МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Web-програмування
Лабораторний практикум

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітніми програмами «Системи, технології та математичні методи
кібербезпеки», «Системи технічного захисту інформації» та «Математичні
методи моделювання, розпізнавання образів та безпека даних»
спеціальності 125 «Кібербезпека» та 113 «Прикладна математика»*

Київ
КПІ ім. Ігоря Сікорського
2021

Web-програмування. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 125 «Кібербезпека» та 113 «Прикладна математика» / А. Ю. Шелестов, Н. М. Куссуль; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1047 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 61 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 4 від 10.12.2020 р.) за поданням Вченої ради Фізико-технічного інституту Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (протокол № 11 від 26.11.2020 р.)

Електронне мережне навчальне видання

Web-програмування
Лабораторний практикум

Укладачі:

Шелестов Андрій Юрійович, д. техн. наук, проф.
Куссуль Наталія Миколаївна, д. техн. наук, проф.

Відповідальний редактор *Смирнов С.А.,* к.ф.-м.н., доц.

Рецензент *Лавренюк А.М.*, доцент кафедри інформаційної безпеки ФТІ, Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського"

Навчальний посібник «Web-програмування. Лабораторний практикум» присвячено вивченню Web-програмування за спеціальністю 125 «Кібербезпека» та 113 «Прикладна математика». Метою викладання курсу є надання студентам знань про Web-програмування, засвоєння можливостей використання PHP, JavaScript, MySQL для програмування динамічних Web-сайтів і Web-інтерфейсів доступу до баз даних. Вивчення курсу «Web-програмування» вимагає цілеспрямованої роботи над вивченням спеціальної літератури, активної роботи на лекціях та практичних заняттях, самостійної роботи та виконання індивідуальних завдань. Посібник містить необхідний теоретичний матеріал, приклади програм, а також завдання для виконання лабораторного практикуму.

ЗМІСТ

Лабораторна робота №1
Створення веб-сторінки формату HTML 5

Мета роботи: отримати навички створення сучасних HTML-сторінок відповідно до стандарту HTML5.

1.1. Теоретичні відомості

HTML (*HyperText Markup Language* — мова розмітки гіпертекстових документів) – стандартна мова веб-розмітки в Інтернеті. Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML містить засоби для:

1. Створення структурованого документу шляхом позначення складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше.
2. Отримання інформації з Інтернету через гіперпосилання.
3. Створення інтерактивних форм.
4. Включення зображень, звуку та інших об’єктів тексту.

Розширення файлу має вигляд “ .html ” або “ .htm ”

Історія розвитку

1989 року Бернерс-Лі запропонував впровадити на базі Інтернет гіпертекстову систему документів. Вже наприкінці 1990 року він розробив HTML і написав браузер та серверне програмне забезпечення для запропонованої системи. Далі HTML розвився наступним шляхом :

HTML (без номера версії, 30 квітня 1993): до тексту додано атрибути, які визначають курсивне або жирне написання літер, та зображення.

HTML 3.2 (14 січня 1997): були додані численні можливості, такі як таблиці, обтікання текстом зображень, інтеграція аплетів.

HTML 4.0 (18 грудня 1997): були додані таблиці стилів, скриптові програми та фрейми – готовий до використання комплекс програмних рішень.

HTML 4.01 (24 грудня 1999): заміна версії HTML 4.0, містить численні інкрементальні виправлення.

HTML 5 (Working Draft, 5 квітня 2008): кінець розробки запланований на 2014 рік.

В **HTML 5** реализовано багато нових синтаксичних особливостей. Введено нові елементи такі як `<video>`, `<audio>` та `<canvas>`, а також з’явилась можливість використовувати **SVG** (*Scalable Vector Graphics*— масштабована векторна графіка) та математичні формули. Нові можливості були запроваджені для спрощення створення та керування графічними та

мультимедійними об'єктами в мережі без необхідності використовувати сторонні API. Такі нові елементи, як `<section>`, `<article>`, `<header>` і `<nav>`, розроблені для того, щоб збагатити семантичний вміст сторінки.

Приклади розмітки сторінки та опис основних дескрипторів

Дескриптор – елемент розмітки гіпертексту.

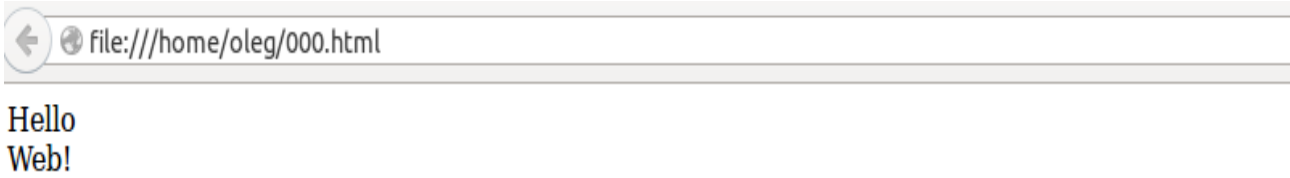
Дескриптор завжди записується між дужками `< >` і має вигляд <Дескриптор параметр1="ЗНАЧЕННЯ" … параметрN = "ЗНАЧЕНННЯ">. *Наприклад*: `<H1 ALIGN="CENTER"> Hello Web! </H1>`, де `H1` – дескриптор заголовку першого рівня, `ALIGN` – атрибут, що визначає вирівнювання заголовку (по центру в даному випадку).

Hello Web!

Атрибути дескриптора задають значення властивостей заданого об'єкту, що знаходиться в контейнері. Значення властивостей, що містять пробіли в назві беруть в лапки, в інших випадках лапки можна пропустити. Тобто `<H1 ALIGN=CENTER> Hello Web! </H1>` та `<H1 ALIGN="CENTER"> Hello Web! </H1>` одинаково відтворюються браузером.

Дескриптори бувають блочні одиночні та контейнерні (парні). *Контейнерами* називаються називається пара: відкриваючий <Дескриптор> та закриваючий </Дескриптор>.

Прикладом одиночного дескриптора є дескриптор переводу переводу рядку `<br>`. Він переводить рядок з того місця, де його зустрів браузер, на наступний. Наприклад : `Hello <br> Web!`.



Вигляд базової веб-розмітки сторінки має наступний вигляд :

```
<! DOCTYPE html>

<HTML>

<HEAD> Заголовок документу </HEAD>

<BODY> Тіло документу </BODY>

</HTML>
```

Оголошення <!DOCTYPE> вказує web-серверу спосіб обробки документа і те, які дескриптори можуть знаходитися в тілі сторінки. Досить часто це оголошення ігнорується браузерами, проте при написані сторінки мовою HTML 5 воно є необхідним.

Дескриптор <HTML> визначає межі документа.

Дескриптор <HEAD> призначений для збереження елементів, які призначені для допомоги браузеру при роботі з даними.

Дескриптор <BODY> призначений для виділення частини документа, яка буде візуалізована для користувача.

BACKGROUND: Атрибут задає графічне зображення, яке як мозайка заповнить фон документу. Синтаксис : <BODY BACKGROUND="(URL)(шлях) ім'я файлу">

BGCOLOR: Атрибут задає колір фону документа за допомогою 16-значних кольорів RGB , або за допомогою прописного літералу, який відповідає за певний колір. Синтаксис: <BODY BGCOLOR="#ff0000"> або <BODY BGCOLOR="RED">

LINK: Атрибут задає колір гіперпосилань, в більшості браузерів він заданий за замовчуванням темно синім. Синтаксис: <BODY LINK="колір">

ALINK: Атрибут задає колір гіперпосилань, при безпосередньому натиснені курсором на гіперпосилання. Синтаксис: <BODY ALINK="колір">

VLINK: Атрибут задає колір гіперпосилань, що вже відвідувались. Синтаксис: <BODY VLINK="колір">

Елементи оформлення тексту:

<I> Елемент </I> використовується для виділення курсивним шрифтом слова чи тексту. Синтаксис: <I> Текст </I>

<B> Елемент </B> використовується для виділення напівжирним шрифтом слова чи тексту. Синтаксис: <B> Текст </B>

<U> Елемент </U> використовується для підкреслення слова чи тексту. Синтаксис: <U> Текст </U>

^{Елемент} використовується для виділення надрядкових слів чи тексту. Синтаксис: ^{Текст}

_{Елемент} використовується для виділення підрядкових слів чи тексту. Синтаксис: _{Текст}

<BIG> Елемент </BIG> використовується для виділення великим шрифтом слова чи тексту, відносно основного тексту. Синтаксис: <BIG> Текст </BIG>

<SMALL> Елемент <SMALL> використовується для виділення малим шрифтом слова чи тексту, відносно основного тексту. Синтаксис: <SMALL> Текст </SMALL>

<FONT> Елемент </FONT> використовується для виділення шрифтом слова чи тексту. Разом з ним використовуютсья два атрибути size та color. Синтаксис: <FONT size=n color="колір">Текст </FONT>

<BASEFONT> використовується як альтернатива атрибуту size елемента <FONT>, він дозволяє задати базовий розмір шрифту у всьому документі та не має кінцевого дескриптору. За замовчуванням,значення задається рівним 3 . Синтаксис: <BASEFONT size=n>

Розглянемо застосування дескрипторів та їх атрибутів на прикладі оформлення сторінки форми анкети студента.

1.2. Приклад

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title> Анкета студента НТУУ "КПІ" </title>
    </head>

    <body bgcolor=blue>
        <header style="background:yellow;
height:30px;width:100%; margin:auto;">
            <p style="text-align: center"> Анкета
студента НТУУ "КПІ"
            </p>
        </header>
        <div style="position:relative;
background:white; margin:auto;height:1000px;
width:80%;margin-top:10px ">
        <div
style="position:absolute;left:10px;top:10px;
background:green; width:240px;height:300px;">
        <p style="text-align: center"> <img>Тут повина
розміщується фото студента
        </p>
            </div>
        <div
style="position:absolute;left:260px;top:10px;margin-
right:10px; width:745px">
                <table border="2" width="100%"
cellpadding="5">
                    <tr>
                <th width="35%"> Прізвище </th>
                <td width="65%"> Іваненко </td>
```

```
                </tr>
                <tr>
                <th width="35%"> Ім’я </th>
                <td width="65%"> Іван </td>
                </tr>
                <tr>
                <th width="35%"> По батькові </th>
                <td width="65%"> Іванович </td>
                </tr>
                <tr>
                <th width="35%"> Рік та дата
народження </th>
                <td width="65%"> 30.09.1992 </td>
                </tr>
                <tr>
                <th width="35%"> Факультет/Інститут
</th>
                <td width="65%"> Фізико-технічний
інститут </td>
                </tr>
                <tr>
                <th width="35%"> Напрямок та курс
</th>
                <td width="65%"> Інформаційна безпека,
2 курс </td>
                </tr>
                <tr>
                <td colspan="2">
                Призер міжнародної олімпіади з
математики, та лауреат конкурсу МАН з інформатики.
Поєднує
                навчання з роботою,що пов’язана з
розробкою програмно забезпечення в сфері інформаційної
безпеки.
                У вільний час займається тяжкою
атлетикою, подорожує у країни ближнього сходу та
займається живописом.
                Найближчим часом планує опанувати
месетецтво хореографії, а також відновити навички гри
на фортепіано.
                </tr>
                </table>

            </div>
            <div style="position:absolute;left:10px;
```

```
top:400px">
            <H3> Улюблений фільм </H3>
            <video width="400" height="300"
controls="controls" poster="video/duel.jpg">
            <source src="video/duel.ogv"
type='video/ogg; codecs="theora, vorbis"'>
            <source src="video/duel.mp4"
type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
            <source src="video/duel.webm"
type='video/webm; codecs="vp8, vorbis"'>
            Тег video не підтримується вашим браузером.
            <a href="video/duel.mp4">Скачайте
відео</a>.</video>
            <H3> Ульблена музична композиція</H3>
            <audio controls>
            <source src="audio/music.ogg"
type="audio/ogg; codecs=vorbis">
            <source src="audio/music.mp3"
type="audio/mpeg">
            Тег audio не підтримується вашим браузером.
            <a href="audio/music.mp3">Скачайте
музыку</a>.
            </audio>
            </div>
        </div>

        <footer
style="background:#0fdfdf;width:80%;margin:auto">
        <p style="text-align: center">База данних
студентів НТУУ "КПІ" 2011-2014 <i>Copyrighting</i> </p>
</footer>

    </body>
</html>
```

На цій сторінці застосовано **ряд нових дескрипторів**:

<meta> – дескриптор, призначений для збереження інформації, що адресується браузерам, та пошуковим системам. Наприклад інформація про кодування для правильного відображення тексту. В нашому випадку, якщо цього не зробити, кирилиця відображатиметься некоректно.

<title> – дескриптор, що задає заголовок сторінки і не відображається на самій сторінці.

 – дескриптор, що задає блочний елемент, який виділяє фрагмент документа з цілю зміни вигляду вмісту або його логічного відокремлення.

<header> та <footer> – дескриптори, введені з HTML 5 для полегшення роботи пошукових систем та браузерів. Вони принципово нічим не відрізняються для людини від дескриптора , проте для машини має зовсім інше значення.

<style> – використовується для стильового оформелення елементів сторінки. Детальніше про цей дескриптор – в лабораторній роботі, присвяченій CSS.

<video> та <audio> – дескриптори, введені в HTML 5, які дозволяють публікувати медіаконтет на веб-сторінці без допомоги стороніх засобів, таких як технології flash.

<table>, <tr>, <td>, <th> – дескриптори, що застосовуються для оформлення таблиць.

1.3. Порядок виконання роботи

1.3.1. Проаналізувати умову задачі.

1.3.2. Здійнити верстку HTML5-сторінки відповідно до завдання.

1.3.3. Результати роботи оформити протоколом, в якому має бути наведено опис нових елементів HTML5, використаних під час виконання лабораторної роботи.

Важливо: код та протокол не можуть бути ідентичними – такі роботи зараховані не будуть.

1.4. Завдання

1. В лабораторній роботі потрібно створити HTML-сторінку, на якій буде розміщена інформація з історії ФТІ та відомості щодо його випускників (джерелом інформації щодо випускників можуть бути такі соціальні мережі як VK, facebook тощо). Дана сторінка має обов'язково містити елементи сучасного HTML5 (роботи, виконані із використанням лише HTML4, зараховані не будуть).
2. Створити окрему сторінку з HTML-формою для реєстрації випускників, яка має містити поля різних типів (в тому числі і такі, які були введені у форматі HTML5).
3. Із використанням атрибутів HTML5 забезпечити перевірку коректності даних, що вводяться користувачем у форму.

1.5. Контрольні запитання

1. Назвати основні протоколи стеку TCP-IP та їх призначення.

2. Порівняти методи передачі даних: `GET` та `POST`.
3. Навести основні відмінності між HTML5 та стандартами-попередниками.
4. Яку інформацію містить HTML-тег `<meta>`?
5. Для чого призначені HTML-форми?
6. Назвати основні елементи форм в мові HTML.
7. Опишіть роботу HTTP-протоколу, детально розкривши процес отримання HTML-сторінки з веб-серверу.

1.6. Додаткові джерела інформації

1. http://htmlbook.ru/html5
2. http://wisdomweb.ru
3. http://www.w3schools.com/html/html5_intro.asp
4. http://www.w3.org/TR/html5/
5. Фрайн Б. "HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств"

Лабораторна робота №2
Форматування HTML-сторінки із використанням каскадних таблиць стилів CSS

Мета роботи: отримати навички створення каскадних таблиць стилів CSS для HTML-сторінок.

2.1. Теоретичні відомості

Cascading Style Sheets (CSS) – спеціальна мова, що використовується для відображення сторінок, написаних мовою розмітки HTML.

Після того, як в HTML 3.2 були додані такі дескриптори як `<font>`, оформлення та розмітка стали дещо складнішою справою (зміст та стилі в одному документі). Для того, щоб вирішити цю проблему, в кінці 1996 року **W3C** консорціум запропонував CSS.

CSS дозволяє зберігати інформацію щодо оформлення HMTL-документу в окремому зовнішньому файлі з розширенням .css. Редагуючи лише цей файл, стало можливим змінювати оформлення веб-сайту. Коли таблиця стилів знаходиться в окремому файлі (`style.css` наприклад), вона може бути під'єднана до веб-документа шляхом використання дескриптора `<link>`, що знаходиться в цьому документі між дескрипторами `<head>` та `</head>`. Правила таблиці стилів діють протягом всього документу.

```
<!DOCTYPE html>
<html>
   <head>
      .....
      <link rel="stylesheet" href="style.css">
   </head>
   <body>
      .....
   </body>
</html>
```

Також цей файл може бути під'єднаний шляхом використання дерективи `@import`, яку треба розмістити між дескрипторами `<style>` та `</style>`, які знаходяться між `<head>` та `</head>`.

```
<!DOCTYPE html>
<html>
   <head>
      .....
      <style media="all">
         @import url(style.css);
      </style>
   </head>
```

```
</html>
```

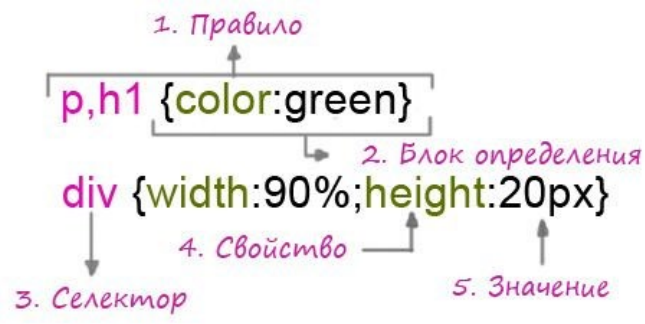
Таблиця стилів може знаходитися безпосередньо в веб-документі як в тілі якогось дескриптора, наприклад (стиль працюватиме лише в межах дескриптора):

```
…..
 <p  style="font-size:  20px;  color:  green;  font-
family: arial, helvetica, sans-serif">
            .....
     </p>
…..
```

Так і окремо від дескрипторів, знаходячись між <style> та </style>:

```
…..
<head>
      .....
      <style>
         body {
            color: red;
         }
      </style>
   </head>
…..
```

Таблиці стилів окремим файлом описують наступним чином. Кожна таблиця складається з *набору правил* (1), кожне правило складається з одного чи кількох *селекторів* (3) та *блоку визначення* (2) . Блок визначення може містити одну чи кілька *властивостей* (4) відокремлених крапкою з комою “ ; ”, причому після останньої властивості крапка з комою не обов’язкова. Кожна властивість має значення записане через двокрапку “ : ”.



Існують наступні види *селекторів* :

Селектор по елементу. Дозволяє відібрати у веб-документі всі однойменні дескриптори. Наприклад змінити колір всіх абзаців <p>:

```
p {
      color:#CC0000;
    }
```

Селектор по класу. Створює клас, який буде використовуватись при виклику його в дескрипторі. Наприклад, потрібно змінити колір тільки перших абзаців. Для цього створимо клас `first` (він задається, починаючи з крапки “.”). Синтаксис :

```
.first{
 color:#CC0000;
    }
```

В веб документі клас викликається наступним чином :

```
<p class ="first">  текст </p>
```

Селектор індетифікатор. Данний селектор застосовуєтсья при потребі виділити один елемент, унікальний до інших елементів в документі. Нехай ім’я селектора `unique`. Задання такого селектора має наступний вигляд :

```
 #unique{
     color:#CC0000;
     }
```

У веб-документі він викликатиметься наступним чином:

```
<p id="unique">  текст </p> s
```

Групові селектори. В імені селектора використовується більше одного імені дескриптора. Наприклад весь текст на сторінці має бути одного кольору. Виберемо всі абзаци і назви розіділів :

```
p, h1{

 color:#вCC0000 ;

   }
```

Також існують селектори **атрибутів**, **контекстний** селектор, селектор **дочірніх елементів**, селектор **псевдокласів** та інші.

Основні параметри шрифту :

1. font-weight: [bold|normal|number] - жирність шрифту
2. font-style: [normal|italic|oblique] - нахил шрифту
3. font-size: number - розмір шрифту
4. font-family: name - гарнітура шрифту
5. color:number - колір шрифту
   er – колір фону
7. background: url – текстурний фон

Псевдокласи посилань

8. A:active{} Таблиця стилів для активних посилань (при натиснені)
9. A:link{} Таблиця стилів для асамих посилань
10. A:visited{} Таблиця стилів для посилань, що вже відвідувались
11. A:hover {} Таблиця стилів для посилань при наведені курсора мишки

Основні параметри абзацу (та елементів типу "Box")

12. text-align: [left|right|center|justify] - вирівнювання
13. text-indent: number – відступ червоної стрічки
14. padding-left: number – відступ від тексту зліва
15. padding-right: number – відступ від тексту справа
16. padding-top: number – відступ від тексту зверху
17. padding-bottom: number - отступ от текста знизу
18. margin-left: number – відступ від границі зліва
    ght: number – відступ від границі справа
20. margin-top: number – відступ від границі зверху
21. margin-bottom: number– відступ від границі знизу

2.2. Приклад

Код HTML файлу

```
<!DOCTYPE html>
<html>
 <head>
     <link rel="stylesheet" href="style.css">
```

```
        <meta charset="utf-8">
        <title> Анкета студента НТУУ "КПІ" </title>
    </head>

    <body bgcolor=blue>
        <header class ="sh">
             Анкета студента НТУУ "КПІ"
        </header>
        <div  class ="main">
            <div  class ="photo">
            <p  style="text-align:  center">  <img>Тут
повина розміщується фото студента  </p>
            </div>
            <div  class ="table">
                <table     border="2"     width="100%"
cellpadding="5">
                    <tr>
                    <th class ="name"> Прізвище </th>
                    <td  class  ="discribe">  Іваненко
</td>
                    </tr>
                    <tr>
                    <th class ="name"> Ім’я </th>
                    <td class ="discribe"> Іван </td>
                    </tr>
                <tr>
                <th class ="name"> По батькові </th>
                    <td  class  ="discribe">  Іванович
</td>
                    </tr>
                <tr>
                <th  class  ="name">  Рік  та  дата
народження </th>
                    <td class ="discribe"> 30.09.1992
</td>
                    </tr>
                <tr>
                <th  class  ="name">  Факультет/Інститут
</th>
                    <td  class  ="discribe">  Фізико-
технічний інститут </td>
                    </tr>
                <tr>
                <th  class  ="name">  Напрямок  та  курс
</th>
```

```
                    <td class ="discribe"> Інформаційна
безпека, 2 курс </td>
                    </tr>
                <tr>
                <td colspan="2">
                Призер    міжнародної    олімпіади    з
математики,  та  лауреат  конкурсу  МАН  з  інформатики.
Поєднує
                навчання  з  роботою,що  пов’язана  з
розробкою програмно забезпечення в сфері інформаційної
безпеки.
                У  вільний  час  займається  тяжкою
атлетикою,  подорожує  у  країни  ближнього  сходу  та
займається живописом.
                Найближчим  часом  планує  опанувати
месетецтво хореографії, а також відновити навички гри на
фортепіано.
                </tr>
                </table>

            </div>
            <div class ="media">
            <H3> Улюблений фільм </H3>
            <video      width="400"      height="300"
controls="controls" poster="video/duel.jpg">
                <source           src="video/duel.ogv"
type='video/ogg; codecs="theora, vorbis"'>
                <source           src="video/duel.mp4"
type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
                <source          src="video/duel.webm"
type='video/webm; codecs="vp8, vorbis"'>
                Тег  video  не  підтримується  вашим
браузером.
                <a      href="video/duel.mp4">Скачайте
відео</a>.</video>
            <H3> Ульблена музична композиція</H3>
            <audio controls>
                <source           src="audio/music.ogg"
type="audio/ogg; codecs=vorbis">
                <source           src="audio/music.mp3"
type="audio/mpeg">
                Тег  audio  не  підтримується  вашим
браузером.
                <a     href="audio/music.mp3">Скачайте
музыку</a>.
```

```
                </audio>
            </div>
        </div>

        <footer class="sf">
        База данних студентів НТУУ "КПІ" 2011-2014
<i>Copyrighting</i>
        </footer>


    </body>

   </html>
```

Код CSS файлу

```
   .sh
   {
   background:yellow;
   height:30px;
   width:100%;
   text-align: center;
   }

   .sf
   {
   background:#0fdfdf;
   width:80%;
   margin:auto;
   text-align: center;
   }

   .main
   {
   position:relative;
   background:white;
   margin:auto;
   height:1000px;
   width:80%;
   margin-top:10px;
   }

   .photo
```

```
{
position:absolute;
left:10px;
top:10px;
background:green;
width:240px;
height:300px;
}

.table
{
position:absolute;
left:260px;
top:10px;
margin-right:10px;
width:745px;
}

.name
{
width:35%;
}

.discribe
{
width:65%;
}

.media
{
position:absolute;
left:10px;
top:400px;
}
```

Таким чином, при потребі змінити текст в кількох блоках, буде достатньо зробити відповідні зміни в `style.css`, а не прописувати вручну всі зміни в html-документі. Крім того, це значно спрощує візуальне сприйняття розмітки сторінки.

2.3. Порядок виконання роботи

2.3.1. Проаналізувати умову задачі.

2.3.2. Відформатувати сторінку, створену під час виконання лабораторної роботи 1, використовуючи каскадні таблиці стилів CSS відповідно до завдання. Каскадні таблиці стилів оформити у вигляді окремого css-файлу. Розроблена таблиця стилів має містити демонстрацію нових можливостей CSS3.

2.3.3. Результати роботи оформити протоколом, в якому має бути наведено опис нових елементів CSS3, використаних під час виконання лабораторної роботи.

Важливо: код та протокол не можуть бути ідентичними – такі роботи зараховані не будуть.

2.4. Завдання

1. Продемонструвати навички створення *селекторів по елементу*, *селекторів по класу* та *групових селекторів* при оформленні вмісту сторінки.
2. Створити випадаюче меню сайту із ефектом активізації пункту меню при наведенні на нього курсору.
3. На сторінці з реєстраційною формою, створеною під час виконання лабораторної роботи №1, додати діалогове вікно, яке міститиме додаткову пояснювальну інформацію та буде виконане із використанням можливостей CSS3 (надалі буде використане у лабораторній роботі №3 для діалогу з користувачем).

2.5. Контрольні запитання

1. Для чого призначені каскадні таблиці стилів CSS?
2. Які існують види селекторів? Зазначити специфіку їх використання.
3. Як можна додати коментар до CSS-коду?
4. Для чого використовується комбінування селекторів?
5. Які 3 способи оголошення стилів можуть бути використані? Вкажіть умови, за яких доцільно застосовувати той чи інший спосіб.
6. Опишіть блокову модель HTML-документів.
7. Для чого в CSS застосовуються умовні коментарі? Наведіть приклад.

2.6. Додаткові джерела інформації

1. http://www.w3schools.com/css/css3_intro.asp
2. http://htmlbook.ru/samcss
3. http://www.wisdomweb.ru/CSS
4. Фрайн Б. "HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств"

Лабораторна робота №3
Мова сценаріїв JavaScript

Мета роботи: отримати навички написання простих сценаріїв JavaScript та використання готових JS-бібліотек.

3.1. Теоретичні відомості

JavaScript – це мова програмування, що дозволяє зробити Web -сторінку інтерактивною, тобто такою що реагує на дії користувача.

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний Web - браузер. Іншими словами, код програми вбудовується в HTML - документ і виконується на боці клієнта. Для виконання програми не потрібно навідь перезавантажувати Web -сторінку, всі програми виконуються в відповідь на будь-яку подію. Наприклад, перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуваним, заборонити відправлення даних.

JavaScript - об'єктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript.

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для надання інтерактивності веб-сторінкам.

Основні архітектурні риси:

- динамічна типізація,
- автоматичне керування пам'яттю,
- прототипне програмування,
- функції як об'єкти першого класу.

Структура мови

Структурно JavaScript можна представити у вигляді об'єднання трьох частин, що чітко різняться одна від одної :

- ядро (ECMAScript),
- об'єктна модель браузера (Browser Object Model або BOM),
- об'єктна модель документа (Document Object Model або DOM).

Об'єктну модель документа іноді розглядають як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежного від мови інтерфейсу документа.

Об'єктна модель браузера

Об'єктна модель браузера - браузероспеціфічна частина мови, яка являється прошарком між ядром і об'єктною моделлю документа. Основне призначення об'єктної моделі браузера - керування вікнами браузера і забезпечення їх взаємодії. Кожне з вікон браузера представляється об'єктом window, центральним об'єктом BOM. Об'єктна модель браузера на даний момент не стандартизована, проте специфікація знаходиться в розробці WHATWG та W3C. Крім управління вікнами, в рамках об'єктної моделі браузера, браузерами зазвичай забезпечується підтримка наступних сутностей:

- керування фреймами;
- підтримка затримки у виконанні коду і зациклювання з затримкою;
- системні діалоги;
- управління адресою відкритої сторінки;
- управління інформацією про браузер;
- управління інформацією про параметри монітора;
- обмежене керування історією перегляду сторінок;
- підтримка роботи з HTTP cookie.

Об'єктна модель документа

Об'єктна модель документа - інтерфейс програмування додатків для HTML і XML-документів. Згідно DOM документом можна поставити у відповідність дерево об'єктів, які мають ряд властивостей, які дозволяють робити з ним різні маніпуляції:

- отримання вузлів;
- зміна вузлів;
- зміна зв'язків між вузлами;
- видалення вузлів.

Область застосування

- Використання на веб-сторінках;
- Розташування всередині сторінки.

Щоб додати JavaScript-код на сторінку, можна використовувати теги <script></script>.

3.2. Приклад

Приклад функції

```
<html>
  <head>
    <title>My First Page</title>
      <script language=JScript>

          function factorial(n)
```

```
            {
              var j, fact = 1;
                for (j = 1; j <= n; j++)
                    fact=fact*j;
              return fact;
            }

        </script>
     </head>
    <body>
    </body>
</html>
```

3.3. Порядок виконання роботи

3.3.1. Проаналізувати умову задачі.

3.3.2. Написати JavaScript-сценарій відповідно до завдання. Продемонструвати використання готових JS-бібліотек для валідації користувацькх даних (завдання 1-3).

3.3.3. Написати JavaScript-сценарій, який демонструє об’єктні можливості JavaScript (завдання 4).

3.3.4. Реалізувати користувацьку функцію відповідно до варіанту (варіант обирається за номером у списку групи, у вирпдку недостатньої кулькості завдань – за модулем номеру у списку групи від кількості варіантів). Функції оформити у вигляді окремого JS-файлу `func.js`

3.3.5. Результати роботи оформити протоколом, який має містити написаний код, відомості про використану готову JS-бібліотеку для валідації користувацьких даних та обєктні можливості JavaScript.

Важливо: код та протокол не можуть бути ідентичними – такі роботи зараховані не будуть.

3.4. Завдання

1. Написати JS-сценарій, який виведе у діалоговому вікні пояснювальну інформацію щодо введення даних до форми, розробленої при виконанні лабораторної роботи №1, у випадку коли користувач протягом деякого інтервалу часу (1 хв, наприклад) не внесе даних до форми – використати запуск JavaScript за розкладом.
2. Здійснити попередню перевірку коректності даних, які вносяться користувачем у форму. Перевірку реалізувати у вигляді окремого JS-

сценарію, для перевірки коректності заповнення полів слід застосувати механіхм регулярних виразів `RegExp`.

3. Реалізувати аналогічну перевірку із використанням бібліотеки jQuery validate.
4. Продемонструвати можливість створення користувацього об'єкту засобами JavaScript. Створити об'єкт `Student` із властивостями `name`, `surname`, `age`, `course` та властивостями `GetOlder()` - зміна віку на вказану кількість років, `ChangeSurname()` - зміна прізвища, `MoveToSecondCourse()` – переведення на наступний курс.
5. Реалізувати користувацьку JS-функцію відповідно до варіанту:
    1) Функція, що виводить перші 20 чисел послідовності Фібоначчі (елемент послідовності визначається як сума двох попередніх). Для виводу результатів використати функцію `document.write()`.
    2) Змінити попереднє завдання таким чином, щоб виводився *n*-й член послідовності Фібоначчі. Використати функцію `prompt()` для введення числа *n*.
    3) Функція, що виводить слова заданого тексту у алфавітному порядку.
    4) Функція, що виводить задану послідовність чисел в оберненому порядку.
    5) Функція, що визначає найбільше значення серед заданих чисел.
    6) Функція, що визначає найменше значення серед заданих чисел.
    7) Вивести таблицю чисел від 5 до 15 та їх другий та третій ступені, використовуючи функцію alert `()`.
    8) Змінити сценарій завдання № 3.3 таким чином, щоб користувач міг вибрати, в якому порядку відсортувати слова (в спадному чи висхідному).
    9) Створити функцію `no_zeros()`, аргументом якої є масив чисел, а результатом дії — модифікований вхідний масив, який не містить нульових значень.
    10) Створити функцію `e_names()`, аргументом якої є масив слів, а результатом дії — кількість слів у вхідному масив, які закінчуються `"ie"` або `"y"`.
    11) Створити функцію `first_vowel()`, аргументом якої є рядок літер, а результатом дії — номер позиції, на якій знаходиться найлівіша гласна літера.
    12) Створити функцію `counter()`, аргументом якої є масив чисел, а результатом дії — номера позицій, на яких знаходяться від'ємні елементи та нулі.
    13) Створити функцію `tst_name()`, аргументом якої є текстовий рядок, а результатом дії — `true`, якщо рядок має вигляд: `string1`, `string2 letter`, де `string1` та `string2` є

слова, що складаються з літер нижнього регістру (за виключенням першої літери слова), `letter` складається з літер верхнього регістру, `false` — в протилежному випадку.

14) Створити функцію `row_averages()`, аргументом якої є масив масивів чисел (двовимірний масив), а результатом дії — масив, елементами якого є середні значення відповідних елементів вхідного масиву масивів чисел.

15) Створити функцію `reverser()`, аргументом якої є текстовий рядок, а результатом дії — відображення вхідного рядку у протилежному порядку.

3.5. Контрольні запитання

1. Призначення JavaScript.
2. Чим JavaScript відрізняється від мови Java?
3. Чи є JavaScript об'єктно-орієнтованою мовою програмування?
4. Яким чином вбудовується код JavaScript в документ HTML?
5. Яким чином JavaScript дозволяє обробляти події?
6. Як можна додати коментар до JavaScript коду?
7. Якими є вимоги до створення та іменування змінних в JavaScript?
8. Як можна оголосити користувацьку функцію? Яким може бути її ім'я?
9. Яка функція називається анонімною? Наведіть приклад.
10. Наведіть перелік основних подій в сценаріях JavaScript. Поясніть за яких умов вони виникають.
11. Опишіть 3 способи створення масивів в JavaScript. Порівняйте особливості роботи з масивами в JavaScript та C++.
12. Чим відрізняються методи `exec()` та `test()`, які використовуються для пошук тексту за певним зразком?
13. Як в JavaScript можна створювати користувацькі об'єкти (назвіть 2 способи)?

3.6. Додаткові джерела інформації

1. http://www.wisdomweb.ru/JS/
2. http://www.w3schools.com/JS/
3. http://www.codecademy.com/en/tracks/javascript
4. Джейсон Ленгсторф "PHP и jQuery для профессионалов"

Лабораторна робота № 4
Використання AJAX (Asynchronous JavaScript and XML)

Мета роботи: отримання студентом інформації щодо основних особливостей методів AJAX, а також отримання навичок щодо створення веб-додатку.

4.1. Теоретичні відомості

AJAX не є технологією за визначенням - це концепція при створенні веб-додатку. Концепція представляє собою використання асинхронних веб-запитів під час роботи сторінки без призупинення виконання іншого JavaScript (далі - JS) коду та блокування інтерфейсу браузера для отримання необхідних даних "на льоту" - без перезавантаження всієї веб-сторінки.

Використання методів AJAX дозволяє отримати максимально чуйний інтерфейс на веб-сторінці, скоротити обсяги переданих даних і прискорити веб-додатки для клієнта. Сьогодні без суб'єкта не обходиться жоден вебсайт.

Всупереч назві, через механізми AJAX можна передавати не тільки XML, але й інші формати даних (HTML, текст, JSON). Останнім часом (на момент 2014) все більше спостерігається тенденція до заміни XML на більш простий в роботі формат JSON (JavaScript Object Notation).

Всього існує кілька методів реалізації цієї ідеї на практиці. Всі вони крім одного пішли в історію, поступившись місцем одному популярному і стандартизованому способу.

Історія

Компанією Microsoft в 1998 році пропонувалося використовувати асинхронні веб-запити для поліпшення чуйності інтерфейсу, використовуючи при цьому Java-аплет, що підключається до веб-сторінці. Також була описана можливість використовувати для цього тег <IFRAME>. В силу труднощів властивих цим двом методам і повільним інтернет-з'єднанням, вони не знайшли належної популярності.

XMLHttpRequest

Пізніше Microsoft для браузера Internet Explorer розробила компонент IXMLHttpRequest, що призначався для забезпечення функціонування їхнього продукту Outlook WebAccess. Пізніше XMLHttpRequest (далі - XHR) був перенесений до складу пакету бібліотек MSXML 2.0, який поставлявся разом з Internet Explorer 5. Ідея такого компонента сподобалася розробникам інших

браузерів. Розробники проекту Mozilla розробили відкриту сумісну реалізацію XHR і впровадили її в браузер Mozilla Suite 0.6. Пізніше цей компонент був привнесений в Opera 8.01, Safari 1.02. Сьогодні XMLHttpRequest є стандартом W3C. Остання версія стандарту - Level 2 (на 2014 рік) має підтримку междоменной запитів (CORS), підтримку інформування про прогрес запиту, механізми роботи з бінарними даними.

Популярність. Масово ж AJAX почав застосовуватися десь після 2005 року, після публікації відомої статті Джесса Гарретта "Ajax: A new approach to Web Applications". У статті був описаний підхід до використання асинхронних запитів для часткового оновлення сторінці на основі XMLHTTPRequest.

Одними з перших AJAX стала застосовувати компанія Google для своїх продуктів Google Mail, Google Maps. Це дозволило дати життя багатьом популярним сьогодні сервісам.

Станом на 2014 рік AJAX використовується майже на будь-якому сайті, якому потрібно якісь запити до віддалених інформаційних баз (пошук, трекінг, інше).

Принципи. У ході роботи JavaScript виникає необхідність отримання актуальних даних сервера без блокування або перезавантаження всієї сторінці. Для досягнення цієї мети, блок коду виробляє дії спрямовані на створення HTTP-запиту і встановлює обробник (event-handler) на подію отримання або помилки в JavaScript (про події в JavaScript можна почитати в попередніх частинах).

Відразу (до моменту передачі запиту за системним мережевим стеком) відбувається повернення в зухвалу функцію, виконання коду JavaScript може бути продовжено до отримання відповіді від сервера.

Способи реалізації

- Основний - використання XHR
- Використання прихованих <iframe>
- Додавання тегів <script>
- Інші (через сторонні плагіни) - Java, Adobe Flash і тп.

Будемо розглядати основний використовуваний сьогодні метод - XMLHttpRequest

Об'єкт XMLHttpRequest надається браузером, його поведінка є повністю документованою. Розглянемо техніку роботи з ним.

Створення об'єкта. Наведемо загальну функцію для створення об'єкта XHR. Ця функцій перебере всі можливі реалізації об'єкта і поверне коректну.

Властивості та функції. Розглянемо функції та властивості об'єкта

(method, URL, async, userName, password) - визначає основні параметри запиту
– рядок - вказує метод ("GET", "POST")
– рядок - URL запиту
– прапор - використовувати асинхронний запит (true - для асинхронности)
– рядок, необов'язково - логін для HTTP-авторизації
– рядок, необов'язково - пароль для HTTP-авторизації
(content) – виробляє відправку. Тут content - вміст запиту, що відправляється.
() – скасовує поточний запит
() – повертає всі заголовки HTTP-відповіді від сервера у вигляді рядка
(headerName) – повертає вміст заголовка HTTP-відповіді, або null у разі відсутності заголовка
(label, value) – встановлює заголовок HTTP-запосіли
- рядок - назва заголовка
- рядок - вміст заголовка
(mimeType) – встановлює MIME-тип для HTTP-відповіді вручну. Метод не сумісний з ранніми версіями Internet Explorer.

Властивості:
- обробник події - викликається при зміні стану готовності документа
- число - стан готовності (0 - не ініціалізовано, 1 - відкритий, 2 - відправка, 3 - отримання, 4 - завантаження завершена)
- рядок - текст відповіді на запитів
- XML документ - подання відповіді у вигляді XML-об'єкта (тип Document)
- число - HTTP-код відповіді від сервера
- рядок - назва стану ("Not Found", "OK", інше).

Послідовність використання.

Для відправлення запиту та успішної обробки слід провести наступну

послідовність дій:

22. Створити XMLHttpRequest (дивіться вище)
23. Встановити базові параметри через open () - URL, метод
24. Встановити необхідні заголовки через setRequestHeader ()
25. Встановити обробник події onreadystatechange
26. Провести відправку вмісту через функцію send ()

Після успішного виконання запиту Ви зможете отримати відповідь використовуючи властивість responseText. Перевірити поточний стан можна використовуючи властивість readyState. Після успішного завершення readyState повинен бути рівний 4. Нижче наведено приклад використання.

ЗАУВАЖЕННЯ!

У браузері Internet Explorer спостерігається проблема кешування запитів. Для вирішення проблеми або додавайте набір випадкових символів в кінець URL при кожному запиті, або у відповіді сервера встановіть заголовки HTTP:

: ТЕКУЩАЯ ДАТА GMT

Робота з XML

Не менш важливою частиною є обробка XML-даних в JavaScript. Розглянемо методи навігації по прийнятому через AJAX XML документу і процес вилучення з нього даних.

Цей розділ дуже важливий, так сьогодні безліч служб повертає відповіді у форматі даних XML, тому розуміння механізмів роботи з ним дозволить швидше відкинути зайві питання. Розуміння механізмів представлення XML-дерева в JavaScript дозволить краще зрозуміти сенс моделі DOM.

Розглянемо XML документ такого змісту. Припустимо кількість і зміст

піделементів <message> може варіюватися, але їх формат затверджений і залишається постійним.

Очевидно, що для отримання будь-яких корисних даних потрібно отримати кореневий елемент <messages>. Кожен елемент XML документа представляється об'єктом типу Node, для якого характерний певний набір властивостей:

- абсолютний базовий URI вузла
- дочірні вузли
- перший дочірній вузол
- останній дочірній вузол
- ім'я вузла
- тип вузла
- значення вузла
- містить посилання на батьківський документ
- батьківський вузол
- текстове вміст сайту

Для представлення документа XML використовується розширення типу Node - тип Element. У нього до властивостей Node додається кілька цікавих нам властивостей і методів.

- представляє назву тега елемента
(name) - повертає значення атрибута елемента
(name) - перевіряє наявність атрибута
() - перевіряє наявність дочірніх вузлів
() - вибирає серед нащадків вузли із заданим тегом і повертає у вигляді колекції.

Текстові елементи XML документа представляються як композиція Element і текстового Node всередині. Тому, при необхідності взяти текстовий вміст всередині тега, необхідно звернутися до першого дочірньому вузлу Element'a.

Наведемо приклад навігації по такому нашому документу:

У функцію як параметр msg необхідно передати раніше отриманий документ з xhr.responseXML. console.log - висновок в консоль розробника. Як бачимо, в навігації немає нічого складного.

Покладемо messagesNode - кореневий вузол - перший вузол з повертаної колекції функцією getElementsByTagName.

Отримуємо колекцію повідомлень використовую функцію getElementsByTagName, викликану на об'єкті messagesNode - кореневому вузлі.

Перевіряємо наявність колекції вузлів і виробляємо обхід використовуючи цикл for.

На кожній ітерації циклу ми отримуємо поточний елемент <message> і його дочірні елементи <text>, <sender>, <date>. Виходячи з зауваження, для отримання текстового вмісту, беремо їх перші дочірні елементи (childNodes [0] або firstChild).

Читач не буде здивований, коли дізнається, що всі елементи DOM мають аналогічні перерахованим вище властивості. До елементів DOM також застосовні зазначені прийоми навігації.

JQUERY AJAX

У бібліотеці jQuery реалізовано інтерфейс для роботи з асинхронними запитами, який суттєво спрощує життя веб-розробника. Сам інтерфейс jQuery для роботи з Ajax складається з наступних функцій:

() - основний метод роботи з Ajax
() - для відправки GET запитів
() - для відправки POST запитів

- асинхронність запиту, за замовчуванням true
- вкл / викл кешування даних браузером, за замовчуванням true

- тип даних, що відправляються MIME - за замовчуванням «application / x-

- передані дані - рядок або об'єкт (аналогічно тому, що передавалося в send)
- фільтр для вхідних даних
- тип даних, що повертаються в callback функцію (xml, html, script, json, text,

- тригер - відповідає за використання глобальних AJAX Event'ов, за замовчуванням true
- тригер - перевіряє чи були зміни у відповіді сервера, щоб не слати ще запит, за замовчуванням false
- перевстановити ім'я callback функції для роботи з JSONP (за замовчуванням генерується на льоту)
- за замовчуванням відправляються даний загортаються в об'єкт, і відправляються як «application / x-www-form-urlencoded», якщо треба інакше – відключаємо.
- кодування для скриптів - актуально для JSONP і підвантаження JavaScript'ов
- час таймаут в мілісекундах
- метод запиту - GET або POST
- URL запитуваної сторінки

ПОДІЇ

$.ajax має кілька подій:

- спрацьовує перед відправкою запиту
- якщо сталася помилка
- якщо помилок не виникло
- спрацьовує по закінченню запиту

4.2. Приклад

Переведемо наш запит з першого розділу на jQuery:

Як бачимо, запити через jQuery слати набагато простіше!

$ .get () і $ .post ()

Ці два методи є обгорткою для функції $.ajax (). Вони відсилають запит методом GET і POST відповідно.

Сигнатура виклику:

Використання цих функції дозволяє скоротити обсяг коду, але повною функціональністю $.ajax вони не володіють.

4.3. Порядок виконання роботи

.3.1. Проаналізувати умову задачі.
.3.2. Виконати завдання згідно з п. 4.4.
.3.3. Результати роботи оформити протоколом.

4.4. Завдання

1. Створити програму, що обробляє дані від форми реєстрації, що було створено в межах 1-2 лабораторних робіт. Забезпечити різноманітність варіантів реагування програми в залежності від даних форми, що відправляються. Забезпечити реагування на всі можливі помилки заповнення форми або інші помилки.

1.1. Розв’язати задачу двома способами та виконати порівняльний аналіз швидкодії цих способів та порівняльний аналіз синхронного та асинхронного методу обробки запитів. При необхідності використати багаторазове виконання потрібних операцій.

- шляхом використання стандартних об’єктів JS.
- шляхом використання можливостей JQuery.

2. Виводити дані необхідно на HTML-сторінку в броузері (echo, printf).

3. Забезпечити перевірку коректності введення даних в формі, а також забезпечити захист від передачі в веб-формі спеціальних символів.

4.5. Контрольні запитання

1. Що таке AJAX? Засоби реалізації.
2. Призначення технології AJAX
3. Переваги та недоліки AJAX.
4. Що таке XmlHttpRequest?
5. Відмінності JSON від XML.
6. Переваги, недоліки та принципи використання кожної з технологій.

4.6. Додаткові джерела інформації

Лабораторна робота № 5
Використання PHP

Мета роботи: отримання студентом інформації щодо основних особливостей мови програмування PHP, а також отримання навичок щодо створення простих програм.

5.1. Теоретичні відомості

PHP (створений у вигляді пре виконувача гіпертекстів — **H**ypertext розробки серверної частини Web. Якщо брати до уваги його простий синтаксис подібний до C і підтримку різних баз даних, об'єктно-орієнтованого підходу і можливість використовувати безкоштовно, буде зрозуміло, чому PHP користується великою популярністю серед WEB-розробників.

Виконання сценаріїв PHP при клієнтському запиті

Для звичайного відображення простої WEB-сторінки створеної за допомогою HTML відправляється запит на WEB – сервер, який у свою чергу виконує по черзі так дії:

- Аналіз HTTP-запиту.
- Пошук на сервері потрібної WEB-сторінки.
- Відправка на машину клієнта відповідних даних.

Якщо дана сторінка має вбудовані PHP – сценарії, то послідовність виконання дій значно ускладнюється. WEB - сервер змушений буде перевіряти та аналізувати PHP файл, і, якщо будуть знайдені відповідні PHP – ідентифікатори, буде викликаний інтерпретатор.

Послідовність дій буде мати такий вигляд:

- Аналіз HTTP-запиту
- Пошук на сервері потрібної WEB-сторінки
- Аналіз сценаріїв знайдених на сторінці за допомогою інтерпретатора і їх виконання
- Відправка на машину клієнта відповідних даних

Після отримання даних на клієнтській сторінці не буде PHP включень. Вони будуть замінені на відповідні результати HTML – коду або інші фрагменти клієнтських сценаріїв (JavaScript, JQuery та інші).

Включення фрагментів PHP в HTML - код

Включити код на сторінку HTML можливо за допомогою :

1. Стандартних дескрипторів

2. Коротких дескрипторів
3. Дескрипторів типу script
4. Дескрипторів типу ASP

Стандартні дескриптори

PHP – код

Короткі дескриптори
<? PHP – код ?>

Дескриптори script
<script language="php">
<?php PHP – код ?>

Дескриптори ASP
<% PHP – код %>

Змінні

В PHP всі змінні починаються з $ і далі пишеться ім'я змінної. PHP регістро-залежна мова отже змінні написані з великої і маленької букви будуть різними.
Наприклад:

```
$username = 'Barry';
$UserName = 'White';
```

Неправильний вигляд будуть мати так змінні :

Суперглобальні змінні. Отримання даних сервером

Відправляючи дані через веб-форму, ми передаємо в скрипт, що вказано в параметрі **action**, певні значення. В залежності від значення параметру **method** можна розрізняти POST- та GET- методи відправлення даних.

В залежності від цього, у скрипті, в який передаються дані, ці дані приймаються через доступ до суперглобальних змінних **$_GET[""]** та **$_POST[""]**.

```
echo 'Привіт, ' . htmlspecialchars($_GET["name"]) . '!';
```

В цьому випадку мається на увазі, що користувач відправляв дані в формі методом **GET**, або ввів в броузері адресу **http://example.com/?name=Vasya** Результатом виконання даного прикладу буде наступне:

Привіт, Vasya!

Аналогічно здійснюється звернення до змінної **$_POST[“”]**.

Наведемо приклад веб-форми, що відправляє запит на скрипт серверу.

```
<form name=”form1” method=”POST” action=”script.php”>
    <input type=”text” name=”name” />
    <input type=”text” name=”age” />

type=”submit” value=”Відправити” />
```

Відправлені дані в скрипті можуть бути оброблені наступним чином:

```
echo 'Доброго дня, ' . htmlspecialchars($_POST['name']);
echo 'Вам ' . (int)$_POST['age'] . ' років.';
```

Типи даних

В PHP є різні вбудовані типи даних:

- Цілий
- Логічний
- З плаваючою точкою
- Масив
- Клас
- Тип NULL
- Рядковий

Тип змінної визначається після її введення чи визначення її вмісту. Визначаються вони через оператор $var.

Наприклад :

```
$var = 2;            //$var – ціле число 2
$var = “2”;          //$var -  рядок що вміщує 2
$var = array(2);     //$var – массив що вміщує
                     //один елемент 2
```

Рядковий тип

В PHP рядок — це послідовність байтів. Якщо в рядку містяться оператори PHP, то при аналізі інтерпретатором під них буде підставлятися їх значення у даний момент. Також не можна у строках використовувати \ та \\. Такі символи використовуються для екранування, отримання, наприклад, переносу на новий рядок — “\n”.

Наприклад:
‘The age is: $age’
“The age is: 12 //Якщо $age буде мати значення 12

Масиви

В PHP масиви можна створити 2 способами:

Прості багатовимірні масиви

Узагальнений синтаксис елементів багатомірного простого масиву:

```
$ім'я[індекс1][індекс2]..[індексN];
```

Приклад простого багатовимірного масиву:

```
$arr[0][0]="Овощи";
$arr[0][1]="Фрукты";
$arr[1][0]="Абрикос";
$arr[1][1]="Апельсин";
$arr[1][2]="Банан";
$arr[2][0]="Огурец";
$arr[2][1]="Помидор";
$arr[2][2]="Тыква";
```

В PHP індексом масиву можу бути не тільки число а й рядок. Такі рядки не мають ніяких обмежень. Вони можуть містити пробіли та мати будь - яку довжину.

Спеціальний тип NULL

Спеціальне значення NULL каже що змінна не має значення. І єдине можливе значення цього типу — NULL.

Змінна є NULL якщо:

- Їй була присвоєна константа NULL
- Їй не було надано ніякого значення
- Вона була видалена за допомоги unset()

Синтаксис типу NULL:

Оператори

Для виконання дій зі змінними існують різні види операторів.
Оператори бувають різних видів:

- Арифметичні оператори.

Приклад	Назва	Результат
-$a	Заперечення	Зміна знака $a.
$a + $b	Додавання	Сума $a та $b.
$a - $b	Віднімання	Різниця $a та $b.
$a * $b	Множення	Добуток $a и $b.
$a / $b	Ділення	Остача від ділення $a на $b.
$a % $b	Ділення по модулю	Цілочислова остача від ділення $a на $b.

- Логічні оператори

Приклад	Назва	Результат
$a and $b	Логічне 'і'	**TRUE якщо** і $a, і $b **TRUE**.
$a or $b	Логічне 'або'	**TRUE якщо** або $a, або $b **TRUE**.
$a xor $b	Виключне 'або'	**T**
! $a	Заперечення	**TRUE якщо** $a не **TRUE**.
$a &&$b	Логічне 'і'	**TRUE якщо** і $a, і $b **TRUE**.
$a || $b	Логічне 'або'	**TRUE якщо** або $a, або $b **TRUE**.

- Рядкові оператори

Перший оператор: '.' - оператор конкатенації(об'єднує праву та ліву частину рядків).

Другий оператор: '.=' - оператор присвоєння конкатенацією

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

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

- Побітові оператори

Приклад	Назва	Результат

$a & $b	Побітове 'і'	Встановлюються ті біти, що встановлені і в $a, і в
$a | $b	Побітове 'або'	Встановлюються ті біти, що встановлені або в $a, або в $b.
$a ^ $b	Виключне 'або'	Встановлюються ті біти, що встановлені або тільки в $a, або тільки в $b.
~ $a	Заперечення	Встановлюються ті біти, що невстановлені в $a
$a << $b	Зсув вліво	Всі біти змінної $a зсуваються на $b позицій вліво (кожна позиція має на увазі 'множення на 2')
$a >> $b	Зсув вправо	Всі біти змінної $a зсуваються на $b позицій вправо (кожна позиція має на увазі 'множення на

Краще не використовувати зсув на 32 біти. Або не зсувати вправо, щоб отримати числа більше 32 біти.

- Оператори присвоєння

Базовий оператор присвоєння '='. Даний оператор вказує, що лівий операнд отримує значення правого виразу.

$a = ($b = 4) + 5; // результат: $a отримує значення 9, змінній $b присвоено значення 4.

Для всіх арифметичних бітових та строкових операторів можливе використання деякого значення в виразі ,а потім встановлення його, як результат деякого виразу.

$a += 5; // встановлює $a значення 8, аналогічно запису: $a = $a + 5;

$b .= "There!"; // встановлює $b рядком "Hello There!", як і $b = $b . "There!";

- Оператори порівняння

Прикла	**Назва**	**Результат**

д		
$a == $b	Рівно	**TRUE якщо** $a рівно $b.
$a === $b	Тотожно рівно	**TRUE якщо** $a рівно $b і має той самий тип.
$a != $b	Не рівно	**TRUE якщо** $a не рівно $b.
$a <>$b	Не рівно	**TRUE якщо** $a не рівно $b.
$a !== $b	Тотожно не рівно	**T** **R**
$a <$b	Менше	**T**
$a >$b	Більше	**T**
$a <= $b	Менше або рівно	**TRUE якщо** $a менше або рівно $b.
$a >= $b	Більше або рівно	**TRUE якщо** $a більше або рівно $b.

Вираз *(expr1) ? (expr2) : (expr3)* інтерпретується як *expr2*, якщо *expr1* вираховується **TRUE**, або як *expr3* якщо *expr1* вираховується **FALSE**.

- Оператори інкременту та декременту

Приклад	**Назва**	**Результат**
++$a	Префіксний інкремент	Збільшує $a на одиницю і повертає значення $a.
$a++	Постфіксний інкремент	Повертає значення $a, а потім збільшує $a на 1.
--$a	Префіксний декремент	Зменшує $a на одиницю і повертає значення $a.
$a--	Постфіксний декремент	Повертає значення $a, а потім зменшує $a на 1.

Змінні логічного типу інкременту та декременту не підлягають!

- Оператор еквівалентності

Базовий оператор еквівалентності '===='. PHP терпимо відноситься до неявного перетворення строк в числа і навпаки.

```
if($a==$b) echo "a і b рівні";// Виведе на екран "a и b равні"
```

Якщо використати оператор еквівалентності, то:

```
if($a===$b) echo "a і b рівні"; // Нічого не виведе на екран
```

Тобто оператор еквівалентності крім значень порівнює ще й типи змінних. За допомогою нього можна порівняти також масиви, об'єкти, класи.

- Операції з символьними змінними

Р
Н
Р До символьних змінних можна застосовувати лише операцію інкременту, а операцію декременту не можна. Тобто 'Z' + 1 = 'AA' в PHP. А в PERL. Тобто 'Z' + 1 = '[' в C : 'Z' + 1 = 91.

/* Результат роботи буде наступний:

Пріоритети операторів

Пріоритет	Оператор	Порядок виконання
13	(постфікс)++ (постфікс)--	зліва направо
12	++(префікс) --(префікс)	справа наліво
11	* / %	зліва направо
10	+ -	зліва направо
9	<<>>	зліва направо
8	<<= >>=	зліва направо
7	== !=	зліва направо
6	&	зліва направо
5	^	зліва направо
4	|	зліва направо
3	&&	зліва направо

2	||	зліва направо
1	= += -= *= /= %= >>= <<== &= ^= |=	справа наліво

Основні конструкції мови PHP :

- Yvовні оператори (if, else);
- Цикли (while, do-while, for, foreach, break, continue);
- Конструкції вибору (switch);
- Конструкції оголошення (declare);
- Конструкції повернення значень (return);
- Конструкції включень (require, include).

Функції

Особливості функцій користувача в PHP:

- Доступні параметри за замовчуванням. Є можливість викликати одну й ту саму функцію з змінним числом параметрів;
- Функції користувача можуть повертати будь-який тип;
- Область видимості змінних усередині функції є ієрархічною (деревоподібною);
- Є можливість змінювати змінні, передані в якості аргументу.

Основний недолік користувача функцій PHP пов'язаний з областю видимості функцій.

Для PHP всі оголошені і використовувані у функції змінні за замовчуванням локальні для функції. Тобто, за умовчанням немає можливості змінити значення глобальної змінної в тілі функції.

Якщо ви в тілі функції користувача будете використовувати змінну з ім'ям, ідентичним імені глобальної змінної (що знаходиться поза користувача функції), то ніякого відношення до глобальної змінної ця локальна змінна мати не буде. У даній ситуації в функції користувача буде створена локальна змінна з ім'ям, ідентичним імені глобальної змінної, але доступна дана локальна змінна буде тільки всередині цієї функції користувача.

Для позбавлення від наведеного недоліку, в PHP існує спеціальна інструкція global, що дозволяє функції користувача працювати з глобальними змінними.

Створення функції

Функція користувача може бути оголошена в будь-якій частині програми (скрипту), до місця її першого використання. І не потрібно ніякого попереднього оголошення, як в інших мовах програмування, зокрема, в C. Переваги застосовуваного в PHP підходу в наступному.

Дійшовши до визначення користувача функції, транслятор перевірить коректність визначення і виконає трансляцію визначення функції у внутрішнє представлення, але транслювати сам код він не буде. Синтаксис оголошення функцій наступний:

```
function Имя (аргумент1[=значення,...,аргумент2=значення2]) {
    тіло_функції
```

Оголошення функції починається службовим словом **function**, потім слідує ім'я функції, після імені функції - список аргументів у дужках. Тіло функції знаходиться у фігурних дужках і може містити будь-яку кількість операторів.

Вимоги, що пред'являються до імен функцій:

- Імена функцій можуть містити російські літери, але давати функцій імена, що складаються з російських букв не рекомендує;
- Імена функцій не повинні містити пробілів;
- Ім'я кожної користувацької функції повинно бути унікальним. При цьому, необхідно пам'ятати, що регістр при оголошенні функцій і зверненні до них не враховується. Тобто, наприклад, функції funct () і FUNCT () мають однакові імена;
- Функцій можна давати такі ж імена, як і змінним, тільки без знаку $ на початку імен.

Типи значень, що повертаються користувача функціями, можуть бути будь-якими. Для передачі результату роботи користувача функцій в основну програму (скрипт) використовується конструкція **return**. Якщо функція нічого не повертає, конструкцію **return** не вказують. Конструкція **return** може повертати все, що завгодно, у тому числі і масиви. Приклад функції користувача:

Класи і об'єкти в PHP

Клас – це базове поняття в об'єктно-орієнтованому програмуванні (ООП). Класи утворюють синтаксичну базу ООП. Їх можна розглядати як свого роду "контейнери" для логічно пов'язаних даних і функцій (зазвичай званих методами - див. Нижче). Якщо сказати простіше, то клас - це своєрідний тип

даних.

Примірник класу – це об'єкт. Об'єкт – це сукупність даних (властивостей) і функцій (методів) для їх обробки. Властивості і методи називаються членами класу. Взагалі, об'єктом є все те, що підтримує інкапсуляцію.

Якщо клас можна розглядати як тип даних, то об'єкт – як змінну (по аналогії). Скрипт може одночасно працювати з декількома об'єктами одного класу, як з декількома змінними.

Усередині об'єкту дані і код (члени класу) можуть бути або відкриті, або ні. Відкриті дані і члени класу є доступними для інших частин програми, які не є частиною об'єкта. А ось закриті дані і члени класу доступні тільки усередині цього об'єкта.

Опис класів в PHP починаються службовим словом **class**:

```
class ім'я_класу {
// Опис членів класу - властивостей і методів для їх обробки
```

Для оголошення об'єкта необхідно використовувати оператор new:

Об'єкт = **new** ім'я_класу;

Дані описуються за допомогою службового слова var. Метод описується так само, як і звичайна призначена для користувача функція. Методу також можна передавати параметри.

Підіб'ємо проміжні підсумки: оголошення класу має починатися з ключового слова class (аналогічно, як оголошення функції починається з ключового слова function). Кожному оголошенню властивості, що міститься в класі, має передувати ключове слово var. Властивості можуть відноситися до будь-якого типу даних, підтримуваних в PHP, їх можна розглядати як змінні з невеликими відмінностями. Після оголошень властивостей слідують оголошення методів, дуже схожі на типові оголошення користувача функцій.

За загальноприйнятим правилам імена класів ООП починаються з прописної букви, а всі слова в іменах методів, крім першого, лише з великої літери (перше слово починається з малої літери). Зрозуміло, ви можете використовувати будь-які позначення, які вважаєте зручними; головне - виберіть стандарт і дотримуйтеся його.

КОНСТРУКТОРИ

PHP 5 дозволяє оголошувати методи-конструктори. Класи, у яких оголошено метод-констуктор, викликатимуть цей метод при кожному створенні нового об'єкта, так що це може виявитися корисним, щоб, наприклад, ініціалізувати який-небудь стан об'єкта перед його використанням. Конструктор, раніше совпадавший з назвою класу, тепер необхідно оголошувати як **__construct()**, що дозволить легше переміщати класи в ієрархіях. Конструктори в класах-батьках не викликаються автоматично. Щоб

викликати конструктор, оголошений в батьківському класі, слід звернутися до методу **parent::__construct ()**.

```
print "Конструктор класу BaseClass\n";
```

```
print "Конструктор класу SubClass\n";
```

Раніше створення об'єкта і ініціалізація властивостей виконувалися роздільно. Конструктори дозволяють виконати ці дії за один етап.

Цікава подробиця: в залежності від кількості переданих параметрів можуть викликатися різні конструктори. У розглянутому прикладі об'єкти класу Webpage можуть створюватися двома способами. По-перше, ви можете викликати конструктор, який просто створює об'єкт, але не ініціалізує його властивості:

По-друге, об'єкт можна створити за допомогою конструктора, визначеного в класі, - в цьому випадку ви створюєте об'єкт класу Webpage і привласнюєте значення його властивості bgcolor:

деструктори

PHP 5 надає концепцію деструкторів, подібну з тими, що застосовуються в інших ГО мовах, таких, як Java: коли звільняється останнє посилання на об'єкт, перед вивільненням пам'яті, займаній цим об'єктом, викликається метод

```
print "Конструктор\n";
```

```
print "Винищується " . $this->name . "\n";
```

Як і у випадку з конструкторами, деструктори, оголошені в батьківському класі, чи не будуть викликані автоматично. Для виклику деструктора, оголошеному в класі-батьку, слід звернутися до методу **parent::__destruct()**.

5.2. Приклад

Приклад класу на PHP:

```
* Визначення MyClass
```

```
echo $obj->public; // Працює
```

```
echo $obj->protected; // Фатальна помилка
echo $obj->private; // Фатальна помилка
$obj->printHello(); // Показує Public, Protected та Private


 * Визначення MyClass2


    // Ми можемо перевизначити public та protected, але не private


echo $obj->public; // Працює
echo $obj2->private; // Не визначено
echo $obj2->protected; // Фатальна помилка
$obj2->printHello(); // Показує Public, Protected2, але не Private
```

Покажчик $this можна також використовувати для доступу до методів, а не тільки для доступу до даних:

5.3. Порядок виконання роботи

5.3.1. Проаналізувати умову задачі.
.3.2. Виконати завдання згідно з п. 5.4.
.3.3. Результати роботи оформити протоколом.

5.4. Завдання

1. Створити програму, що обробляє дані від форми реєстрації, що було створено в межах 1-2 лабораторних робіт. Забезпечити різноманітність варіантів реагування програми в залежності від даних форми, що відправляються. Забезпечити реагування на всі можливі помилки заповнення форми або інші помилки.

2. Створити клас **User**, що міститиме основні властивості: Прізвище, Ім'я, По-батькові, дату народження; а також основні методи: Отримання Прізвища І.П., отримання дати народження, отримання віку людини на основі дати народження, запис Прізвища, Імені, По-батькові (редагування).

3. Створити клас **Student**, що наслідує клас **User** та містить такі основні властивості на доданок до наслідуваних: Факультет, Кафедру та групу у форматі “XX-99”, а також нові методи окрім наслідуваних — отримання ріку вступу та номеру підгрупи на основі даних про групу в заданому форматі.

4. Виводити дані необхідно на HTML-сторінку в броузері (echo, printf).

5. Забезпечити перевірку коректності введення даних в формі, а також забезпечити захист від передачі в веб-формі спеціальних символів.

5.5. Контрольні запитання

1. Що таке PHP?
2. Які відмінності PHP від інших мов програмування?
3. Особливості іменування змінних в PHP.
4. Назвіть логічні оператори.
5. Що таке асоційований масив? Принцип роботи з ним.
6. Функції в PHP. Іменування, створення.
7. Типи даних, які використовує PHP, особливості роботи з типами даних.

5.6. Додаткові джерела інформації

Лабораторна робота № 6
Реляційні бази даних MYSQL

Мета роботи: вивчення основ роботи реляційних баз даних на прикладі СУБД MySQL, основних інструкцій для створення запитів до бази, проектування розподілених Web-застосувань.

6.1. Теоретичні відомості

MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на MsSQL, та з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Визначення та приклади

Фундаментальним поняттям реляційної моделі даних є поняття **відношення**.

Атрибут відношення є пара виду <Імя_атрібута: ім'я_домена>.

Імена атрибутів повинні бути унікальні в межах відношення. Часто імена атрибутів відношення збігаються з іменами відповідних доменів.

Відношення, визначене на множині доменів (не обов'язково різних), містить дві частини: заголовок і тіло.

Заголовок відношення містить фіксовану кількість атрибутів відношення:

Тіло відношення містить множину кортежів відношення. Кожен кортеж відношення є множиною пар виду <Ім’я_атрибуту: Значення_атрибуту>, таких що значення атрибуту належить домену.

Реляційної базою даних називається набір відношень.

Схемою реляційної бази даних називається набір заголовків відношень, що входять в базу даних.

Хоча будь-яке відношення можна зобразити у вигляді таблиці, потрібно чітко розуміти, що відношення не є таблицями. Це поняття близькі, але такі, що не збігаються. Відмінності між відношеннями і таблицями будуть розглянуті нижче.

Терміни, якими оперує реляційна модель даних, мають відповідні "табличні" синоніми:

Реляційний термін термін

База даних

Схема бази даних

Відношення

Заголовок відношення
Тіло відношення
Атрибут відношення
Кортеж відношення
Ступінь (-арність) відношення
Потужність відношення
Домени і типи даних
Відповідний "табличний"
Набір таблиць
Набір заголовків таблиць
Таблиця
Заголовок таблиці
Тіло таблиці
Найменування стовпця таблиці
Рядок таблиці
Кількість стовпців таблиці
Кількість рядків таблиці Типи даних в комірках таблиці
Наведемо приклад, для більш предметного розуміння.

Приклад. Розглянемо відношення "Співробітники" задане на доменах "Номер_співробітника", "Прізвище", "Зарплатня", "Номер_відділу". Так як всі домени різні, то імена атрибутів відношення зручно назвати так само, як і відповідні домени. Заголовок відношення має вигляд:

Працівники (Номер_співробітника, Прізвище, Зарплатня, Номер_відділу)
Нехай в даний момент відношення містить три ***кортежи***:
(1, Іванов, 1000, 1)
(2, Петров, 2000, 2)
(з, Сидоров, 3000, 1)
таке відношення природним чином представляється у вигляді таблиці:

	Номер_співробітника	Прізвище	Зарплатня	Номер_відділу
1	Іванов	1000		1
2	Петров	2000		2
3	Сидоров	3000		1

Для роботи з реляційними базами даних використовується запитів до БД — мова структурованих SQL (Structured Query Language), основні елементи якої буде

розглянуто дещо далі в даних методичних вказівках. Для роботи із відношеннями в SQL реалізовано наступні типи даних.

Типи даних в MYSQL

Цілі числа. Загальний вигляд вказівки типу даних:
префікс **INT** [UNSIGNED]
Необов'язковий прапор UNSIGNED задає, що буде створено поле для зберігання беззнакових чисел (великих чи рівних 0).
Типи цілих чисел:

- **TINYINT** Може зберігати числа від -128 до 127
- **SMALLINT** Діапазон від -32 768 до 32 767
- **MEDIUMINT** Діапазон від -8388608 до 8388607
- **INT** Діапазон від -2147483648 до 2147483647
- **BIGINT** Діапазон від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

Дробові числа

Точно так само, як цілі числа поділяються в MySQL на кілька різновидів, MySQL підтримує і кілька типів дробових чисел.

У загальному вигляді вони записуються так:

Ім'я Типа [(length, decimals)] [UNSIGNED]

де length - кількість знаків (ширина поля), в яких буде розміщено дробове число при його передачі.
decimals - кількість знаків після десяткової точки, які будуть враховуватися.
Типи дробових чисел:

- **UNSIGNED** - задає беззнакові числа.
- **FLOAT** Число з плаваючою точкою слабкий точності.
- **DOUBLE** Число з плаваючою точкою подвійної точності.
- **REAL** Синонім для DOUBLE.
- **DECIMAL** Дробове число, що зберігається у вигляді рядка.
- **NUMERIC** Синонім для DECIMAL.

Рядки

Рядки являють собою масиви символів. Зазвичай при пошуку за текстовим полям за запитом SELECT не береться до розгляд регістр символів, тобто рядка "Вася" і "ВАСЯ" вважаються однаковими.
Типи Рядків:

- **VARCHAR** Може зберігати не більше 255 символів.
- **TINYTEXT** Може зберігати не більше 255 символів.
- **TEXT** Може зберігати не більше 65 535 символів.
- **MEDIUMTEXT** Може зберігати не більше 16777215 символів.
- **LONGTEXT** Може зберігати не більше 4294967295 символів.

Найчастіше застосовується тип TEXT, але якщо ви не впевнені, що дані

не будуть перевищувати 65 536 символів, використовуйте LONGTEXT.

Бінарні дані

Бінарні дані - це майже те ж саме, що і дані у форматі TEXT, але тільки при пошуку в них враховується регістр символів.
Типи бінарних даних:

- **TINYBLOB** Може зберігати не більше 255 символів.
- **BLOB** Може зберігати не більше 65 535 символів.
- **MEDIUMBLOB** Може зберігати не більше 16777215 символів.
- **LONGBLOB** Може зберігати не більше 4294967295 символів.
- **BLOD** - дані не перекодуються автоматично, якщо при роботі з встановленим з'єднанням включена можливість перекодування тексту "на льоту".

Дата і час

MySQL підтримує кілька типів полів, спеціально пристосованих для зберігання дат і часу в різних форматах.

Типи "Дата і час":

- **DATE** Дата у форматі YYYY -MM -DD
- **TIME** Час у форматі HH: MM: SS
- **DATETIME** Дата і час у форматі YYYY -MM -DD HH: MM: SS
- **TIMESTAMP** Дата і час у форматі timestamp. Однак при отриманні значення поля воно відображається не у форматі timestamp, а у вигляді YYYYMMDDHHMMSS, що сильно применшує переваги його використання в PHP

Модифікатори

До типу даних можна приєднати модифікатори, які задають його "поведінку" і ті операції, які можна (або, навпаки, заборонено) виконувати з відповідними стовпцями. Типи Модифікаторів:

- not null - Чи означає, що поле не може містити невизначене значення, тобто полі обов'язково має бути ініціалізованих при вставці нового запису в таблицю (якщо не задано значення за замовчуванням).

- primary key - Відбиває, що поле є первинним ключем, тобто ідентифікатором запису, на який можна посилатися.

- auto_increment - При вставці нового запису поле отримає унікальне значення, так що в таблиці ніколи не існуватимуть два поля з однаковими

номерами.

- default - Задає значення за замовчуванням для поля, яке буде використано

CREATE TABLE tel_numb (fio text, address text DEFAULT 'Не вказаний', tel

Створення бази даних

Створення бази даних виконується за допомогою команди CREATE DATABASE.
Синтаксис команди:

де database_name - Ім'я, яке буде присвоєно створюваній базі даних.

У наступному прикладі ми створимо базу даних db_test:

Видалення бази даних mysql

Для видалення бази даних використовується команда DROP DATABASE.
Синтаксис:

де database_name - задає ім'я бази даних, яку необхідно видалити.
У наступному прикладі ми видалимо базу даних db_test:

Створення таблиці MYSQL

Створення таблиці проводиться командою CREATE TABLE.
Синтаксис:
CREATE TABLE table_name (column_name1 type, column_name2 type, ...) де table_name - ім'я нової таблиці;
column_name - імена колонок (полів), які будуть присутні в створюваній таблиці.
type - визначає тип створюваної колонки.

Припустимо, нам треба створити таблицю телефонних номерів друзів.
Наша таблиця буде складатися з трьох стовпців: ПІБ друга, адреса і телефон:
CREATE TABLE tel_numb (fio text, address text, tel text)

Видалення таблиці MYSQL

Видалення таблиці проводиться командою DROP TABLE
Синтаксис:

де table_name - ім'я видаляється таблиці.

У наступному прикладі ми видалимо таблицю tel_numb:

Внесення змін у структуру таблиці MYSQL

Перейменування таблиці можна зробити за допомогою наступної конструкції:

де table_name_old - старе ім'я таблиці, яке нам потрібно перейменувати;
table_name_new - нове ім'я таблиці.

Вставку нового стовпця можна здійснити за допомогою наступної конструкції:

де table_name - ім'я таблиці, в яку буде вставлений новий стовпець;
field_name - ім'я стовпчика, що вставляється;
parametrs - параметри, що описують вставляється стовпець.
Обов'язковим параметром є вказівка типу даних.

Змінити властивості одного або декількох стовпців можна за допомогою наступної конструкції:
CHANGE field_name_old field_name_new parametrs де table_name - ім'я таблиці, в якій знаходиться змінюваний стовпець;
field_name_old - ім'я стовпця змінюваного стовпчика;
field_name_new - нове ім'я змінюваного шпальти (повинна дорівнювати field_name_old, якщо ми не хочемо поміняти ім'я шпальти);
parametrs - нові параметри стовпця.

У наступному прикладі встановимо тип рядка field_1 як текст:

У разі, якщо треба змінити властивості відразу декількох стовпців, то конструкцію **CHANGE** field_name_old field_name_new parameters повторюємо через кому для кожного шпальти:

Видалення стовпця можна зробити за допомогою наступної конструкції:

де table_name - ім'я таблиці, в якій буде видалено стовпець;
field_name - ім'я стовпця, що видаляється.

Вставка рядків в таблиці MYSQL

Вставка запису здійснюється командою INSERT INTO

Дана команда додає в таблицю table_name запис, у якої поля, що позначені як field_nameN, встановлені в значення contentN.

Ті поля, які небули перераховані в команді вставки, отримують "невизначені" значення (невизначене значення - це не порожній рядок, а просто ознака, який говорить MySQL, що у даного поля немає ніякого значення).

Треба відзначити, що якщо при створенні таблиці поле було відзначено прапором NOT NULL, і воно при вставці запису отримало невизначене значення, то MySQL поверне помилку.

При вставці в таблицю бінарних даних (або текстових, містять апострофи і слеші) деякі символи повинні бути захищені Зворотний слеш, а саме, символи:

- \
- ‘ •

і символ з нульовим кодом.

Видалення рядків в таблиці MYSQL

Видалення запису здійснюється командою DELETE FROM
DELETE FROM table_name WHERE (вираз)

Ця команда видаляє з таблиці table_name всі записи, для яких виконано вираз. Вираз - це просто логічне вираження.

Наприклад нам треба видалити запис з таблиці, яка містить ПІБ, адреса і телефон: **DELETE FROM** tel_numb **WHERE** (fio = 'Вася Пупкін') або, якщо треба видалити за кількома параметрами
DELETE FROM tel_numb WHERE (fio = 'Вася Пупкін' && tel = '23-45-45 ')

У виразі, крім імен полів, констант і операторів, можуть також зустрічатися найпростіші обчислювані частини, наприклад: (id <10 +4 * 5).

Оновлення записів в таблиці MYSQL

Оновлення запису здійснюється командою UPDATE
field_name2 = 'var2', ... WHERE (вираз)

Дана команда для всіх записів в таблиці table_name, що задовольняють висловом вираз, встановлює зазначені поля field_nameN в значення varN.

Цю команду зручно застосовувати, якщо не потрібно оновлювати не всі поля якийсь запису, а потрібно відновити лише деякі.

Пошук записів в таблиці MYSQL

Пошук записів здійснюється командою SELECT
(вираз) [ORDER BY field_name [DESC] [ASC]]

Ця команда шукає всі записи в таблиці table_name, які задовольняють висловом вираз.

Якщо записів декілька, то при вказівці реченні order by вони будуть відсортовані по тому полю, ім'я якого записується правіше цього ключового

слова (якщо задано слово desc, то упорядкування відбувається в зворотному порядку). В пропозиції order by можуть також задаватися кілька полів.

Особливе значення має символ *. Він наказує, що з відібраних записів випливає витягти всі поля, коли буде виконана команда отримання вибірки.

Наприклад, нам треба знайти в таблиці, яка містить повідомлення в гостьовій книзі, всі записи, які залишив певний користувач.

// Db_guest - ім'я таблиці, яка містить повідомлення, залишені в гостьовій

Але що робити, якщо шуканий текст не займає все поле, а є частиною цього поля (наприклад, при пошуку слова або словосполучення в загальному масиві тексту)?

// Search - містить шуканий тест

Якщо кількість входжень рядка **$search** в полі **content** більше 0 (тобто воно взагалі є), то запис додається до решти знайдених записів.

Сортування по будь-якому стовпчику здійснюється за допомогою конструкції **ORDER BY**:

Time - стовпець, що містить час написання запису

// У форматі " UNIX timestamp "

Тобто дані були відсортовані за зменшенням.

Якщо нам треба відсортувати дані за зростанням, треба замість ключового слова DESC застосувати ASC:

Для вибору кількості записів із відповідними параметрами використовується функція **COUNT**()

Можливість роботи з MYSQL

Через консоль

Робота з MySQL завжди починається з авторизації. Потрібно передати логін і пароль параметрами командного рядка.

Таким чином, ви відкриваєте сеанс роботи з MySQL сервером.

Так ви можете переглянути список усіх баз даних:

Не забувайте ставити крапку з комою в кінці кожної команди, це говорить серверу, що команда закінчена.

Переключитися на потрібну базу даних можна, використовуючи команду USE:

Після цього можна виконувати запити до БД, наприклад, командою SELECT:

Вийти з сеансу роботи ще простіше, ніж зайти:

Далі наведено декілька прикладів роботи через консоль.

Експорт бази даних в файл

Імпорт бази даних з файлу

Запуск MySQL запиту з консолі

Через систему адміністрування

Більш популярним засобом взаємодії із СУБД є система адміністрування СУБД MySQL - **phpMyAdmin** (версія 4.2.11 :: http://www.phpmyadmin.net/home_page/index.php), що можна встановити окремо, а можна — разом із стандартним пакетом для розробника Денвер. Від наочно дозволяє працювати з БД.

Взаємодія php з MYSQL

Для більш детального розгляду, розберемо приклад взаємодії PHP з СУБД. Для роботи нам необхідно буде з'єднатися із сервером баз даних (для чого треба мати користувацький запис, що має певні права), обрати відповідну базу даних, вже після чого виконувати необхідні запити до БД.

Слід звернути особливу увагу на обробку даних, отриманих в результаті запиту до БД. Обробка здійснюється за допомоги інструментів роботи з масивами — foreach ($key as $value), що перебирає всі значення масиву за ключем.

6.2. Приклад

```
// З'єднуємося з сервером баз даних

// У разі помилки з'єднання видаємо помилку
or die('Не удалось соединиться: ' . mysql_error());
echo 'Соединение успешно установлено';
// Обираємо базу даних
```

```
// У разі помилки з'єднання видаємо помилку or die(Н удалось выбрать базу данных');
// Створюємо змінну із текстом SQL-запиту

// Виконуємо цей запит
y) or die('Запрос не удался: ' . mysql_error());
// Виводимо результати в html
// відкриваємо таблицю

// циклом отримуємо масив записів, що є результатом роботи попередньої функції while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {

// формуємо асоційований масив та виводимо його в html




// Звільняємо пам'ять від результатів

Закриваємо з'єднання
```

6.3. Порядок виконання роботи

- Проаналізувати умову задачі.
- Виконати завдання згідно з п. 6.4.
- Результати роботи оформити протоколом.

6.4. Завдання

ВАЖЛИВІ ЗАУВАЖЕННЯ!

1. При наявності бажання та/або додаткових напрацювань в даній лабораторній роботі можна розробити та здавати проекти розподіленої системи на іншій мові програмування, відмінній від PHP (C#, Node.JS, Python тощо). Головне - щоб у вашому рішенні була клієнтська та серверна частина, а також серверна СУБД. Перед початком роботи або здачею таких робіт прошу всіх **попередньо погодити це з викладачем**.
2. Для обраної теми або системи розробити та надати разом з програмним кодом **діаграму прецедентів** (Use Case Diagram), **діаграму класів** (Class Diagram) та **діаграму розгортання** (Deployment Diagram). Для отримання додаткової інформації по цим питанням можна скористуватись книгою Крега

Лармана «Применение UML 2.0 и шаблонов проектирования» або консультацією викладача.

Щодо конкретного завдання та обрання таблиць БД — проконсультуватися із викладачем відповідно до обраної теми. Роботу із БД виконати в межах завдання із використанням мови програмування PHP.

1. Підготувати SQL-запит, що створюватиме базу (**CREATE DATABASE**) даних відповідно то тематики сайту, а також створити необхідні таблиці (**CREATE TABLE**) з відповідними параметрами. В таблицях, де це є необхідним, вказати первинні ключі з автоінкрементом.
2. Створити SQL-запити до бази, що додають дані у кожну з таблиць бази даних, використовуючи інструкцію **INSERT**.
3. Внести зміни в базу даних, використовуючи інструкцію **ALTER**, додавши поле до однієї з таблиць. Заповнити всі значення доданого поля значенням за замовчуванням, використовуючи інструкцію **UPDATE**.
4. Вибрати всі значення із таблиць (**SELECT**) із різними модифікаціями:
    1. Вибрати всі значення з таблиці.
    2. Вибрати перші 10 значень з таблиці, відсортовані у зростаючому порядку за одним із полів
    3. Вибрати кількість значень із таблиць.
5. Видалити одну із таблиць із використанням інструкції **DELETE**.

6.5. Контрольні запитання

1. Що таке SQL?
2. Які типи даних використовуються в MySQL?
3. Які види запитів до СУБД існують?
4. Використання інструкції SELECT.
5. Використання інструкції INSERT.
6. Використання інструкції UPDATE.
7. Використання інструкції DELETE.
8. Яким чином здійснюється сортування результатів запиту?

6.6. Додаткові джерела інформації

1. http://progbook.ru/bd/mysql/
2. http://progbook.ru/bd/mysql/541-velling-mysql-uchebnoe-posobie.html