МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Ю.О. Каліберда, І. В. Стовпченко

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

«ВЕБ-технології та ВЕБ-дизайн»

для студентів напряму 050101 – “Комп’ютерні науки”

Дніпропетровськ НМетАУ - 2019

Лабораторна робота №1

Основи HTML

HTML (HyperText Markup Language, мова розмітки гіпертексту) - це система верстки, яка визначає, як і які елементи повинні розташовуватися на веб-сторінці. Інформація на сайті, спосіб її подання та оформлення залежать виключно від розробника і тих цілей, які він перед собою ставить.

Тег **<html>** є контейнером, який містить в собі весь вміст веб-сторінки, включаючи теги **<head>** і **<body>**. Відкриваючий та закриваючий теги **<html>** в документі необов'язкові, але хороший стиль диктує неодмінне їх використання. Як правило, тег **<html>** йде в документі другим, після визначення типу документа (**Document Type Definition**, **DTD**), встановлюваного через елемент **<!DOCTYPE>**. Закриваючий тег **<html>** повинен завжди стояти в документі останнім.

Тег **<head>** призначений для зберігання інших елементів, мета яких - допомогти браузеру в роботі з даними. Також всередині контейнера **<head>** знаходяться метатеги, які використовуються для зберігання інформації призначеною для браузерів і пошукових систем. Наприклад, механізми пошукових систем звертаються до метатегах для отримання опису сайту, ключових слів та інших даних.

Вміст тега **<head>** не відображається безпосередньо на веб-сторінці, за винятком тега **<title>** встановлює заголовок вікна веб-сторінки.

<meta> визначає метатеги, які використовуються для зберігання інформації призначеною для браузерів і пошукових систем. Наприклад, механізми пошукових систем звертаються до метатегах для отримання опису сайту, ключових слів та інших даних. Дозволяється використовувати більш ніж один метатег, всі вони розміщуються в контейнері **<head>**.

Завдання:

1) Створити звичайну HTML сторінку

а) Створити файл index.html та додати у нього стандартну структуру HTML документу:

```
<!DOCTYPE html>
<html>
   <head>
   </head>
   <body>
   </body>
</html>
```

б) Зберегти файл у кодуванні UTF-8 та відповідно до цього за допомогою тегу `<meta>` вказати тип кодування для браузера:

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

в) Додати до сторінки титул та заголовок за допомогою тегів `<title>` та `<h1>` у відповідних розділах сторінки із змістом: "Лабораторна робота №1" після чого відкрити документ у браузері.

2) Створення переходу за посиланням.

а) Створіть файл `page2.html` та скопіювати до нього вміст першого документу.

б) Створити у обох документах посилання на інший:

```
1 <a href="page2.html">посилання</a>
2 <a href="index.html">посилання</a>
```

3) Створення переходу за якорем

а) У документі **index.html** додати багато будь-якого тексту, щоб сторінка займала більше ніж 1 екран. (щоб з'явилася полоса прокрутки) (**тег <p> для тексту**)

б) Створити якір - тег **<a>**, якому необхідно надати ім'я (**name="anchor"**)

в) У кінці документа додати посилання на якір у вигляді:

```
<a href="#anchor">ДО ЯКОРЯ</a>
```

Додатково може буди використано посилання на стандартний якір “top”, що відповідає початку сторінки.

Лабораторна робота №2
Поглиблення HTML

2.1 Списки

Списком називається взаємопов'язаний набір окремих фраз або пропозицій, які починаються з маркера або цифри. Списки надають можливість упорядкувати і систематизувати різні дані і представити їх у наочному і зручному для користувача вигляді.

Тег **<ol>** встановлює нумерований список, тобто кожен елемент списку починається з числа або букви і збільшується по наростаючій.

Тег **<ul>** становлює маркований список, кожен елемент якого починається з невеликого символу - маркера.

```
<ul type="disc | circle | square">...</ul>
```

Тег **<li>** визначає окремий елемент списку. Зовнішній тег **<ul>** або **<ol>** встановлює тип списку - маркований або нумерований.

```
<ul>
    <li>Чебурашка</li>
    <li>Крокодил Гена</li>
    <li>Шапокляк</li>
</ul>
```

2.2. Таблиці

Елемент **<table>** служить контейнером для елементів, що визначають вміст таблиці. Будь-яка таблиця складається з рядків і осередків, які задаються за допомогою тегів **<tr>** і **<td>**. Усередині **<table>** допустимо використовувати наступні елементи: **<caption>**, **<col>**, **<colgroup>**, **<tbody>**, **<td>**, **<tfoot>**, **<th>**, **<thead>** і **<tr>**.

Таблиці з невидимим кордоном довгий час використовувалися для верстки веб-сторінок, дозволяючи розділяти документ на модульні блоки. Подібний спосіб застосування таблиць знайшов втілення на багатьох сайтах, поки йому на зміну не прийшов більш сучасний спосіб верстки за допомогою шарів.

Тег **<table>** має наступні основні атрибути: **align** (визначає вирівнювання таблиці), **background** (задає фоновий малюнок в таблиці), **bgcolor** (колір фону таблиці), **border** (товщина рамки в пікселах), **bordercolor** (колір рамки), **cellpadding** (відступ від рамки до вмісту комірки), **cellspacing** (відстань між осередками), **cols** (число колонок в таблиці), **frame** (повідомляє браузеру, як відображати межі навколо таблиці), **height** (висота таблиці), **width** (ширина таблиці).

2.3 Фрейми

Фрейми розділяють вікно браузера на окремі області, розташовані поруч один з одним. У кожну з таких областей завантажується самостійна веб-сторінка. Оскільки навколо фреймів існує багато розмов про їх необхідність, далі наведемо достоїнства і недоліки фреймів, щоб можна було самостійно вирішити чи варто їх використовувати на своєму сайті.

При використанні фреймів необхідно як мінімум три HTML-файли: перший визначає фреймову структуру і ділить вікно браузера на дві частини, а інші два документа завантажуються в задані вікна. Кількість фреймів не

обов'язково дорівнює двом, може бути і більше, але ніяк не менше двох, інакше взагалі втрачається сенс застосування фреймів.

У разі використання фреймів у першому рядку коду пишеться наступний тип документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

Тег **<frameset>** визначає структуру фреймів на веб-сторінці. Фрейми розділяють вікно браузера на окремі області, розташовані впритул один до одного. У кожну з таких областей завантажується самостійна веб-сторінка визначається за допомогою тега **<frame>**.

Приклад коду:

```
<frameset cols="180,*">
    <frame src="sidebar.html" name="MENU">
    <frame src="content.html" name="CONTENT">
</frameset>
```

Тег **<iframe>** дозволяє додавати плаваючий фрейм до поточного html документу із вмістом іншої сторінки. Тег **<iframe>** підтримується специфікацією HTML5, на відміну від **<frameset>**, тому його використання доцільніше.

```
<iframe src="http://www.w3schools.com"></iframe>
```

Завдання:

1. Створіть нумерований список групи (5-6 студентів)

2. Створіть ненумерований список будь-яких речей із маркером у вигляді квадрата.

3. Створіть таблицю множення (Таблицю Піфагора до 5), за допомогою тега **Table**.

4. Створіть міні-сайт такого вигляду:

а) основна сторінка поділяється на 3 частини (Шапка сайту, зліва блок меню, та основна - змістовна частина), за допомогою **frameset**.

б) шапка сайту, та меню зробити в окремих файлах.

в) за допомогою меню сайту зробити так, шоб при натисненні посилань меню змінювалася змістовна частина (зміст - 1,2,3 пункти цієї лабораторної).

5.Створіть сторінку, подібну до пункту 4, але використовуючи тег **iframe**.

Лабораторна робота 3

Основи CSS

3.1 Каскадні таблиці стилів (CSS)

Стилем або **CSS** (**Cascading Style Sheets**, каскадні таблиці стилів) називається набір параметрів форматування, який застосовується до елементів документа, щоб змінити їх зовнішній вигляд. Перевагою стилів є те, що вони пропонують набагато більше можливостей для форматування, ніж звичайний HTML. Стилі являють собою набір параметрів, керуючих видом і положенням елементів веб-сторінки.

Основні способи додавання стилів до сторінки:

- Пов'язані стилі

При використанні пов'язаних стилів опис селекторів і їх значень розташовується в окремому файлі, як правило, з розширенням css, а для зв'язування документа з цим файлом застосовується тег <link>. Даний тег поміщається в контейнер <head>. Значення атрибута тега <link> - rel залишається незмінним незалежно від коду, як наведено в даному прикладі.

Значення href задає шлях до CSS-файлу, він може бути заданий як відносно, так і абсолютно. Зауважте, що таким чином можна підключати таблицю стилів, яка знаходиться на іншому сайті. Приклад:

```
<head>

  <title>Vanilla JS Note App</title>

  <link rel="stylesheet" href="css/style.css">

</head>
```

- Глобальні стилі

При використанні глобальних стилів властивості CSS описуються в самому документі і розташовуються в заголовку веб-сторінки. За своєю гнучкості і можливостям цей спосіб додавання стилю поступається попередньому, але також дозволяє зберігати стилі в одному місці, в даному випадку прямо на тій же сторінці за допомогою контейнера <style>. Приклад:

```
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
  <style>
   H1 {
    font-size: 120%;
    font-family: Helvetica, sans-serif;
    color: #333366;
   }
  </style>
 </head>
```

- Внутрішні стилі

Внутрішній або вбудований стиль є по суті розширенням для одиночного тега використовуваного на поточній веб-сторінці. Для визначення стилю використовується атрибут style, а його значенням виступає набір стильових правил. Приклад:

```
<body>
  <p style="font-size: 120%; font-family: monospace; color: #333366">Текст знаходиться тут</p>
</body>
```

Всі описані методи використання CSS можуть застосовуватися як самостійно, так і в поєднанні один з одним. У цьому випадку необхідно пам'ятати про їх ієрархії. Першим має пріоритет внутрішній стиль, потім глобальний стиль і в останню чергу пов'язаний стиль.

Завдання:

1. Додайте до документу із першої лабораторної роботи файл стилів CSS.

2. Змініть колір фону документу, та колір написів.

3. Змініть шрифт тегу <h1> на Arial.

4. Додайте до посилань клас “mybutton” та за допомогою нього оформіть посилання у вигляді кнопок.

5. Додайте до класу “mybutton” зміну кольору при наведенні.

6. За допомогою додаткових тегів додайте до документу рамку.

7. Зробіть кнопку із написом “приз”, яка буде зникати при наведенні. (щоб кнопка не мерехтіла треба “обгорнути” тег кнопки, ще одним, прозорим тегом із визначеним розміром, так застосовувати зникання кнопки при наведення на зовнішній тег)

Лабораторна робота 4

Поглиблення CSS

Завдання:

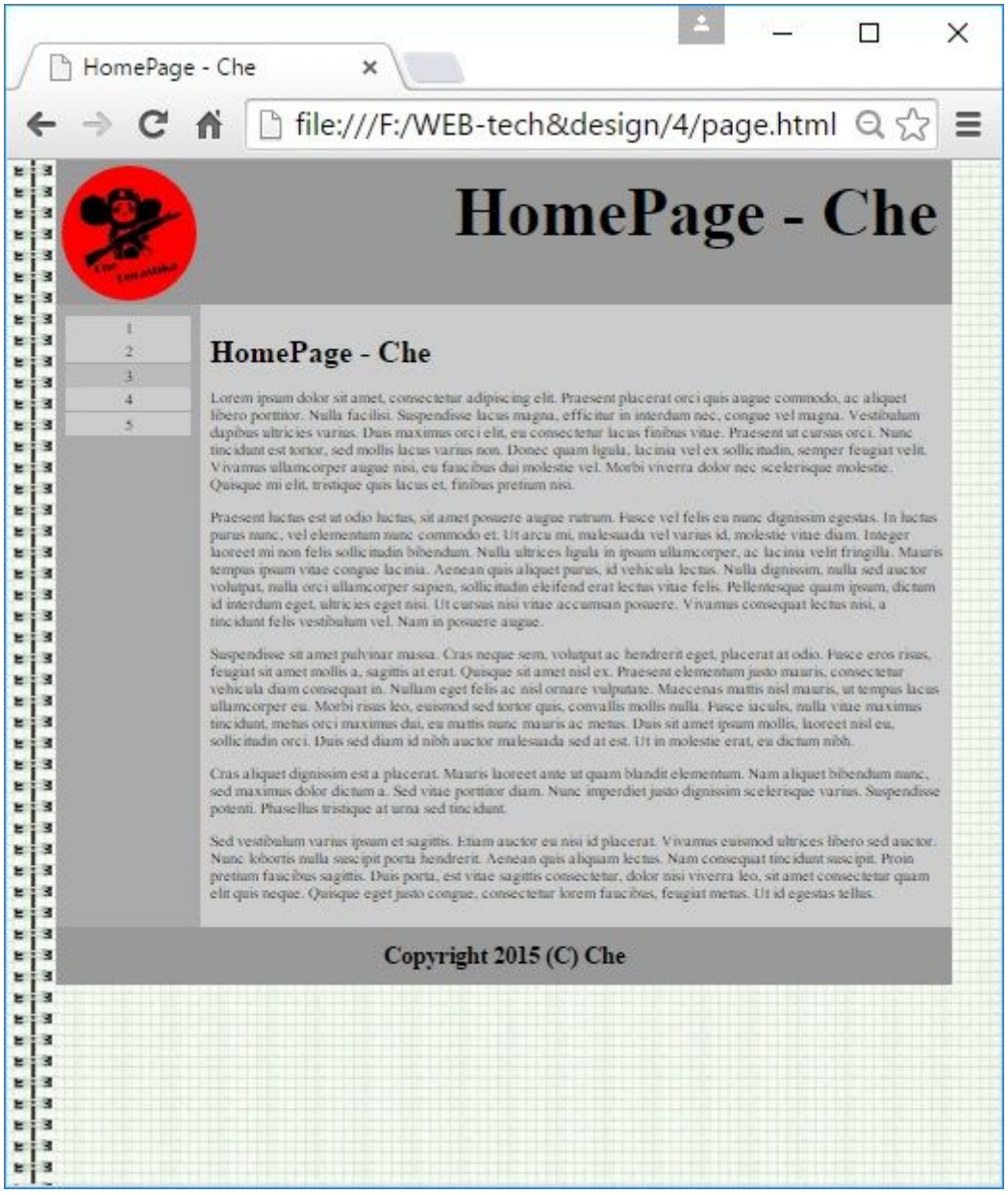
1 Підготуйте для цієї роботи декілька графічних файлів, а саме:

а) Файл для аватарки

б) файл для фону

в) файл або файли для

2 Використовуючи властивості позиіоннування CSS створіть особисту сторінку наступного вигляду, але про себе:


HomePage - Che
file:///F:/WEB-tech&design/4/page.html
HomePage - Che
1
2
3
4
5
HomePage - Che
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent placerat orci quis augue commodo, ac aliquet libero porttitor. Nulla facilisi. Suspendisse lacus magna, efficitur in interdum nec, congue vel magna. Vestibulum dapibus ultricies varius. Duis maximus orci elit, eu consectetur lacus finibus vitae. Praesent ut cursus orci. Nunc tincidunt est tortor, sed mollis lacus varius non. Donec quam ligula, lacinia vel ex sollicitudin, semper feugiat velit. Vivamus ullamcorper augue nisi, eu faucibus dui molestie vel. Morbi viverra dolor nec scelerisque molestie. Quisque mi elit, tristique quis lacus et, finibus pretium nisi.
Praesent luctus est ut odio luctus, sit amet posuere augue rutrum. Fusce vel felis eu nunc dignissim egestas. In luctus purus nunc, vel elementum nunc commodo et. Ut arcu mi, malesuada vel varius id, molestie vitae diam. Integer laoreet mi non felis sollicitudin bibendum. Nulla ultrices ligula in ipsum ullamcorper, ac lacinia velit fringilla. Mauris tempus ipsum vitae congue lacinia. Aenean quis aliquet purus, id vehicula lectus. Nulla dignissim, nulla sed auctor volutpat, nulla orci ullamcorper sapien, sollicitudin eleifend erat lectus vitae felis. Pellentesque quam ipsum, dictum id interdum eget, ultricies eget nisi. Ut cursus nisi vitae accumsan posuere. Vivamus consequat lectus nisi, a tincidunt felis vestibulum vel. Nam in posuere augue.
Suspendisse sit amet pulvinar massa. Cras neque sem, volutpat ac hendrerit eget, placerat at odio. Fusce eros risus, feugiat sit amet mollis a, sagittis at erat. Quisque sit amet nisl ex. Praesent elementum justo mauris, consectetur vehicula diam consequat in. Nullam eget felis ac nisl ornare vulputate. Maecenas mattis nisl mauris, ut tempus lacus ullamcorper eu. Morbi risus leo, euismod sed tortor quis, convallis mollis nulla. Fusce iaculis, nulla vitae maximus tincidunt, metus orci maximus dui, eu mattis nunc mauris ac metus. Duis sit amet ipsum mollis, laoreet nisl eu, sollicitudin orci. Duis sed diam id nibh auctor malesuada sed at est. Ut in molestie erat, eu dictum nibh.
Cras aliquet dignissim est a placerat. Mauris laoreet ante ut quam blandit elementum. Nam aliquet bibendum nunc, sed maximus dolor dictum a. Sed vitae porttitor diam. Nunc imperdiet justo dignissim scelerisque varius. Suspendisse potenti. Phasellus tristique at urna sed tincidunt.
Sed vestibulum varius ipsum et sagittis. Etiam auctor eu nisi id placerat. Vivamus euismod ultrices libero sed auctor. Nunc lobortis nulla suscipit porta hendrerit. Aenean quis aliquam lectus. Nam consequat tincidunt suscipit. Proin pretium faucibus sagittis. Duis porta, est vitae sagittis consectetur, dolor nisi viverra leo, sit amet consectetur quam elit quis neque. Quisque eget justo congue, consectetur lorem faucibus, feugiat metus. Ut id egestas tellus.
Copyright 2015 (C) Che

Лабораторная работа 5

Вивчення основ JavaScript

JavaScript спочатку створювався для того, щоб зробити web-сторінки «живими». Програми на цій мові називаються скриптами. У браузері вони підключаються безпосередньо до HTML і, як тільки завантажується сторінка - тут же виконуються.

Сучасний JavaScript - це «безпечна» мова програмування загального призначення. Він не надає низькорівневих засобів роботи з пам'яттю, процесором, оскільки спочатку був орієнтований на браузери, в яких це не потрібно.

Що ж стосується інших можливостей - вони залежать від оточення, в якому запущений JavaScript. У браузері JavaScript вміє робити все, що відноситься до маніпуляції зі сторінкою, взаємодії з відвідувачем і, в якійсь мірі, з сервером:

- Створювати нові HTML-теги, видаляти існуючі, міняти стилі елементів, ховати, показувати елементи і т.п.
- Реагувати на дії відвідувача, обробляти кліки миші, переміщення курсора, натискання на клавіатуру і т.п.
- Посилати запити на сервер і завантажувати дані без перезавантаження сторінки (ця технологія називається "AJAX").
- Отримувати і встановлювати cookie, запитувати дані, виводити повідомлення та багато іншого.

Hello World приклад

Тег **script** містить виконуваний код. Попередні стандарти HTML вимагали обов'язкового зазначення атрибуту type, але зараз він вже не потрібен.

Дії браузеру, коли він бачить **<script>**:

1) Починає відображати сторінку, показує частину документа до script;

2) Зустрівши тег script, перемикається в JavaScript-режим і не показує, а виконує його вміст;
3) Закінчивши виконання, повертається назад в HTML-режим і тільки тоді відображає решту документа;

Приклад:

```
<script>
    alert( 'Hello, World!' );
  </script>
```

Зовнішні скрипти, порядок виконання

Якщо JavaScript-коду багато - його винести в окремий файл, який підключається в HTML:

```
<script src="/path/to/script.js"></script>
```

Можно указать и полный URL. Вы также можете использовать путь относительно текущей страницы. Например, src="lodash.js"обозначает файл из текущей директории.

Коментарі

З часом програма стає великою і складною. З'являється необхідність додати коментарі, які пояснюють, що відбувається і чому. Коментарі можуть перебувати в будь-якому місці програми і ніяк не впливають на її виконання. Інтерпретатор JavaScript просто ігнорує їх. Однорядкові коментарі починаються з подвійного слеша //. Багаторядкові коментарі починаються слешем-зірочкою "/ *" і закінчуються зірочкою-слешем "* /"

Змінні

У сценаріях JavaScript можна використовувати змінні, звертаючись до них за назвою. Змінні можуть бути глобальними або локальними. Глобальні змінні досяжні з довільного місця сценарію. Область дії локальних змінних обмежено кодом функції, всередині якого оголошено ці змінні. При створенні сценаріїв JavaScript рекомендовано оголошувати змінні до їхнього використання та надавання початкових величин. Це спрощує відлагодження сценаріїв і зменшує ймовірність помилки.

Оголосити змінну у JavaScript можна при допомозі ключового слова *var*, таким чином:

var *назва змінної*

Тип змінної JavaScript надає лише при наданні змінній певної величини. Після такого надання назва змінної стане доступною, навіть якщо її попередньо не було оголошено.

Назва змінної:

- має починатися начинаться з літери латиниці або символів "_" чи "$", містити лише літери латиниці, цифри і символи "_" та "$";
- не може збігатися з жодним з зарезервированих слів JavaScript.

Надання змінним величини здійснюють за допомогою оператора "=". Після здійснення такої операції тип змінної буде змінено, але у процесі інтерпретації сценарію не буде зроблено жодного попереджувального повідомлення. Змінній можна надати спеціальну величину *null*. У цьому випадку змінній не призначено жоден з типів.

Тип *число* допускає використання різних форматів, наприклад,:

25 — запис цілого числа у десятковій системі числення;

0137 — запис цілого числа у вісімковій системі числення;

0xFF — запис цілого числа у шістнадцятковій системі числення;

386.7 — запис дійсного числа с плавающей десятковою крапкою;

25e5 або 25E5 — запис дійсного числа в науковій нотації, дорівнює $25 \cdot 10^5$.

Мова JavaScript містить шість типів даних: **Undefined** (непроникний), **Null** (нульовий), **Boolean** (логічний), **String** (рядковий), **Number** (числовий) і **Object** (об'єктний).

Якщо результат виконання операцій з числами не можна подати числом, то повертається величина NaN (від англійською *Not a Number*). Тип *текстовий рядок* описує послідовність символів між лапками (' або ", але одного вигляду). Слід зауважити, порожній рядок відмінний від *null*. Тип *логічних*

даних передбачає лише дві величини: *true* і *false*. Якщо змінну було оголошено, але їй не було надано величини, вона має *невизначений тип*.

Якщо у виразах зустрічаються змінні різних типів, інтерпретатор JavaScript може автоматично перетворити числові дані в рядки тексту. Перетворення рядка у число виконують лише з допомогою спеціальних функцій *parseInt* (у ціле) і *parseFloat* (у дійсне).

Завдання:

1. Вивести за допомогою alert своє ім’я;
2. Винести в зовнішній файл alert.js рішення попередньої задачі і прикріпити до поточного HTML-документу
3. Створіть 2 змінні: foo та bar. Запишіть до foo своє ім’я. Скопіюйте з foo до bar своє ім’я. Виведіть за допомогою alert змінну bar.
4. За допомогою циклу вивести 10 строк
5. Зробіть (за допомогою JS), щоб при наведенні на кнопку-посилання в неї змінювався вміст. Наприклад було написано “Натисни сюди”, а стане “Не натискай”.
6. За допомогою чистого JS (без використання фреймворків) створіть “бігучу строку”.

Лабораторная работа 6

Завдання:

1. За допомогою бібліотеки JQuery створіть меню із анімованими елементами.
2. За допомогою бібліотеки JQuery, або просто на JavaScript, зробіть Sprite анимацію.

Лабораторна робота 7

«Ознайомлення з PHP»

Вивести на екран повідомлення «Hello World» і інформацію про PHP за допомогою команди PHPINFO().

У цій лабораторній роботі коротко викладені основи мови PHP.

Теоретична частина.

1.Основи мови.

Почнемо з того, що для **PHP** існує чотири способи відділення його від загального коду **HTML**

1. *<? echo ("SGML інструкції\n"); ?>*
2. *<?php echo("XML документ\n"); ?>*
3. *<script language="php">*

   echo ("спеціально для FrontPage");

   </script>
4. *< % echo ("ASP-стиль"); %>*

   < %= $variable; # Коментар "<%echo .." %>

Далі необхідно звичайно сказати і про те, що інструкції у **PHP** відокремлюються друг від друга сішно-паскалевсько-перловим способом - крапкою з комою. Хоча перед закриваючим тегом (**?>**) крапку з комою ставити не обов'язково.

<?php echo "This is a test";
echo "This is a test also" ?>

PHP підтримує коментарі в стилі **C**, **C++** і **Unix shell**. Наприклад:

```
<?php
    echo "test"; // Коментар у стилі C++
    /* Це багатостроковий
    коментар у стилі C++*/
    echo "test2";
    echo "Test3"; # Це unix-shell коментар
?>
```

ТИПИ

PHP підтримує наступні типи даних:

- **integer** - цілі
- **floating-point numbers** чи **double** - числа із плаваючою комою
- **string** - рядок, текст
- **array** - масиви
- **object** – об’єкти

ОБЛАСТЬ ВИДИМОСТІ ЗМІННИХ

По-перше, усі змінні починаються із символу **$**

По-друге, існують границі визначення змінних. Наприклад, щоб використовувати глобальні змінні у функціях, необхідно їх спочатку декларувати як глобальні, інакше ви будете використовувати локальні змінні. Поясню на прикладі:

$a = 1; / глобальне визначення */*

```
Function Test (){
        $a=2;
        echo $a; /* локальна змінна */
}
Test ();
echo $a;
```

У такий спосіб у функції використовується локальна, власна змінна і результатом програми буде висновок чисел 2 і 1, а не 2 і 2, як деякі могли подумати.

Ну, а якщо ви хочете у функції використовувати глобальні змінні, необхідно зробити так (декларувати за допомогою оператора **global**):

```
$a = 1; /* глобальне визначення */
Function Test (){
        global $a;
        $a=2;
        echo $a; /* локальна змінна */
}
Test ();
echo $a;
```

Ви можете використовувати не тільки змінні, але і константи за допомогою функції **define()**. Гляньте на наступний приклад:

```
define("CONSTANT", "Hello world.");
echo CONSTANT;
```

зверніть увагу на те, що перед ім'ям константи не пишеться символ змінної **$** і це правильно.

Список визначених констант:

- **__FILE__** - Ім'я файлу виконуваного скрипта.

- **__LINE__** - Кількість ліній, інтерпретованих на даний момент у цьому скрипті.
- **PHP_VERSION** - Отут зберігається версія **PHP**. Наприклад: **'3.0.8-dev'**.
- **PHP_OS** - Ім'я операційної системи, на якій виконується **PHP**-скрипт.
- **TRUE** - Істина.
- **FALSE** - Неправда.
- **E_ERROR** - Описує помилку, що виникла, після якої продовження роботи неможливо.
- **E_WARNING** - Описує помилку, після якої продовжується виконання скрипта.
- **E_PARSE** - Описує синтаксичну помилку, при розборі інтерпретатором тексту скрипта.
- **E_NOTICE** - Просто якесь повідомлення від інтерпретатора. Можливо помилка, а можливо і немає.

АРИФМЕТИЧНІ ОПЕРАЦІЇ

Арифметичних операцій у **PHP** усього п'ять, от вони:

$a + $b

$a - $b

$a * $b

$a / $b

$a % $b

Результатом виконання останньої операції буде залишок від ділення **$a** на **$b**.

СТРОКОВІ ОПЕРАЦІЇ

Строковою операцією вважається операція додавання двох рядків. Причому виглядає вона досить незвичайно, але практично:

$c = $a . $b;

Тобто символом цієї операції є крапка. А результатом її виконання буде звичайний рядок, що складається з **$a** і **$b**.

ОПЕРАЦІЯ ПРИСВОЮВАННЯ

Це знак **'='** і природно цей знак означає, що змінної ліворуч від нього буде привласнене значення, отримане в результаті виконання яких або чи операцій змінної/константи з правої сторони. Причому отут можливі деякі **C**++-ні варіанти, як:

```
$a = ($b = 4) + 5;          // $a дорівнює 9, а $b 4.
$a += 5;                    // аналогічно  $a = $a + 5;
$b = "Привіт ";
$b .= "усім!";              // аналогічно $b="Привіт усім!"
```

БІНАРНІ ОПЕРАЦІЇ

- **$a & $b** - Побітове І (AND)
- **$a | $b** - Побітове ЧИ (OR)
- **~ $a** - Виключаюче ЧИ (XOR)
- **$a << $b** - Зрушення вліво на **$b** бітів
- **$a >> $b** - Зрушення вправо на **$b** бітів

ЛОГІЧНІ ОПЕРАЦІЇ

- **$a and $b** - І (AND)
- **$a && $b** - Теж саме, що і попереднє
- **$a or $b** - Чи (OR)
- **$a || $b** - Теж, що і попереднє
- **$a xor $b** - Виключаюче ЧИ(XOR)
- **! $a** - Інверсія (NOT)

ОПЕРАЦІЇ ПОРІВНЯННЯ

- **$a == $b**
- **$a != $b**
- **$a < $b**
- **$a > $b**
- **$a <= $b**

- **$a >= $b**

Практична частина.

Завдання.

Вивести в браузері повідомлення «Hello World this is PHP !» і інформацію про PHP, це можна зробити за допомогою функції echo, а інформацію про PHP можна вивести за допомогою функції phpinfo(), що виводить у браузері інформацію про версію, опції що включені у файлі конфігурації *PHP.INI*, підключених модулях.

```
<?  echo"<H1 align=center> Hello World this is PHP! </H1>";
phpinfo(); ?>
```

Лабораторна робота №8

«Написи, посилання, робота з циклами»

Ціль:«Вивчення елементів мови»

Написати скрипт, який виводить у браузері напису, посилання, списки і т.д.

Теоретична частина

Ініціалізація Масивів

Масив може ініціалізуватися одним із двох способів: послідовним присвоєнням значень, чи за допомогою конструкції array().

При послідовному додаванні значень у масив ви просто записуєте значення елементів масиву, використовуючи порожній індекс. Кожне наступне значення буде додаватися як останній елемент масиву.

$names[] = "Jill";   // $names[0] = "Jill"

$names[] = "Jack";   // $names[1] = "Jack"

Як у C і Perl, елементи масиву нумеруються, починаючи з 0, а не з 1.

Маніпуляції з масивом

PHP підтримує як скалярні так і асоціативні масиви. Фактично, між ними немає різниці. Ви можете створити масив, використовуючи функції list() чи array(), чи можна явно задати значення кожного елемента масиву.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

Можна також створити масив, просто додаючи в нього значення.

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

Підрахунок кількості елементів масиву здійснюється функцією count().

Переміщатися по масиву дозволяють функції next() і prev(). Іншим типовим способом переміщення по масиві є використання функції each().

Робота з циклами.

Цикл WHILE.

Цикл WHILE - найпростіший тип циклу в PHP. Він діє як і його аналог у C. Основна форма оператора WHILE :

WHILE(expr) statement

Зміст оператора WHILE простий. Він наказує PHP виконувати вкладений(і) оператор(и) доти, поки expr дорівнює TRUE. Значення вираження перевіряється щораз при початку циклу, так, що якщо значення вираження зміниться всередині циклу, то він не перерветься до кінця поточної ітерації (виконання всього блоку вкладених операторів - це одна ітерація). Іноді, якщо значення expr дорівнює FALSE із самого початку, цикл не виконується жодного разу.

Як і в IF, ви можете згрупувати кілька операторів усередині фігурних дужок чи використовувати альтернативний синтаксис :

WHILE(expr): вираження ... ENDWHILE;

Наступні приклади ідентичні - обоє виводять номера з 1 по 10:

```
/* example 1 */
$i = 1;
while ($i <= 10) {
print $i++;  }
/* example 2 */
$i = 1;
while ($i <= 10):
print $i;
$i++;
endwhile;
```

Цикл DO..WHILE.

Цикл DO..WHILE дуже схожий на WHILE за винятком того, що значення логічного вираження перевіряється не до, а після закінчення ітерації.

Основна відмінність у тім, що DO..WHILE гарантовано виконається хоча б один раз, що у випадку WHILE не обов'язково.

Для циклів DO..WHILE існує тільки один вид синтаксису:

```
$i = 0;
do {
        print $i;
} while ($i>0);
```

Цей цикл виконається один раз, тому що після закінчення ітерації буде перевірене значення логічного вираження, а воно дорівнює FALSE ($i не більше 0), і виконання циклу завершиться.

Досвідчені програмісти на C може бути знайомі з іншим використанням DO..WHILE, що дозволяє припинити виконання блоку операторів у середині шляхом впровадження його в цикл DO..WHILE(0) і використання оператора BREAK. Наступний код демонструє таку можливість :

```
do {
        if ($i < 5) {
        print "i is not big enough";
        break;
        }
        $i *= $factor;
        if ($i < $minimum_limit) {
        break;
        }
        print "i is ok";
        ...process i...
} while(0);
```

Не турбуйтеся, якщо ви не зовсім зрозуміли це. Ви можете програмувати дуже могутні скрипти і без цієї можливості.

Цикли FOR

Цикли FOR - найбільш могутні цикли в PHP. Вони працюють подібно їх аналогам у C. Синтаксис циклу FOR :

FOR (expr1; expr2; expr3) statement

Перше вираження (expr1) безумовно, обчислюється на початку циклу.

На початку кожної ітерації обчислюється expr2. Якщо воно дорівнює TRUE, то цикл продовжується й виконуються вкладений(і) оператор(и). Якщо воно дорівнює FALSE, то цикл закінчується.

Наприкінці кожної ітерації обчислюється expr3.

Кожне з цих виражень може бути порожнім. Якщо expr2 порожньо, то цикл продовжується нескінченно (PHP за замовчуванням вважає його рівним TRUE, як і C). Це не так даремно, як могло б показатися, тому що найчастіше вам потрібно закінчити виконання циклу, використовуючи, оператор BREAK у сполученні з логічною умовою замість використання логічного вираження в FOR.

Розглянемо наступні приклади. Усі вони виводять номера з 1 по 10 :

/* приклад 1 */

```
for ($i = 1; $i <= 10; $i++) {
        print $i;
}
```

/* приклад 4 */

```
for ($i = 1; $i <= 10; print $i, $i++) ;
```

Звичайно, перший варіант здається кращим, але як здається можливість використання порожніх виражень у циклі FOR найчастіше виявляється корисною.

Робота з посиланнями й написами.

Вставка посилань і написів у документ здійснюється аналогічно, за допомогою функцій:

ECHO “<посилання>”,

де <посилання> - звичайне посилання HTML, тобто <a href=”<path>”>.

Також можна використовувати функцію:

PRINT(“<посилання>”),

де <посилання> - звичайне посилання HTML, тобто <a href=”<path>”>.

Також можна використовувати функцію PRINTF аналогічну функції PRINTF у Ci;

PRINTF(“<посилання>”,arguments);

/* приклад */

printf(”<a href=\"%s?id=%s\">%s>”,$php_self);

Практична частина.

У даній лабораторній роботі потрібно написати скрипт, що виводить у браузері написи, посилання, списки.

Для виводу в браузері написів можна використовувати функцію printf(), на мій погляд, ця функція найбільш зручна для виводу текстових повідомлень у вікні браузера, тому що в ній легко вставляти змінні в текстові повідомлення:

PRINTF(“< текст >”,arguments);

У текст замість змінних вставляється %S, це означає, що в цьому місці в текст повинна бути вставлена змінна, котра описана в arguments. Але якщо потрібно вивести просте повідомлення, що не містить змінних, то краще використовувати функції echo “” чи print(“”);

Наприклад:

```
echo "<br><H1 align=center> Лабораторна робота №2 </H1>";



printf("Квадрат числа %s дорівнює %s <br>",$i,$i*$i);

print("<h2 align=center>Демонстрація роботи з посиланнями</H2>").
```

Перейдемо тепер до висновку в браузер списків. Допустимо потрібно вивести список із п'яти пунктів, для цього в стандартному HTML потрібно написати 5 рядків коду:

Приклад:

```
<li>Квадрат числа 1 дорівнює 1</li>
<li>Квадрат числа 2 дорівнює 4</li>
<li>Квадрат числа 3 дорівнює 9</li>
<li>Квадрат числа 4 дорівнює 16</li>
<li>Квадрат числа 5 дорівнює 25</li>
```

У PHP для цього можна й потрібно використовувати цикл, тому що це істотно скорочує кількість рядків коду, це в даному прикладі рядків усього 5, а якщо їх буде 10 чи 20, тому використання циклів для виводу інформації такого типу дуже доцільно.

Приклад:

```
<?
for ($i=1;$i<=5;$i++)
{ printf("<li>Квадрат числа %s дорівнює %s </li>",$i,$i*$i); }
```

?>.

Вставка посилань у документ здійснюється тими ж функціями, що використовуються для висновку рядків, тому що по суті справи посилання це той же рядок, що тільки посилається на інший документ або на мітку в цьому ж документі. Тобто вставка посилання в документ може бути здійснена за допомогою функцій ECHO, PRINT, PRINTF; якщо потрібно просто вставити посилання в тіло документа те краще використовувати функції ECHO і PRINT, або просто написати тег HTML. Якщо ж дані введені у форму потрібно обробити і потім вивести у виді посилання, то в цьому випадку незамінною виявиться функція PRINTF.

Наприклад:

Звичайне посилання

```
Print(“<a href="mailto:zave@pisem.net"> Mail to Alex </a>”);
```

При обробці форми:

```
print("<h2 align=center>Демонстрація роботи з посиланнями</H2>");
if (@($undo)):
unset($do);
endif;
if (!@($do)):
?>
<form action="lab2.php">
<table>
<tr><td>Введіть своє ім'я :
<td><input type=text name="name">
<tr><td>Введіть свій E-MAIL :
<td><input type=text name="email">
```

```
<tr><td><td align=center><input type=submit name="undo"
value="Reset"><input type=submit name="do" value="Send">
</table>
</form>
<?
else:
?>
<form action="lab2.php">
<table>
<tr><td>Введіть своє ім'я :
<td><input type=text name="name">
<tr>
<td>Введіть свій E-MAIL :
<td><input type=text name="email">
<tr><td>
<td align=center><input type=submit name="undo" value="Reset"><input
type=submit name="do" value="Send">
</table>
</form>
<hr size=2>
<? printf("Ваше ім'я %s, E-mail:<a
href=\"mailto:%s\">%s</a>",$name,$email,$email); ?>



<hr size=2>
<? endif; ?>
```

Приклад виконання даного завдання:

```
<html>
<body>
<?
echo "<br><H1 align=center> Лабораторна робота №2 </H1>";
echo "<br><hr size=2>";
?>
<h2 align=center>Демонстрація роботи з циклами </H2>
<?
for ($i=1;$i<=5;$i++)
{
  printf("<li>Квадрат числа %s дорівнює %s </li>",$i,$i*$i);
}
?>
<br>
<hr size=2>
<?
print("<h2 align=center>Демонстрація роботи з посиланнями</H2>");
if (@($undo)):
unset($do);
endif;
if (!@($do)):
```

```
?>
<form action="lab2.php">
<table>
<tr>
<td>Введіть своє ім'я :
<td><input type=text name="name">
<tr>
<td>Введіть свій E-MAIL :
<td><input type=text name="email">
<tr>
<td>
<td align=center><input type=submit name="undo" value="Reset"><input
type=submit name="do" value="Send">
</table>
</form>
<?
else:
?>
<form action="lab2.php">
<table>
<tr>
<td>Введіть своє ім'я :
<td><input type=text name="name">
<tr>
<td>Введіть свій E-MAIL :
<td><input type=text name="email">
<tr>
<td>
```

```
<td align=center><input type=submit name="undo" value="Reset"><input
type=submit name="do" value="Send">
</table>
</form>
<hr size=2>
<? printf("Ваше ім'я %s, E-mail:<a
href=\"mailto:%s\">%s</a>",$name,$email,$email); ?>
<hr size=2>
<? endif; ?>



</html>
```

Лабораторна робота №9

«Робота з графічною бібліотекою GD»

Теоретична частина

Що потрібно для роботи з графікою.

Для роботи з графічними функціями потрібно підключити GD-бібліотеку. Для цього в **php.ini** треба раскоментувати рядок "*extension=php_gd.dll*" (і запустити знову Apache).

Також важлива правильність шляху розташування dll-файлів бібліотек для PHP (рядок "*extension_dir* =" у php.ini).

Графічні функції у PHP працюють із трьома форматами: GIF, JPEG і PNG. Насправді підтримуються тільки два формати. GIF у PHP не підтримується. Тобто при виконанні скрипта PHP, зустрівши функцію GIF-групи, видасть повідомлення про те, що використовується невизначена (undefined function) функція, і нічого робити не буде.

Тепер про функції роботи з графікою.

Логіка роботи з графікою проста. Є html-сторінка, у якій треба вивести картинку. Нехай саму картинку генерує скрипт, що знаходиться у файлі *image.php*. У файлі, який викликає картинку буде написано щось типу цього:

<img src="image.php">

Тут керування передається скрипту генеруючому картинку (у прикладі буде сірий прямокутник).

У *image.php* буде такий код:

<?php

Header ("Content-type: image/jpeg"); //HTTP-заголовок для картинки

$im = ImageCreate (500, 30); //створюємо image

$background = ImageColorAllocate ($im, $r, $g, $b); //створюємо колір тла

ImageJpeg ($im); //виводимо image

ImageDestroy ($im); //звільнимо пам'ять, виділену під image

?>

Коментарі тут такі. Перше - у файлі *image.php* перед тегом, що відкриває php-секцію, не повинно бути жодного символу (навіть пробілів!). До функції **HEADER()** також не повинне бути жодного оператора **ECHO** чи **PRINT**. Файл не повинний починатися з декларації html (<*HTML*>).

Це важливо тому, що з появою символів на вивід web-сервер припиняє формування HTTP-заголовка і переходить до формування секції даних (наприклад, власне html-сторінки). А функція **HEADER()** саме і додає/заміняє секцію заголовка (у даному випадку, що відповідає за тип даних).

Функція **IMAGECREATE()** створює image розміру, зазначеного в параметрах функції: ширина й висота. Значення, що повертається - ціле, що є ідентифікатором ресурсу image.

IMAGECOLORALLOCATE() повертає ідентифікатор кольору, формованого на основі RGB-значень (2..4 параметри; перший параметр - ідентифікатор image-ресурсу).

IMAGEJPEG() формує картинку (у даному випадку, jpeg-формату) для передачі сервером клієнтському браузеру. Усього такого роду функцій три. Крім зазначеної для jpeg-формату є ще **IMAGEPNG()** і **IMAGEGIF()** для png- і gif-форматів відповідно (формально, а насправді gif-ы, як я вже говорив, не підтримуються, так, що усього функцій, що формують картинки, дві).

IMAGEDESTROY() звільняє пам'ять, виділену функцією **IMAGECREATE()** під image. Єдиний параметр - ідентифікатор image-ресурсу.

Тілом картинки стає перший створений після створення image колір. А як зробити тіло картинки прозорим? Дуже просто. Для цього є функція

IMAGECOLORTRANSPARENT(), що ставить для зазначеного кольору в зазначеній картинці атрибут *transparent*. Перший параметр цієї функції - ідентифікатор image-ресурсу, а другий (необов'язковий) - ідентифікатор створеного кольору, наприклад,:

imagecolortransparent ($im, $color);

Взагалі говорячи, ідентифікатори створюваних кольорів представляють із себе, як я зрозумів, цілі числа, що починаються з нуля (перший виклик **IMAGECOLORALLOCATE()**) і збільшуються на одиницю при кожнім наступному виклику функції створення кольору. До речі, якщо картинка все-таки не формується, перевірте синтаксис. У випадку помилок PHP виводить у вихідний потік (сервер віддасть цей висновок браузеру) повідомлення про помилку (якщо в php.ini параметр "error_reporting" виставлений у "on"). Причому виводить до виконання скрипта, тобто на етапі компіляції, а не виконання, а виходить, до виконання функції **HEADER()** - див. вище.

Якщо ж у php.ini параметр "display_errors" замість значення за замовчанням значення "E_ALL & ~E_NOTICE" буде дорівнює, скажемо, "E_ALL" (тобто виводити всі повідомлення, включаючи попередження), те залишитися без картинки можна, наприклад, використовувавши неоголошену змінну.

Лабораторна робота №10

«Робота з MySQL»

Ціль: «Створення WEB-сайту з використанням команд MySQL. Робота з базою даних із браузера»

Т.е. фактично потрібно створити базу даних і основні функції роботи з нею (додавання, видалення, відновлення й перегляд)

Теоретична частина.

Для роботи з MySQL потрібно щоб був інстальований пакет MySQL. Ніяких додаткових настроювань для PHP робити не потрібно, тому що починаючи з версії 4.0 у PHP убудована підтримка MySQL.

1. З'єднання з MySQL.

Перед початком роботи з MySQL виконуємо з'єднання із сервером:

mysql_connect -- відкриває з'єднання з MySQL сервером

int mysql_connect(string [hostname] [:port] , string [username] , string [password]);

Повертає: Правильний ідентифікатор зв'язку MySQL при успішному виконанні, чи false при помилці.

$link=mysql_connect('localhost') or die(mysql_error());

mysql_connect() установлює з'єднання з MySQL сервером. Всі аргументи, доповняльні (опціональні), і якщо вони пропущені, то встановлюються за замовчуванням -('localhost', ім'я користувача, що володіє процесом, порожній пароль). Рядок "ім'я сервера"("hostname") також може містити номер порту(подібно "hostname:port").

У випадку, якщо буде зроблений другий виклик mysql_connect() із тими ж аргументами, ніякий новий зв'язок не установиться - замість цього, буде повернутий ідентифікатор зв'язку уже відкритого зв'язку.

Наступна функція - MYSQL_SELECT_DB() – Вибирає Базу Даних MySQL.

int mysql_select_db(string database_name, int [link_identifier]);

Повертає: true при успішному виконанні , false при помилці

mysql_select_db() установлює поточну активну базу даних у сервері, що зв'язується з визначеним ідентифікатором зв'язку. Якщо не визначений ідентифікатор зв'язку, приймається останній відкритий зв'язок. Якщо зв'язок не відкритий, функція спробує встановити зв'язок, як якби була викликана функція mysql_connect()

Вона має два параметри: перший - ім'я бази, і другий (необов'язковий) - **id** з'єднання (**$link**). Під необов'язковістю ідентифікатора з'єднання як параметра розуміється наступне. Якщо параметр не зазначений, то використовується останнє відкрите з'єднання. А якщо жодне з'єднання не відкрите, то неявно виконується connect (тобто при використанні одного з'єднання **MYSQL_CONNECT()** явно викликати не обов'язково).

2. Створення бази даних.

MySQL підтримує наступні типи/групи типів даних:

1. Числові. Можливі модифікатори: UNSIGNED для оголошення беззнаковості, ZEROFILL для заповнення лідируючих пробілів нулями (мається на увазі зовнішній вигляд при висновку). Можуть бути створені як цілі (TINYINT, SMALLINT, INT, BIGINT і ін.), так і числа з крапкою, що плаває, (FLOAT, DOUBLE, REAL і ін.).
2. Строкові. Можливі модифікатори: BINARY для оголошення, поля як бінарного (будь-які коди збережених символів), NATIONAL - модифікатор за замовчуванням, - використання набору символів для сортування, порівняння й ін. Цікавий модифікатор. Відповідає за конструкцію SET-групи: SET CHARACTER SET character_set_name | DEFAULT, де character_set_name може приймати значення cp1251_koi8. Однак ці установки, виставлені за замовчуванням. Виходить, без модифікатора результат той же, що і з ним. Я так зрозумів, що ці фішки

для майбутнього використання. Модифікатор для типу CHAR VARYING створює строкове поле перемінної довжини.

3. BLOB-поля - поля для збереження двоїнних даних.

Типи даних, що не потрапили в попередні три групи:

4. TIMESTAMP - поле зберігає дату й час останньої зміни запису. Це значить, що, додавши в таблицю поле типу TIMESTAMP (наприклад, скориставшись конструкцією ALTER TABLE table_name ADD COLUMN column_name TIMESTAMP), Ви, не роблячи ніяких змін поля типу TIMESTAMP, будете в ньому мати час останньої операції із записом, що впливає на вміст рядка таблиці.
5. DATE, TIME, DATETIME - полючи збереження дати, часу, і того, і іншого. Отут, я думаю все ясно.
6. YEAR - поле, додане у версії 3.22, - для збереження року в інтервалі з 1901 по 2155.
7. ENUM - поле, що зберігає одне зі значень, зазначених у списку при створенні (модифікації структури) таблиці, наприклад, ALTER TABLE tab_name ADD COLUMN col_enum ENUM('Ага', 'Угу', 'Ну його).

Для того щоб створити власне базу даних, потрібно послати в MySQL відповідний запит, що здійснюється за допомогою функції

mysql_query – Відправляє SQL-запит на MySQL

Опис

int mysql_query(string query, int [link_identifier]);

mysql_query() надсилає запит у базу даних, до дійсного часу активну на сервері, що зв'язаний з визначеним ідентифікатором зв'язку. Якщо link_identifier не зазначений, використовується останній відкритий зв'язок.

mysql_create_db

mysql_create_db -- Створює базу даних MySQL

Опис

int mysql_create_db(string database name, int [link_identifier]);

mysql_create_db() намагається створювати нову базу даних на сервері зв'язаному з визначеним ідентифікатором зв'язку.

mysql_create_db('G_book') or die(mysql_error()).

Тепер база даних створена, але в неї немає таблиць. Для того щоб створити в ній таблицю (ці), потрібно послати MySQL відповідний запит наступного виду:

```
$sql="CREATE TABLE gbook (
        id int(10) DEFAULT '0' NOT NULL auto_increment,
      name varchar(30),
        email varchar(30),
        hpage varchar(30),
        subj varchar(150),
     PRIMARY KEY (id)
)",
```

де id, name, email, hpage, subj – це імена полів таблиці.

Тепер у створеній раніше базі даних є таблиця, але вона порожня і щоб неї заповнити потрібно надіслати відповідний запит MySQL.

Розглянемо основні команди MySQL для роботи з таблицями:

$sql="INSERT INTO " table_name" (fieldname1, fieldname2, ..., fieldnameN) VALUES ("fieldvalue1", "fieldvalue1", ..., "fieldvalueN")";

такий SQL запит уставляє в таблицю з ім'ям *" table_name"* значення *"fieldvalue..."*у зазначені імена полів *fieldname.....*

$sql="UPDATE " table_name" SET fieldname1="fieldvalue1", fieldname="fieldvalue" WHERE id=$id";

такий SQL запит обновляє значення в рядку номер id таблиці з ім'ям *" table_name"*.

$sql="DELETE FROM " table_name" WHERE id=$id";

такий SQL запит видаляє рядок номер id таблиці з ім'ям *“ table_name”*.

*$sql="select * from “ table_name”";*

такий SQL запит використовується для перегляду таблиці з ім'ям *“ table_name”*. Результати такого запиту можна вивести в такий спосіб:

```
$result = mysql_db_query("database","select * from table");
 while($row = mysql_fetch_array($result))
 {
 echo $row["user_id"];
 echo $row["fullname"];
 }
```

Практична частина.

У даній лабораторній роботі потрібно створити базу даних і її оброблювач, що реалізує основні функції роботи з базою даних.

Для створення таблиці необхідно виконати наступне:

1. З'єднатися із сервером MySQL;

$link=mysql_connect('localhost') or die(mysql_error());

2. Створити базу даних, це реалізується за допомогою функції mysql_create_db(), параметром якої є ім'я бази даних. mysql_create_db('Cars') or die(mysql_error());

конструкція or die(mysql_error()) – використовується для того, щоб при виникненні помилки транслятор видавав повідомлення про помилку, яка відбулася. Після виконання цієї команди в директорії DATA, що знаходиться усередині директорії куди був інстальований пакет MySQL, з'явиться нова порожня директорія з ім'ям бази даних, директорія порожня, тому що в базі даних не створено ні однієї таблиці.

3. Наступним кроком є створення MySQL запиту, в який міститься інформація про створювану таблицю(назви полів, їхня довжина);

$sql="CREATE TABLE cars (

id int(10) DEFAULT '0' NOT NULL auto_increment,

name varchar(30),

speed varchar(10),

color varchar(10),

price varchar(10),

PRIMARY KEY (id)

)";

4. Посилка запиту MySQL, цей крок здійснюється за допомогою функції mysql_db_query("Cars",$sql); як параметри цієї функції використовуються ім'я бази даних і SQL запит. Після виконання цієї команди в директорії з базою даних з'являються три файли **cars.frm** – структура таблиці, **cars.isd** – дані, що знаходяться в таблиці, **cars.ism** – інформація про ключі таблиці.

Після виконання перерахованих вище кроків база даних і таблиця створені, можна приступати до написання скриптів реалізуючих функції роботи з базою даних.

Для того щоб додати запис у базу даних потрібно надходити в такий спосіб:

1. З'єднатися із сервером MySQL, і вибрати базу даних із якою передбачається працювати. З'єднання виробляється за допомогою функції mysql_connect('localhost') or die(mysql_error()):

$link=mysql_connect('localhost') or die(mysql_error());

а вибір бази даних виробляється за допомогою функції mysql_select_db("ім'я бази даних",$db);

mysql_select_db("Cars",$db);

після виконання цих команд установлюється зв'язок виробляється вибір бази даних.

2. Після того як база даних обрана, можна переходити до роботи з нею:

для роботи з базою даних використовуються SQL конструкції, які можна виконати за допомогою функції mysql_query(), в якості аргументу якої використовується SQL конструкція. Розглянемо SQL конструкції реалізуючі основні функції роботи з базою даних.

Функція додавання запису в таблицю.

Для додавання запису в таблицю варто використовувати SQL конструкцію наступного виду:

$sql="INSERT INTO “ table_name” (fieldname1, fieldname2, ..., fieldnameN) VALUES (“fieldvalue1”, “fieldvalue1”, ..., “fieldvalueN”)";

такий SQL запит уставляє в таблицю з ім'ям *“ table_name”* значення *“fieldvalue1,...,fieldvalueN”,*у зазначені імена полів *fieldname1,...,fieldnameN...* Важливим моментом є те, що як значення *fieldvalue* звичайно, використовуються змінні отримані при обробці форми і їхні імена можуть не відповідати назвам полів таблиці, тому значення записувані в таблицю *“fieldvalue1,...,fieldvalueN”*, повинні знаходитися в тій же черговості, що й імена полів.

Приклад.

```
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
$sql="INSERT INTO cars (id,name,speed,color,price) VALUES
('$ID','$name','$speed','$color','$price')";.
```

Функція видалення запису з таблиці.

Для видалення запису з таблиці варто використовувати SQL конструкцію наступного виду:

$sql="DELETE FROM “table_name” WHERE id=$id";

такий SQL запит видаляє рядок номер id таблиці з ім'ям *“table_name”*.

Приклад.

```
$db=mysql_connect("localhost");
```

```
mysql_select_db("Cars",$db);
$sql="DELETE FROM cars WHERE id=$id";
$result=mysql_query($sql);.
```

Функція відновлення запису в таблиці.

Для відновлення запису в таблиці варто використовувати SQL конструкцію наступного виду:

$sql="UPDATE " table_name" SET fieldname1="fieldvalue1", fieldname="fieldvalue" WHERE id=$id";

такий SQL запит обновляє значення в рядку номер id таблиці з ім'ям *" table_name"* і записує обновлене значення в поле *fieldname="fieldvalue"*.

Приклад.

```
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
$sql="UPDATE cars SET name='$name',speed='$speed',color='$color'
,price='$price' WHERE id=$id";
$result=mysql_query($sql);.
```

Функція перегляду таблиці.

Для перегляду записів у таблиці варто використовувати SQL конструкцію наступного виду:

*$sql="select * from " table_name"";*

такий SQL запит використовується для перегляду таблиці з ім'ям *" table_name"*. Результати такого запиту можна вивести в такий спосіб:

Приклад.

```
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
$result=mysql_query("SELECT * FROM cars",$db);
```

```
echo "<table border=0 bordercolorlight=white bordercolordark=black
align=center>\n";
echo "<tr align=center bgcolor=blue><td><font color=white>ID</td><td><font
color=white>Name</td><td><font color=white>Speed</td><td><font
color=white>Color</td><td><font color=white>Price</td></tr>\n";
do{
printf("<tr align=center bgcolor=lightblue><td><a
href=\"%s?id=%s\">%s</td><td><a
href=\"%s?id=%s\">%s</td><td>%s</td><td>%s</td><td>%s</td></tr></a>\n",
$PHP_SELF,$myrow["id"],$myrow["id"],$PHP_SELF,$myrow["id"],$myrow["na
me"],$myrow["speed"],$myrow["color"],$myrow["price"]);
}while($myrow=mysql_fetch_array($result));
}.
```

У цьому прикладі виробляється вибірка значень з таблиці й висновок їх у виді таблиці.

Приклад сайта реалізуючого базові функції для роботи з базою даних.

INDEX.PHP

```
<HTML>

<FRAMESET COLS="160,*">
  <FRAME NAME="main" SRC="main.php">
  <FRAME NAME="m1" SRC="main1.php">
</FRAMESET>
```

```
</HTML>
```

MAIN.PHP

```
<html>
<head>
<title>Search the database</title>
</head>
<body bgcolor=#eeeeee>
<font face=arial color=#330066 size=4>
<center>
<h3><a href="add.phtml" target="m1">Add to database </a><br>
<h3><a href="admin.phtml" target="m1">Admin database </a><br>
<h3><a href="search.phtml" target="m1">Search </a><br>
</h3>
</center>
</body>
</html>
```

MAIN1.PHP

```
<html>
<head>
<title>Лабораторна робота №4</title>
</head>
<body bgcolor=#eeeeee>
<font face=arial color=#330066 size=4>
<center><H1>
Лабораторна робота №4
</H1>
</center>
</body>
```

```
</html>
```

ADD.PHTML

```
<html><head>
<title>Adding to database</title></head>
<body bgcolor=#eeeeee><font face=arial>
<?php
if (!@($name)): $f=1;endif;
if (!@($speed)): $f=1;endif;
if (!@($color)): $f=1;endif;
if (!@($price)): $f=1;endif;
if(($submit) & (!@($f))){
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
echo "$count\n";
$sql="INSERT INTO cars (id,name,speed,color,price) VALUES
('$ID','$name','$speed','$color','$price')";
$result=mysql_query($sql);
echo "<font size=4 face=arial color=#330066>\n";
while (list($name,$value)=each($HTTP_POST_VARS)){
echo "$name = $value<br><br>\n";
}
echo "<font size=5 face=arial color=red>";
echo "<center>Thank you! Information has been added to database.</font>\n";
echo "<br><br><Br>\n";
echo "<form method=\"post\" action=\"$PHP_SELF\">\n";
echo "<input type=\"submit\" value=\"Go Back To Main\">\n";
echo "</form>\n";
}
```

```
else{
$count=1;
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
$temp=mysql_query("SELECT * FROM cars");
while($myrow=mysql_fetch_array($temp)){
$count=$count+1;//test how many records have been in database.
}
echo "<Br><Br>\n";
echo "<table boarder=0><tr><td align=right>\n";
echo "<font size=4 face=arial color=#330066>\n";
echo "<form method=post action=\"$PHP_SELF\">\n";
echo " <font color=red>Your ID will be $count\n</font><br>";
echo "<input type=\"hidden\" value=$count name=\"ID\"><br>\n";
echo "Name: <input type=\"text\" name=\"name\"><Br><Br>\n";
echo "Speed: <input type=\"text\" name=\"speed\"><Br><Br>\n";
echo "Color: <input type=\"text\" name=\"color\"><Br><br>\n";
echo "Price: <input type=\"text\" name=\"price\"><Br><br>\n";
echo "<input type=\"submit\" name=\"submit\" value=\"add to database\">\n";
echo "</form>\n";
echo "</td></tr></table>\n";?>
<H3 align=center>
<a href="main1.php" target="m1">  Back to Main </a>
</H3>
<?
}
?>
</body>
```

```
</html>
```

ADMIN.PHTML

```
<html><head>
<title></title></head>
<body bgcolor=#eeeeee link=#330066 vlink=#330066>
<?php
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
if($modify)
{
$sql="UPDATE cars SET name='$name',speed='$speed',color='$color'
,price='$price' WHERE id=$id";
$result=mysql_query($sql);
echo "<font size=4 face=arial color=#330066>\n";
echo "Here is the updated information<br><br>\n";
while (list($name,$value)=each($HTTP_POST_VARS)){
echo "$name = $value<br><br>\n";
}
echo "<font size=5 face=arial color=red>";
echo "<center>Information has been updated.</font>\n";
echo "<br><br>\n";
echo "<form method=\"post\" action=\"$PHP_SELF\">\n";
echo "<input type=\"submit\" value=\"Go Back To Main\">\n";
echo "</form>\n";
}
else if($delete)
{
$sql="DELETE FROM cars WHERE id=$id";
```

```
$result=mysql_query($sql);
echo "<font size=4 face=arial color=#330066>\n";
echo "The following record has been deleted:<br><br>\n";
while (list($name,$value)=each($HTTP_POST_VARS)){
echo "$name = $value<br><br>\n";
}
echo "<font size=5 face=arial color=red>";
echo "<center>Information has been deleted.</font>\n";
echo "<br><br>\n";
echo "<form method=\"post\" action=\"$PHP_SELF\">\n";
echo "<input type=\"submit\" value=\"Go Back To Main\">\n";
echo "</form>\n";
}
else
if($id)
{
$result=mysql_query("SELECT * FROM cars WHERE id=$id");
$myrow=mysql_fetch_array($result);
?>
<table boarder=0 width=250><tr><td align=right>
<font size=4 face=arial color=#330066>
<form method=post action="<?php echo $PHP_SELF ?>" >
ID: <input type="text" value="<?php echo $myrow["id"] ?>" name="id"><br>
Name: <input type="text" value="<?php echo $myrow["name"] ?>"
name="name"><Br><Br>
Speed: <input type="text" value="<?php echo $myrow["speed"] ?>"
name="speed"><Br><Br>
```

```
Color: <input type="text" value="<?php echo $myrow["color"] ?>"
name="color"><Br><br>
Price: <input type="text" value="<?php echo $myrow["price"] ?>"
name="price"><Br><br>
<input type="submit" name="modify" value="Modify It">
<input type="submit" name="delete" value="Delete It">
</form>
</td></tr></table>
<H3 align=center>
<a href="main1.php" target="m1">  Back to Main </a>
</H3>
<?php
}
else{
echo"<center><h3> Для чи видалення зміни значення виберіть потрібний
запис</center><br><hr size=2>";
echo "<font face=arial>\n";
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
$result=mysql_query("SELECT * FROM cars",$db);
echo "<table border=0 bordercolorlight=white bordercolordark=black
align=center>\n";
echo "<tr align=center bgcolor=blue><td><font color=white>ID</td><td><font
color=white>Name</td><td><font color=white>Speed</td><td><font
color=white>Color</td><td><font color=white>Price</td></tr>\n";
do{
printf("<tr align=center bgcolor=lightblue><td><a
href=\"%s?id=%s\">%s</td><td><a
```

```
href=\"%s?id=%s\">%s</td><td>%s</td><td>%s</td><td>%s</td></tr></a>\n",
$PHP_SELF,$myrow["id"],$myrow["id"],$PHP_SELF,$myrow["id"],$myrow["na
me"],$myrow["speed"],$myrow["color"],$myrow["price"]);
}while($myrow=mysql_fetch_array($result));
}
?>
</body></html>
```

SEARCH.PHTML

```
<html>
<title>Search the database</title></head><body bgcolor=#eeeeee>
<font face=arial color=#330066 size=4>
<center>
<H3 align=center>
<a href="main1.php"  target="m1">  Back to Main </a>
</H3>
<form method=post action="<?php echo $PHP_SELF ?>">
<font size=3 face=arial color=#330066><b>Search by:   </b></font>
<SELECT NAME="value">
<option value="id">ID
<option value="name">Name
<option value="speed">Speed
<option value="color">Color
<option value="price">Price
</select>
<br><br>
<input type=text name="keyword" size=35>
<input type=submit value="Search">
```

```
</form><br><hr color=red height=8>
<?php
if($keyword){
$db=mysql_connect("localhost");
mysql_select_db("Cars",$db);
if($value==id){
$sql="SELECT * FROM cars WHERE id='$keyword'";
}
else if($value==name){
$sql="SELECT * FROM cars WHERE name='$keyword'";
}
else if($value==speed){
$sql="SELECT * FROM cars WHERE speed='$keyword'";
}
else if($value==color){
$sql="SELECT * FROM cars WHERE color='$keyword'";
}
else{
$sql="SELECT * FROM cars WHERE price='$keyword'";
}
$result=mysql_query($sql);
if($myrow=mysql_fetch_array($result))
{
echo "<Center><font size=4 color=red face=arial><b>Result:</font>\n";
echo "<table border=1 bordercolorlight=white bordercolordark=black>\n";
echo
"<tr><td>ID</td><td>Name</td><td>Speed</td><td>Color</td><td>Price</td><
/tr>\n";
```

```
do{
printf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%s</td><td>%s</td></tr>
\n",$myrow["id"],$myrow["name"],$myrow["speed"],$myrow["color"],$myrow["
price"]);
}while($myrow=mysql_fetch_array($result));
}
else{
echo "<center><font color=red face=arial size=4>\n";
printf("Sorry, No records were found in our database!\n");
}
}
?>

</body>
</html>
```

Створення бази даних.

CREATE_C.PHP

```
<?
$link=mysql_connect('localhost') or die(mysql_error());
mysql_create_db('Cars') or die(mysql_error());
$sql="CREATE TABLE cars (
                id int(10) DEFAULT '0' NOT NULL auto_increment,
            name varchar(30),
                speed varchar(10),
                color varchar(10),
                price varchar(10),
        PRIMARY KEY (id)
)";
```

```
mysql_db_query("Cars",$sql);
echo " The table is CREATED !";
?>
```

Початкова сторінка.

Додавання запису.

Пошук у базі даних

Перегляд записів у таблиці

Видалення і відновлення запису в таблиці.