посилання До змісту повинна приводити сторінку в первинний вигляд і зникати на сторінці змісту.
19. Перенесіть файли testphp3_inc1.html, testphp3_inc2.html і testphp3_inc3.html в папку includes і внесіть відповідні зміни у скрипт.

Работа с файлами.

20. Створіть файл testphp4_form.html, помістіть в нього форму, яка буде містити текстове поле *<textarea name = message cols = 10 rows = 4> </ textarea>,* поле редагування *<input type = text name = person>* і кнопку для відправки даних.
21. Налаштуйте форму таким чином, щоб дані прямували скрипту *testphp4.php.*

Створіть файл *testphp4.php*, помістіть в нього наступний код:

```
<html>
<head>
<title>Гостевая книга</title>
</head>
<body bgcolor=silver>
<?php
$filename = "messages.txt";
$fp = fopen($filename,"a");
if($fp)
{
 fputs($fp, $message." ".$person);
 fclose($fp);
}
include($filename);
?>
</body>
</html>
```

22. Створіть порожній файл *messages.txt* - в ньому будуть зберігатися всі повідомлення користувачів.
23. Запустіть файл *testphp4_form.html*, переконайтеся в працездатності скрипта.
24. Повинно відбуватися наступне: змінні *$ message і $* person приймаються скриптом *testphp4.php*, відкривається файл *messages.txt* (робиться це методом *fopen*, докладніше про нього читайте), поміщає в його кінець рядок, що складається з значень переданих змінних, розділених пропуском (робиться це методом *fputs*, докладніше про нього читайте).
25. Змініть роботу скрипта таким чином, щоб повідомлення і імена користувачів виводилися в відформатованому вигляді, наприклад, помістіть їх в осередки таблиці.
26. Змініть роботу скрипта таким чином, щоб форма була в файлі *testphp4.php*, іншими словами, видаліть файл *testphp4_form.html*.
27. На початок сторінки помістіть *guestbook.jpg*.

Лабораторна робота № 15 Змінні в PHP

Мета роботи: У цій лабораторній роботі ви познайомиьесь з такими об’єктами мови php, як змінні.

Теоретичні відомості

Із змінними ви зустрічались вже в інших мовах програмування. У PHP зміст поняття «змінна» не відрізняється від інших мов програмування.Імена змінних в PHP починаються з символу$(наприклад, $i, $I,$count), регістр при найменуванні змінних має важливе значення, тому $i та $I це дві різні змінні. Оператором присвоєння є символ =.

PHP підтримує наступні типи змінних:

- Ціле число (Integer);
- Подвійної точності із плаваючою комою (Double);
- Стрічка (String);
- Масив (Array);
- Об'єкт (Object);
- Pdfdoc (тільки, якщо допускається підтримка формату PDF);
- Pdfinfo (тільки, якщо допускається підтримка формату PDF);

Опис типу змінної не є обов’язковим і тому програмістом може бути і не визначений. Кожна змінна автоматично перетворюється в кожній з типів й різні функції використовують потрібний тип. Проте, існує декілька функцій, для яких важливий тип змінної. Для ініціалізації (визначення) змінної необхідно їй просто присвоїти значення.

```
<?php
$a = 5; //змушує змінну $a стати змінною типу Integer
$b = 5.0; //змушує змінну $b стати змінною типу Double
$c = "5"; //змушує змінну $c стати змінною типу String
?>
```

При використанні змінних типу string можна використовувати як подвійні так і одинарні лапки. Але між ними є різниця :

```
<?
$opilla='Оболонь';
$s='Пиво $opilla';
echo $s; //виводить Пиво $opilla echo '<br>';
$s="Пиво $opilla";
```

```
echo $s; // виводить Пиво Оболонь
?>
```

Змінна розглядається як масив, якщо до її імені додається значення індекса, взятого у квадратні дужки []. Наприклад

```
<?php
$a[0] = 10;
$a[1] = 14;
$a[2] = 8;
//Дістали масив $a з трьох елементів
?>
```

Індексом масиву може бути або ціле число(Integer)або стрічка(string), елементами масиву можуть бути значення будь-якого типу.

```
<?
$a[‘name’]=’shoues’;
$a[‘color’]=’red’;
$a[‘size’]=36;
?>
```

Масив, в якому індексами є стрічки називається асоціативним.

15.1 Операції

Операції, що визначені в мові php, описані в таблицях 1 та 2

15.1.1 Арифметичні операції

Арифметичні операції Таблиця 15.1

Приклад	Назва	Результат
$a + $b	Додавання	Сума $a і $b
$a - $b	Віднімання	Різниця $a і $b
$a * $b	Множення	Добуток $a і $b
$a / $b	Ділення	Частка від ділення $a на $b
$a % $b	Остача	Остача від цілочисельного ділення $a на $b

15.1.2 Логічні операції

Логічні операції Таблиця 15.2

Приклад	Назва	Результат
$a == $b	дорівнює	TRUE, якщо $a дорівнює $b.
$a === $b	ідентично	TRUE, якщо $a дорівнює $b і вони одного типу. (тільки в PHP 4)
$a != $b	не дорівнює	TRUE, якщо $a не дорівнює $b.
$a <> $b	не дорівнює	TRUE, якщо $a не дорівнює $b.
$a !== $b	не ідентично	TRUE, якщо $a не дорівнює $b або вони різних типів. (тільки в PHP 4 і більше)
$a < $b	менше	TRUE, якщо $a строго менше $b.
$a > $b	більше	TRUE, якщо $a строго більше $b.
$a <= $b	менше або дорівнює	TRUE, якщо $a менше або дорівнює $b.
$a >= $b	більше або дорівнює	TRUE,якщо$a більше або дорівнює $b.
! $a	Заперечення	TRUE, якщо $a не TRUE.
$a && $b	Логічне «і»	TRUE, якщо і $a, і$b TRUE.
$a || $b	Логічне «або»	TRUE, якщо $a або$b TRUE.

15.1.3 Стрічкові операції

Основними операціями із стрічками є:

конкатенація ('.') – повертає об'єднання із правого та лівого аргументів.

присвоєння ('.=') – приєднує правий аргумент до лівого аргументу.

```
<?
$a = "Hello ";
$b = $a."World!";//тепер $b містить "Hello World!"

$a = "Hello ";
$a .= "World!";        // тепер $a містить "Hello World!"
```

```
?>
Стрічку можна розглядати і як масив символів
<?
$a=’Hello’;
$c=$a[1];
// c буде мати значення ‘e’
?>
```

Нумерація символів, як і в масивах, починається з нуля.

15.2 Операції присвоєння

Крім базової операції присвоєння (=), існують "комбіновані операції" для всіх бінарних, арифметичних і стрічкових операцій, які дають змогу використати значення у виразі, а потім установити його значення в результат цього виразу. Наприклад:

```
$a = 3;
$a += 5; // установлює в $a 8,
//ніби ми сказали: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // встановлює в $b "Hello There!",
// аналогічно $b = $b . "There!";
```

15.3 Операції інкременту/декременту

PHP підтримує операції pre- і post-інкременту й декременту.

Операції pre- і post-інкременту Таблиця 15.3

Приклад	Назва	Ефект
++$a	Pre-increment	Збільшує $a на 1, потім повертає $a.
$a++	Post-	Повертає $a, потім збільшує $a на 1.
--$a	Pre-decrement	Зменшує $a на 1, потім повертає $a.
$a--	Post-	Повертає $a, потім зменшує $a на 1.

Наведемо приклад скрипту, який демонструє дію цих операцій:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Повинне бути 5: " . $a++ . "<br> ";
echo "Повинне бути 6: " . $a . "<br>";
echo "<h3>Preincrement</h3>";
```

```
$a = 5;
echo "Повинне бути 6: " . ++$a . "<br>";
 echo "Повинне бути 6: " . $a . "<br>";
echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Повинне бути 5: " . $a-- . "<br>";
echo "Повинне бути 4: " . $a . "<br>";
echo "<h3>Predecrement</h3>";
$a = 5;
echo "Повинне бути 4: " . --$a . "<br>"; echo "Повинне бути 4: " . $a . "<br>";
?>
```

15.4 Передача зовнішніх змінних

Передати зовнішні змінні вPHP-скрипт можна включивши їх в стрічку запиту при виклику скрипта в браузері, тоді вони автоматично будуть доступні всередині скрипту через глобальний асоціативний масив $_GET (див нижче).

Порядок виконання роботи.

1. Створіть файл test_param.php в якому напишить

```
<html>
<head>
<title> Параметри в PHP </title>
</head>
<body>

<?php
echo $_GET['var1']; echo “<br>”;
echo “          ”;
echo “<br>”;
echo $_GET['var2'];
?>
</body>
</html>
```

2. Збережить створений файл на сервері. У стрічці адреси браузера напишемо http://localhost/wampserver/test_param.php?var1=35&var2=RTFM
3. У стрічці запиту ми передали php-скрипту дві змінні var1 і var2, присвоївши їм значення відповідно35і“RTFM”і доступ до значень цих змінних здійснюється через глобальний масив $_GET з ключами var1 і var2. Зауважимо, що в стрічці адреси, список змінних відділяється від імені файла знаком ?, імена змінних пишуться без знака $,змінні

розділяються між собою знаком & (амперсант), а стрічки пишуться без лапок.

4. Написати php-скрипт який видає суму двох зовнішніх параметрів parA і parB. Проекспериментуйте із значеннями параметрів що передаються, наприклад надайте parA значения ’3 хлопці’, а parB значення ’6 дівчат’.

Яке значення матиме змінна $a після виконання наступного коду:

```
$a=2;
$a.=($b=('Привіт'.$a));
$a+=$b+$b;
```

Лабораторна робота № 16 Передача параметрів за допомогою форм

Мета роботи: Навчитися передавти параметри за допомогою форм

Теоретичні відомості

У лабораторній роботі №15 ми розглянули змінні в PHP, і розглядали спосіб передачі зовнішніх даних (змінних) в php-скрипт. Цей спосіб полягав у тому, що імена змінних і їхні значення вписувались безпосередньо в стрічку запиту. Як ви змогли переконатись, цей спосіб є досить незручним для користувача, тому розглянемо інший спосіб передачі параметрів – за допомогою форми.

Форма–складова HTML-документу, яка містить елементи графічного інтерфейсу користувача (поля введення, списки вибору, кнопки, тощо) і призначена для відправки введених користувачем даних для опрацювання на Web-сервері.

Опис форми розпочинається тегом *<form>*і закінчується тегом *</form>.* Параметри цього тега, та теги, які описують елементи керування (поля введення, списки вибору, кнопки і т.д.), описані на сайті https://www.php.net.

Приклад: створити форму для реєстрації користувача чату, яка містить дані(ім'я,прізвище,адресу електронної пошти,псевдоніму користувача(нік), пароль та поле для повторного вводу паролю).

Створимо файл *register.html* в якому напишемо наступний код

```
<html>
<head>
<title> Реєстрація</title>
</head>
<body>
<!--тут починається опис форми-->
<form action="adduser.php" >
Ім'я <input type="text" name="name" maxlength="20" size="25"><br>
Прізвище <input type="text" name="surname" maxlength="20" size="25"><br>
Адреса e-mail<input type="text" name="mail" maxlength="20" size="25"><br>
Пседонім<input type="text" name="nik" maxlength="20" size="25" ><br> Пароль<input type="password" name="password" maxlength="20" size="25" ><br>
```

```
Пароль ще раз<input type="password" name="confirm_pass" maxlength="20"
size="25"><br>
<input type="submit" value="Зареєструватися">
</form>
<!--кінець форми-->
</body>
</html>
```

Після відкриття даної сторінки в браузері , отримаємо форму яка зображена на Рис. 1

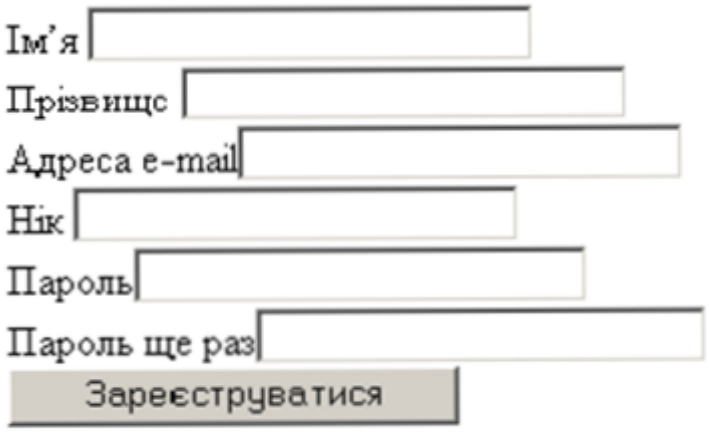


Рис. 16.1 Форма що була отримана

Оскільки ми ще не написали скрипта adduser.php, який обробляє дані, які користувач ввів в форму, то при натискуванні на кнопку «зареєструватися» нічого не відбудеться.

Нижче наведено один з варіантів скрипта adduser.php

```
<?
echo 'Ви ввели такі дані про себе:'.'<br>';
echo "<b>Ім/'я:</b>".$_GET['name']."<br>";
echo "<b>Прізвище:</b>".$_GET['surname']."<br>";
echo "<b>Адреса e-mail</b>".$_GET['mail']."<br>";
echo "<b>Нік:</b>".$_GET['nik']."<br>";
?>
```

Зауважимо, що в даному прикладі файл з формою register.html і файл із скриптом adduser.php мають лежати на сервері в одному каталозі.

Оскільки ми створили скрипт для обробки даних з форми, то тепер при натискуванні на кнопку «зареєструватися» виконається скрипт з файлу adduser.php (той файл який вказаний в параметрі action форми).

У цьому файлі дані з форми будуть доступними через глобальний асоціативний масив $_GET з ключами, рівними значенням, які вказані в атрибуті name при заданні тих чи інших елементів форми.

Зверніть увагу, що при передачі змінних всі поля передаються за допомогою методу GET , в тому числі і поле паролю, передаються в відкритому вигляді. Цього недоліку можна уникнути, змінивши метод передачі даних формою з GET (він використовується за замовчуванням) на POST. Тоді доступ до переданих значень форми буде здійснюватись через глобальний масив $_POST з відповідними ключами.

```
<form action="adduser.php" method="post" >
```

Форми можуть використовуватись також для завантаження файлів на сервер. Наведемо найпростіший приклад форми для завантаження файлів.

```
<form enctype="multipart/form-data"
 action="savefile.php"method="post">
        <input type="hidden" name="MAX_FILE_SIZE" value="1000">Send this file: <input
        name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

Приховане поле MAX_FILE_SIZE має бути описаним перед полем вводу файлу, і його значення відповідає максимальному розміру файлу, який передається. Значення вказується в байтах.

Завантажений файл буде збережено на сервері в тимчасовому каталозі, а в скрипті savefile.php для доступу до цього файлу будуть визначені наступні змінні:

$_FILES['userfile']['tmp_name'] – тимчасове ім'я файлу, під яким завантажений файл був збережений на сервері.

$_FILES['userfile']['name'] –оригінальне ім'я або шлях до файлу на системі відправника.

$_FILES['userfile']['size'] – розмір завантаженого файлу в байтах.

$_FILES['userfile']['type']–mime-тип файлу,якщо браузер надав

цю інформацію.

Зауважимо, що вся інформація про завантажений на сервер файл міститься в глобальному масиві $_FILES. Цей масив з'явився починаючи з PHP 4.1.0. Ключем цього масиву для доступу до завантаженого файлу на сервер є 'userfile' – це ім'я відповідного елемента форми, за допомогою якого було зроблене це завантаження.

Наведемо приклад скрипта, який опрацьовує дані отримані з вищенаведеної форми

```
<?
echo '<b> Файл був збережений на сервері під іменем </b>'.
$_FILES['userfile']['tmp_name'].'<br>';
echo '<b>У вас на машині він називався : </b>'. $_FILES['userfile']['name'].'<br>';
echo '<b>Розмір завантаженого файлу '. $_FILES['userfile']['size'].'<br>';echo
'<b>Mime тип файлу: '. $_FILES['userfile']['type'].'<br>';
?>
```

Для переміщення файлів, що завантажуються на сервер з тимчасового каталогу зручно використовувати функцію move_uploaded_file. Опис цієї функції наведено на сайті https://www.php.net/manual/ru/index.php.

Модифікуємо вищенаведений скрипт, щоб завантажені файли переміщувались з тимчасового каталогу в каталог images, який розміщений у тому ж каталозі, що і скрипт. Файл збережемо під таким же іменем як і на комп’ютері користувача.

```
<?
echo '<b>Файл був збережений на сервері під іменем </b>'.
$_FILES['userfile']['tmp_name'].'<br>';
        echo '<b>У вас на машині він називався : </b>'.  $_FILES['userfile']['name'].'<br>';
echo '<b>Розмір завантаженого файлу '.$_FILES['userfile']['size'].'<br>';
        echo '<b>Mime тип файлу: '. $_FILES['userfile']['type'].'<br>';
        move_uploaded_file($_FILES['userfile']['tmp_name'],"images/".$_FILES['userfile']['
        name']);
?>
```

Порядок виконання роботи

1. Розробити форму для завантаження на сервер електронного підручника з наступною обробкою його даних. Для цього використати наступні поля: автор книги, назва книги, видавництво, кількість сторінок, формат файлу, шлях до файлу.

2. Розробити форму для реєстрації користувача з наступними даними: псевдонім, пароль, підтвердження пароля, ім’я, прізвище, адреса e-mail, країна (вибір із списку), область, місто(село), вулиця.

3. Розробити форму для завантаження на сервер статей. Для цього використати наступні поля: автор статті, опис статті , журнал, у якому була надрукована стаття, кількість сторінок, рік видання, формат файлу, шлях до файлу.

Список літератури

1. Пасічник О.Г. Основи веб-дизайну /О.Г.Пасічник, О.В.Пасічник, І.В.Стеценко: [Навч.посіб.].-К.:Вид. група BHV.- 2009 .- 336 с.: іл.

Web-ресурси

2. Довідник по HTML https://css.in.ua/html/tags
3. Довідник по CSS https://css.in.ua/css/properties
4. Довідник по JavaScript https://w3schoolsua.github.io/jsref/index.html
5. Довідник по PHP https://www.php.net/docs.php

ДОДАТОК А

(довідковий)

Список тегів HTML

Шаблон

```
<html>
<head>
<title></title>
    Мета теги
    CSS
    Javascript
</head>
<body>
   Контент
</body>
</html>
```

CSS Media

all
handheld
print
projection
screen

Meta Types

http-equiv
name

Події

onLoad
onBrul
onChange
onFocus
onReset
onSelect
onSubmit
onKeydown
onKeypress
onKeyup
onClick
onDblclick
onMousedown
onMousemove
onMouseout
onMouseover
onMouseup

Символи

	пробіл
&quot;	"
&amp;	&
&lt;	<
&gt;	>
&at;	@
&euro;	€
&bull;	•
&trade;	™
&pound;	?
&copy;	©
&apos;	‘
&frac14;	?
&frac12;	?
&para;	¶
&sum;	Σ
&asymp;	≈
&plusmn;	±
&forall;	∀
&exist;	∃
&ordm;	◦

Синтаксис

Основний
HTML: <tag></tagclose> або <tag>
XHTML:<tag></tagclose> або <tag />
З атрибутами
HTML:<tag attribute="?">
XHTML:<tag attribute="?" />

Загальні

<html>	Зовнішній тег HTML сторінки
<head>	Частина документа, що не відображується на сторінці
<title>	Заголовок документа в рядку браузера
<body>	Видима частина сторінки

Гіперпосилання

<img src="URL">	Відображення картинки
<a href="#?">	Посилання на якір у поточній сторінці
<a href="URL">	Посилання на іншу сторінку
<a href="URL#">	Посилання на якір на іншій сторінці
<a href="mailto:email">	Посилання на e-mail

Форматування

 	Перехід на новий рядок
<code>	Програмний код
	Виділення блоку
, <I>	Акцентування шрифту
<h1>..<h6>	Заголовки, від великого до дрібного
<hr>	Горизонтальна лінія
<p>	Абзац
<pre>	Форматування із збереженням пробілів
	Виділення всередені сторінки
, <b>	Жирний текст
<sub>	Нижній індекс
<sup>	Верхній індекс

Фрейми

<frame>	Одиночний фрейм
<frameset>	Структура декількох фреймів
<iframe>	Фрейм всередені документу

Приклад фрейма

```
<frameset rows="80,*"  cols="*">
   <frame src="top.html" name="topFrame"
          scrolling="no" noresize>
   <frameset cols="80,*">
     <frame src="left.html" name="leftFrame"
            scrolling="no" noresize>
     <frame src="main.html" name="MFrame">
   </frameset>
</frameset>
```

Параметри тега <form>

action	Адреса CGI-обробника
enctype	MIME-тип інформації форми
method	Метод протоколу HTTP
name	Ім'я форми
target="_blank|_self|_parent|_top"	

Заголовок

<link rel="stylesheet" href="?" type="text/css">	посилання на зовнішній файл CSS
<script language="Javascript" type="text/javascript">	Зовнішній файл javascript
<meta http-equiv="content-type" content="?"; charset="?">	Мета інфор-мація

Таблиці

<caption>	Заголовок таблиці
<table>	Таблиця
<tbody>	Тіло таблиці
<td>	Комірка таблиці
<td colspan="?">	Число об'єднаних стовпців
<td rowspan="?">	Число об'єднаних рядків
<tfoot>	Підвал таблиці
<th>	Комірки шапки таблиці
<th colspan="?">	Число об'єднаних комірок в шапці
<thead>	Шапка таблиці
<tr>	Рядок таблиці

Списки

	Нумерований список
	Маркерованний список
	Елемент списку
<dl>	Список термінів
<dt>	Терміни
<dd>	Визначення терміна

```
<ol type="A|a|I|i|1">..</ol>
<ul type="disc|circle|square">...</ul>
```

Форми

<form>	Форма даних
<fieldset>	Групування елементів
<input type="?">	Елемент введення
<option>	Рядок в випадаючому списку
<select>	Випадаючий список
<textarea>	Багаторядкове поле введення

Карти зображень

```
<img usemap="?"/>Зображення карта
<map name="?">   Карта зображень
<area href="URL" shape="?" coords="?,?">
  Активна область зображення
```

Параметри тега <area>

alt	Альтернативний текст для області зображення
coords	Координати активної області
href	Задає адресу документа, на який слід перейти
nohref	Область без посилання на інший документ
shape=	"circle|poly|rect" Форма області
target	Ім'я вікна або фрейма, в який браузер завантажить документ

Таблиці

```
<table>

  <thead>
   <tr>
     <th>
     </th>
   </tr>
  </thead>

  <tbody>
    <tr>
      <td>
      </td>
    </tr>
  </tbody>

  <tfoot>
    <tr>
      <td>
      </td>
    </tr>
  </tfoot>

</table>
```

Input types

button
checkbox
file
hidden
image
password
radio
reset
submit
text

Комент.

```
<!--
Текст
коментаря
--!>
```

Об'єкт

```
<object>
    Об'єкт
<param />
    Параметр
```

Списки

```
<ol>
 <li>... </li>
</ol>

<ul>
 <li>...</li>
</ul>

<dl>
 <dt>Термін</dt>
 <dd>Визначення
    терміна </dd>
</dl>
```

ДОДАТОК Б

(довідковий)

Розміри

В CSS існують наступні міри довжини:

em - еквівалентні обчисленому значенню властивості 'font-size' того елемента, в якому воно використовується. Виключенням є той випадок, коли 'em' з'являється в самому значенні властивості 'font-size', тоді воно відноситься до розміру шрифту батьківського елемента. Може бути використано для вертикального або горизонтального виміру.

ex - визначаються 'x-height' шрифту. x-height називається так, тому що вона часто дорівнює висоті "x" нижнього регістра. Однак 'ex' визначені навіть для тих шрифтів, які не містять "x".

px - пікселі відносні до дозволу пристрою перегляду, тобто найчастіше - дисплея комп'ютера. Якщо щільність пікселів пристрою виведення сильно відрізняється від щільності типового комп'ютерного дисплея, ПА повинен перемасштабувати піксельні значення. Рекомендується, щоб піксель як точка відліку був візуальним кутом одного пікселя на пристрої із щільністю пікселів 90dpi на відстані витягнутої руки від читача. При нормальній довжині руки 28 дюймів візуальний кут буде приблизно 0.0227 градусів.

in - дюйми - 1 дюйм дорівнює 2.54 сантиметри.

cm - сантиметри

mm - міліметри

pt - пункт, використовуваний в CSS2, дорівнює 1/72 дюйма.

pc - піки. 1 піка дорівнює 12 пунктам.

```
div { margin: 0.5em } /* em */
span { margin: 1ex } /* ex */
P { font-size: 12px } /* px */
H1 { margin: 0.5in } /* дюйми */
H2 { line-height: 3cm } /* сантиметри */
H3 { word-spacing: 4mm } /* міліметри */
H4 { font-size: 12pt } /* пункти */
H4 { font-size: 1pc } /* піки */
```

ДОДАТОК В

(довідковий)

Кольори

Кольори це або визначене слово, або числова специфікація RGB.

Список назв кольорів: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white й yellow. Ці 16 кольорів визначені в HTML 5.0. Користувач може додатково специфікувати ключові слова, що відповідають кольорам певних об'єктів в середовищі користувача.

```
p {color: red}
```

Формат значення RGB у 16-річному запису - '#' за яким відразу йдуть три або шість 16-річних символів. Трисимвольний запис RGB (#rgb) конвертується в шестисимвольну форму (#rrggbb) шляхом дублювання цифр, але не доповненням нулями. Наприклад, #fb0 розширюється до #ffbb00. Це гарантує, що білий (#ffffff) можна специфікувати скороченим записом (#fff) і видалити залежність від глибини кольорів на дисплеї.

```
EM { color: #ff0000 }
```

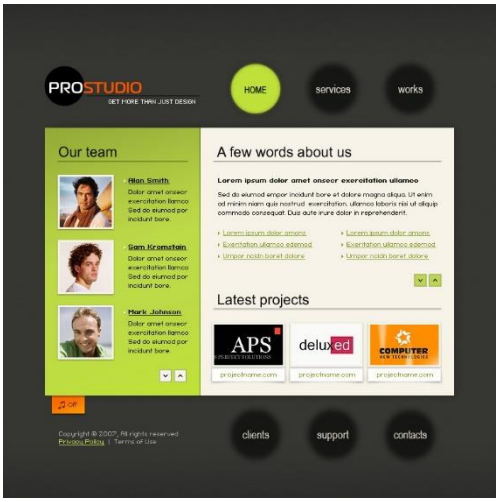
Формат значення RGB у функціональному записі - 'rgb(' за яким іде список розділених комами трьох числових значень (або трьох цілих, або трьох процентних) з наступної ')'. Ціле значення 255 відповідає 100% й F або FF в 16-річному запису: rgb(255,255,255) = rgb(100%,100%,100%) = #FFF.

```
EM { color: rgb(255,0,0) }
```

```
EM { color: rgb(100%, 0%, 0%) }
```

Варіанти завдань для самостійної роботи

Варіант 01. Весільний салон



Варіант 02. Професійна рекламна студія

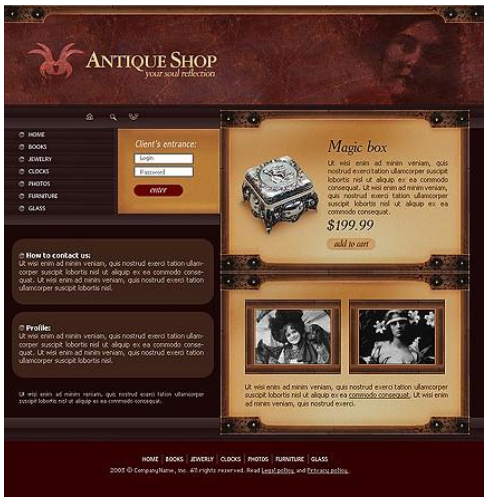
Варіант 03. Ресторан

Варіант 04. Будівельна компанія

Варіант 05. Боулінг клуб

Варіант 06. Зоопарк

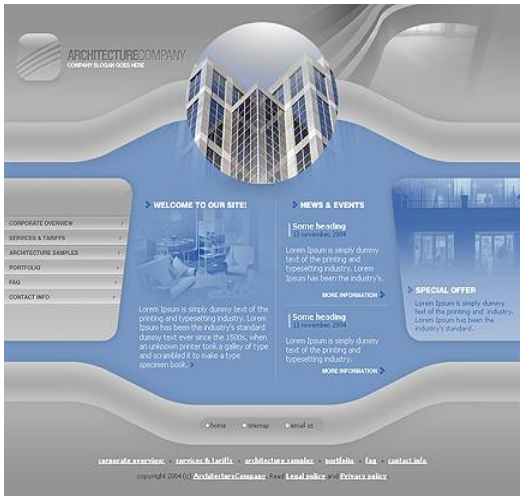
Варіант 07. Ювелірний магазин



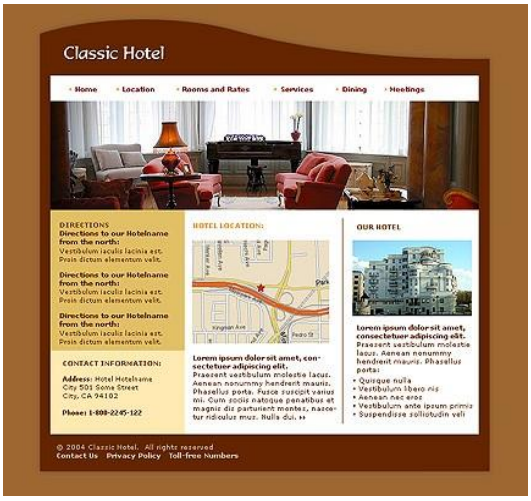
Варіант 08. Магазин антикваріату

Варіант 09. Футбольний клуб

Варіант 10. Інтернет розробники



Варіант 11. Архітектурна компанія



Варіант 12. Гостиниця

Варіант 13. Забудова котеджів

Варіант 14. Бізнес компанія

Варіант 15. Кінокомпанія

Варіант 16. Магазин одягу

Варіант 17. Портфоліо фотографа

Варіант 18. Дизайн студія

ДОДАТОК Д

ПРИКЛАД ОФОРМЛЕННЯ ТИТУЛЬНОГО ЛИСТА

ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«МЕТІНВЕСТ ПОЛІТЕХНІКА»

Кафедра цифрових технологій та проектно-аналітичних рішень

ЗВІТ

про виконання лабораторних робіт

з дисципліни

«WEB- дизайн»

Лабораторна робота №__

Виконав:

студент групи _____

_____(ПІБ)_____

Прийняв:

к.т.н., доцент

Кабак Л.В.

Запоріжжя 20__