МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»

МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ
ДО ЛАБОРАТОРНИХ РОБІТ З КУРСУ

ТЕХНОЛОГІЇ РОЗРОБКИ WEB-ДОДАТКІВ

для студентів 5-го курсу інженерно-технічного факультету
спеціальності «комп’ютерні системи та мережі»

Ужгород – 2016

Методичні вказівки і завдання до лабораторних робіт

з курсу «Технології розробки web-додатків»

для студентів 5-го курсу інженерно-технічного факультету

спеціальності «комп’ютерні системи та мережі»

Укладачі: Чернявська Х.М. – ст. викладач кафедри комп’ютерних систем та мереж.

Відповідальний за випуск – Король І.Ю., канд. фіз.-мат. наук, доцент, завідувач кафедри комп’ютерних систем та мереж.

Методичні вказівки розглянуто та схвалено на засіданні кафедри комп’ютерних систем та мереж, протокол №5 від 28 січня 2016р. та методичної комісії інженерно-технічного факультету, протокол №1 від 05 лютого 2016р.

ЗМІСТ

Лабораторна робота №1

Тема: Створення статичних html-сторінок.

Мета: Ознайомитись з структурою html-документу. Навчитись використовувати основні теги форматування тексту, додавання зображень та гіперпосилань. Ознайомитись з використанням атрибутів. Навчитися використовувати мову розмітки гіпертексту html для структурованого подання різних видів текстової і графічної інформації.

Теоретичні відомості

Практично кожна веб-сторінка, на яку Ви дивитесь, написана мовою HTML. Можна розглядати HTML як скелет, який задає структуру сторінки. В даній лабораторній роботі Ви навчитеся використовувати HTML для додавання абзаців, заголовків, списків, таблиць, зображень та посилань.

HTML розшифровується як Hypertext Markup Language. Hypertext означає "текст з посиланнями".

Структура HTML-документу

Мова розмітки є мовою програмування, покликаною перетворити текст на щось більше: HTML може перетворити текст в зображення, посилання, таблиці, списки і багато іншого.

Для створення першої html-сторінки нам достатньо скористатись звичайний редактором «Блокнот», або ж більш зручними безкоштовними редакторами – Notepad++ або PSPad, які мають вбудовану підсвітку коду та підсказки, що зробить процес створення сторінки більш зручним. Зберігати створені файли потрібно з розширенням .html, а для перегляду користуватись наявним браузером.

Перше, що ми повинні зробити, це створити «скелет» сторінки:

```
<!DOCTYPE html>
<html>
</html>
```

a) Завжди пишіть `<! DOCTYPE HTML>` в першому рядку. Це вказує браузеру, якою мовою він повинен читати дану сторінку (в даному випадку, HTML).
b) Завжди використовуйте `<html>` у наступному рядку. Цей елемент вказує на початок HTML документу.

c) Завжди використовуйте </html> в останньому рядку. Цей рядок вказує на завершення HTML-документу.

Щоб дізнатися більше про HTML, ми повинні розібратись в термінології мови HTML. Ви вже помітили, що у мові HTML часто використовуються < >. Те, що знаходиться всередині цих дужок, називається тегами. Теги майже завжди використовуються парами: відкриваючий тег та закриваючий:

Приклад відкриття тега: <html>

Приклад закриття тега: </html>

Ви можете сприймати теги як круглі дужки: щоразу, коли ви відкриваєте тег, ви повинні закрити його. Теги можна вкладати один в оден, тому ви повинні закрити їх в правильному порядку: останній відкритий тег повинен бути першим закритим, як у наведеному нижче прикладі.

```
<1ий тег> <2ий тег> Текст </2ий тег> </1ий тег>
```

Весь HTML-документ повинен розміщуватись між відкриваючим тегом <html> та закриваючим </html>.

HTML-документ містить дві частини: шапку (голову) та тіло. Шапка може містити різну інформацію про Ващ документ, наприклад, його назву. Всередині тіла документу описуються всі елементи, які повинні бути відображені в браузері. Для опису шапки використовується тег <head>, а для назви документу - <title>. Тіло документу розміщується всередині тегів <html>, відразу після тегів <head>:

```
<html>
  <head>
    <title>Моя веб-сторінка</title>
  </head>
  <body>
  </body>
</html>
```

Структуроване відображення тексту

Найпростішими тегами для структурованого відображення тексту є тег абзацу та теги заголовків. Кожен абзац починається з відкриваючого тегу та закінчується закриваючим тегом . Аналогічним чином використовуються теги заголовків різних рівнів (від першого до шостого) - <h1>, <h2>, … , <h6>.

Для прикладу додамо в наш документ заголовок першого рівню та два абзаци:

```
<html>
  <head>
    <title> Моя веб-сторінка</title>
  </head>
  <body>
    <h1>Заголовок першого рівню</h1>
      <p>Це перший абзац, описаний мовою HTML</p>
    <p>Це другий абзац, описаний мовою HTML </p>
  </body>
</html>
```

Для вирівнювання абзаців та заголовків використовується атрибут `align`, який може приймати наступні значення: `left`, `right`, `center`, `justify`. Наприклад:

```
<p align=left>Параграф вирівняний по лівому краю</p>
<p align=right>Параграф вирівняний по правому краю</p>
```

Розміщення графічних зображень

Для того, щоб додати зображення на сторінку, використовується тег `<img>`, обов'язковим атрибутом якого є атрибут `src`, який вказує на адресу, за якою розміщене зображення. Ця адреса може бути відносною, тобто вказувати на розміщення зображення відносно файлу `html`, або може вказувати на `URL`-адресу зображення, розміщеного в мережі. Наприклад, такий рядок добавить на нашу сторінку зображення, розміщене в теці "images":

```
<img src="images/gl_3.jpg"/>
```

А такий запис додасть на сторінку зображення логотипу УжНУ, яке розміщене на сайті університету:

```
<img src= "http://www.uzhnu.edu.ua/images/layout/uzhnu_logo2.png"/>
```

Атрибут `alt` дозволяє задати текстовий рядок, що відображається замість графічного зображення в тих випадках, коли браузер не може показувати графічні зображення чи коли така можливість відключена. Атрибути `height` та `width` задають ширину та висоту зображення відповідно, а атрибут `border` – товщину рамки навколо зображення в пікселях.

Організація списків

Списки зустрічаються в документах дуже часто. Звичайно списки бувають нумерованими і ненумерованими (неупорядкованими). В останньому випадку кожен елемент списку виділяється яким-небудь символом.

Теги `<ul>` і `</ul>` позначають початок і кінець ненумерованого списку `(unnumbered list)`, кожному елементу якого передує тег `<li> (list item)`. Кожен елемент списку виводиться з нового рядка і позначається символом (за замовчуванням жирною чорною крапкою). Змінити цей символ можна за допомогою атрибуту `type`, який може мати наступні значення: `disk` – чорна жирна крапка, `circle` – кружечок, `square` – маленький чорний квадрат.

Розглянемо приклад неупорядкованого списку, використовуючи різні значення параметру `type`.

```
<ul>
      <li type=disk>перший рядок
      <li type=circle>другий рядок
      <li type=square>третій рядок
</ul>
```

Для створення нумерованих списків призначений тег `<ol>`, який необхідно використовувати разом з тегом `</ol>`. Нумерований список створюється аналогічно неупорядкованому списку. Кожен рядок у списку повинний бути відзначений тегом `<li>`.

Для тегу `<ol>` можна вказувати наступні параметри:

`start` - початковий номер для списку

`type` - тип нумерації (`A` - прописними літерами, `a` - малими літерами, `I` - прописними римськими цифрами, `1` - арабськими цифрами)

Таблиці в документах HTML

Дуже часто в документі HTML потрібно розміщати табличні дані. Для цього існують спеціальні теги. Вони відкривають широкі можливості: можна використовувати рамки навколо всіх чи тільки деяких комірок і рядків таблиці, створювати таблиці, що мають заголовки і підписи, розміщати в осередках таблиці текст, графіку чи довільні об'єкти, використовувати як фон для комірок растрові графічні зображення.

У тексті документа HTML таблиця визначається між тегами <table> і </table>. У найпростішому випадку рядки таблиці обмежені тегами <tr> і </tr>, а стовпці - тегами <td> і </td>, наприклад:

```
<TABLE>
      <TR>
           <TD>123</TD><TD>234</TD><TD>345</TD>
      </TR>
      <TR>
           <TD>523</TD><TD>634</TD><TD>745</TD>
      </TR>
</TABLE>
```

Для всіх трьох тегів, що використовуються для формування таблиць, визначений досить великий перелік атрибутів, які дозволяють змінювати розміри комірок, об'єднувати їх, змінювати вирівнювання, розміри, кольори. Ознайомтесь з цими атрибутами самостійно.

Використання посилань

Основна особливість мови HTML полягає в можливості зв'язку окремих частин тексту й ілюстрацій з іншими документами. Гіпертекстові посилання виділяються в тексті документа спеціальним кольором (і/чи підкресленням), і, активізовуючи їх мишею, можна переміщатися по документах або їх частинах.

Посилання створюється за допомогою тегу <a>, що використовується в парі з тегом . Між ними розташовується текст посилання, що відображається у вікні браузера. Атрибут href вказує на url-адресу посилання.

Приклад посилання на інший документ:

```
<a href="http://www.uzhnu.edu.ua"> Офіційний сайт УжНУ
</a>
```

Якщо довжина документа велика, є сенс організувати посилання на його окремі логічні частини, розташувавши їх, наприклад, на початку документа. Такі посилання називаються локальними.

Нехай необхідно створити документ з ім'ям book.html, що складається з декількох розділів. На початку документа бажано розмістити зміст, що містить посилання на окремі розділи.

Насамперед визначимо на початку кожної глави локальні мітки. Для цього необхідно використовувати тег <a> з атрибутом name, як показано нижче:

```
<h2><a name="chap1">Перший розділ</a></h2>
```

```
        . . .
        Зміст розділу
        . . .
<h2><a name="chap2">Другий розділ</a></h2>
        . . .
        Зміст розділу
        . . .
```

Посилання на створені в такий спосіб локальні мітки виконуються за допомогою тега <a>, що має параметр href:

```
<a href="book.htm#chap1">Перший розділ</a>
<a href="book.htm#chap2">Другий розділ</a>
```

Контрольні питання

1. Які елементи містить html-документ?
2. Чим відрізняються парні теги від непарних?
3. Які елементи html-документа є обов'язковими?
4. Які списки і як саме можна подавати в html-документах?
5. Які елементи використовуються для логічного форматування тексту?
6. Як вставити в html-документ графічні зображення?
7. Яким чином формуються таблиці в мові розмітки html?

Завдання для виконання лабораторної роботи:

Завдання 1. Ознайомитися з теоретичним матеріалом про призначення гіпертексту, області його застосування і засоби подання гіпертекстової інформації).

Завдання 2. Вивчити синтаксис мови html, елементи, що використовуються, і їхні атрибути. Для отримання додаткової інформації рекомендується використовувати інформаційні ресурси Інтернет.

Завдання 3. Виконати навчальні приклади, результати зберегти.

Завдання 4. Створіть власну html-сторінку, в якій використайте всі елементи, з якими ви ознайомились в лабораторній роботі. Тематику оберіть на власний розсуд (улюблена країна, місто, вид спорту, вид творчості, хоббі, та інше).

Завдання 5. Відповісти на контрольні питання, відповіді включити в звіт по лабораторній роботі.

Лабораторна робота №2

Тема: Використання каскадних таблиць стилів.

Мета: Ознайомитись з способами підключення каскадних таблиць стилів. Навчитись використовувати селектори класів, ідентифікаторів, комбіновані селектори та псевдокласи гіперпосилань. Набути навики налаштування стилів для різнотипних елементів: тексту, фону, списків та інших.

Теоретичні відомості

Звичайний HTML дозволяє задавати колір і розмір тексту за допомогою атрибутів або тегів, які використовуються для форматування. Якщо знадобиться змінити параметри однотипних елементів на сайті, доведеться переглядати код всіх сторінок, щоб знайти всі елементи та змінити їхні властивості.

Каскадні таблиці стилів (Cascading Style Sheets, CSS) дозволяють зберігати колір, розміри тексту і інші параметри в стилях. Стилем називається набір правил форматування, які застосовується до елементів документа, щоб швидко змінити його зовнішній вигляд.

Стилі дозволяють однією дією застосувати нові атрибути до всіх однотипних елементів сторінки. З їх допомогою можна, наприклад, змінити вигляд всіх заголовків. Якщо потрібно швидко змінити зовнішній вигляд тексту, досить змінити параметри стилю, і вигляд тесту зміниться автоматично на всіх сторінках.

Ще одна вагома перевага CSS полягає в тому, що стилі надають набагато більше можливостей для форматування, ніж простий HTML. Крім того, стилі можуть зберігатися в зовнішньому файлі, браузер кешує такі документи, тому завантаження сайту буде відбуватися трохи швидше.

CSS представляють собою потужну систему для розробників сайтів, розширюючи їх можливості по дизайну і верстці веб-сторінок.

Підключення CSS

Таблиці стилів можуть бути додані на веб-сторінку трьома способами, які розрізняються за своїми можливостями:

Таблиці зв'язаних стилів

Найпотужніший і найзручніший спосіб визначення стилів і правил для сайту. Стилі зберігаються в окремому файлі, який може бути використаний для будь-яких веб-сторінок. Для підключення таблиці зв'язаних стилів використовується тег `<link>` в блоці `<head>` сторінки.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Підключення зв&rsquo;язаних стилів</title>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
    <link rel="stylesheet" type="text/css" href="http://mysite.ua/my.css">
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

Шлях до файлу зі стилями може бути як відносним, так і абсолютним.

Переваги даного способу:

1. Використовується один файл зі стилем для будь-якої кількості веб-сторінок, також можливо його застосовувати на інших сайтах.
2. Можна змінювати таблицю стилів без модифікації веб-сторінок.
3. При зміні параметрів стилю в одному файлі, стиль автоматично застосовується до всіх сторінок, де є на нього посилання. Вказавши розмір шрифту в одному місці, ми змінюємо стиль для всіх веб-сторінок сайту.
4. Файл із стилем при першому завантаженні поміщається в кеш на локальному комп'ютері, окремо від веб-сторінок, тому завантаження сайту відбувається швидше.

Таблиці глобальних стилів

Стиль визначається в самому html-документі і зазвичай розташовується в заголовку веб-сторінки. За своєю гнучкості і можливостям цей спосіб gslrk.xtyyz стилю поступається попередньому, але також дозволяє розміщувати всі стилі в одному місці (в даному випадку безпосередньо в тілі документа). Визначення стилю задається тегом `<style>`.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Підключення глобальних стилів</title>
```

```
    <style type="text/css">
      H1 {
        font-size: 120%; /* розмір шрифту */
        font-family: Verdana, Arial, Helvetica, sans-serif;
        color: #336; /* колір тексту */
       }
    </style>
  </head>
  <body>
    <H1>Hello, world!</H1>
  </body>
</html>
```

В даному прикладі задається стиль для заголовку `<h1>`. На веб-сторінці тепер достатньо вказати тільки цей тег і стилі будуть додані до нього автоматично.

Внутрішні стилі

Внутрішній стиль (inline-стилі) є по суті заданням параметрів для одиночного тегу. Для визначення стилю використовується атрибут `style`, а його значення зазначаються за допомогою синтаксису мови таблиці стилів.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Використання внутрішніх стилів</title>
  </head>
  <body>
    <H1 style="font-size: 120%; font-family: Verdana, Arial,
Helvetica, sans-serif; color: #336">Hello, world!</H1>
  </body>
</html>
```

Рекомендується використовувати внутрішній стиль лише для одиночних випадків або відмовитися від його використання взагалі, оскільки зміна стилю декількох елементів стає проблематичною. Внутрішні стилі не відповідають ідеології структурованого документа, коли вміст і його оформлення розділені.

Всі описані методи використання CSS можуть застосовуватися як самостійно, так і в поєднанні один з одним. В цьому випадку необхідно пам'ятати про їх ієрархію. Першим завжди застосовується внутрішній стиль, потім таблиця глобальних стилів і в останню чергу таблиця пов'язаних стилів. У наступному прикладі використовуються відразу два методи додавання таблиць стилів в документ.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Стили</title>
    <style type="text/css">
      H1 { font-size: 120%;
           font-family: Arial, Helvetica, sans-serif;
           color: green; }
    </style>
  </head>
  <body>
    <H1 style="font-size: 36px; font-family: Times, serif;
color: red;">Hello, world!</H1>
    <H1>Hello, world!</H1>
  </body>
</html>
```

У наведеному прикладі перший заголовок задається червоним кольором розміром 36 пікселів, а наступний - зеленим кольором і іншим шрифтом.

Комбінуючи підключення стилів різними способами, слід знати пріоритетність каскадних таблиць стилів:

1. Внутрішні стилі (вбудовані за допомогою атрибута `style` безпосередньо в теги документа) - найвищий пріоритет, будуть застосовані браузером в будь-якому випадку, навіть якщо виникає конфлікт з впровадженими або зовнішніми стилями;
2. Глобальні стилі (перераховані в тезі-контейнері `<style>` в «голові» документа) - трохи менший пріоритет, будуть застосовуватися у всіх випадках, крім випадків виникнення конфлікту з inline-стилями (при виникненні такого конфлікту будуть застосовані inline-стилі);
3. Зв'язані стилі (стилі, приєднані до `html`-файлу за допомогою тега `<link>`) - найменший пріоритет, будуть застосовані тільки після того, як браузер переконається у відсутності аналогічних правил у всіх інших типах стилів.

Таким чином, розглянувши три способи підключення стилів, та знаючи, в якій послідовності браузер аналізує таблиці стилів, можна підключати одну загальну зв'язану таблицю для всіх сторінок сайту і при цьому гнучко управляти стилями окремих сторінок і окремих елементів на сторінці. Саме з цією особливістю і пов'язане слово «каскадні» в назві `CSS (cascading style`

sheets) - кілька таблиць стилів, приєднаних до html-файлу, проходять через аналізатор (браузер) таким собі «каскадом» в порядку їх пріоритетності.

Властивості тексту

За допомогою CSS можна визначати стиль і вид тексту. Зміна вигляду шрифту і його розміру відбувається через властивості CSS, які описані в наступній таблиці:

Властивість	Значення	Опис	Приклад
font-family	Назва шрифту	Задає список шрифтів	P {font-family: Arial, serif}
font-style	normal italic	Звичайний шрифт Курсив	P {font-style: italic}
font-weight	normal lighter bold bolder 100–900	Нормальна жирність світле накреслення напівжирний жирний 100 - світлий шрифт, 900 - самий жирний	P {font-weight: bold}
font-size	normal pt px %	нормальний розмір пункти пікселі проценти	font-size: normal font-size: 12pt font-size: 12px font-size: 120%

Крім зміни параметрів шрифте, можна управляти і властивостями всього тексту. Значення властивостей тексту наведені в наступній таблиці:

Властивість	Значення	Опис	Приклад
line-height	normal множитель значение %	Міжрядковий інтервал	line-height: normal line-height: 1.5 line-height: 12px line-height: 120%
text-decoration	none underline overline line-through blink	Відключити все оформлення Підкреслення Лінія над текстом Перекреслення Миготіння тексту	text-decoration: none

text-transform	none capitalize uppercase lowercase	Відключити всі ефекти Почати з прописних літер Всі прописні всі рядкові	text-transform: capitalize
text-align	left right center justify	Вирівнювання тексту	text-align: justify
text-indent	значення %	Відступ першого рядка	text-indent: 15px; text-indent: 10%

Властивості кольору

CSS має кілька опцій для визначення кольору тексту і фонових областей на веб-сторінці, основні з яких перераховані в таблиці:

Властивість	Значення	Опис	Приклад
color	Колір	Встановлює колір тексту	P { color: #330000 }
background-color	Колір transparent	Колір фону	BODY { background-color: #6699FF }
background-image	URL none	Фоновий малюнок	BODY { background-image: url (bg.gif) }
background-repeat	repeat repeat-x repeat-y no-repeat	Повтор фонового малюнку	BODY { background-image: url (bg.gif) background-repeat: repeat-y }
background-attachment	scroll fixed	Прокручування фонового малюнку разом з текстом	BODY { background-image: url (bg.gif) background-attachment: fixed }
background-position	Відсотки Пікселі top center bottom	Початкове положення фонового малюнку	BODY { background-position: left top }

	left right		

Встановлювати колір можна трьома способами:

1. По його назві (браузери підтримують деякі кольори по їх назві):

```
P {
 color: navy; /* Колір тексту */
 background-color: yellow; /* Колір фону */
}
```

2. По шістнадцятковому значенню (колір можна встановлювати по його шістнадцятковому значенні):

```
H1 { color: #fc0; }
P {
   color: #F9E71A;
   background-color: #98560F;
  }
```

Також допустимо використовувати скорочений запис, на зразок `#fc0`. Кожен символ дублюється, в результаті отримаємо `#ffcc00`.

3. З використанням RGB:

```
P {
   color: RGB(249, 231, 16);
   background-color: RGB(85%, 24%, 5%);
  }
```

Можна визначити колір, використовуючи значення червоної, зеленої та синьої компоненти в десятковій системі числення. Значення кожного з трьох кольорів може набувати значень від 0 до 255. Також можна задавати колір в процентному відношенні.

Псевдокласи гіперпосилань CSS - link, visited, active, hover

Псевдоклас дозволяє враховувати різні умови або події при визначенні властивостей `HTML` тегу. Наприклад посилання специфікуються в `HTML` тегом `<a>`, який, можна розглядати як селектор `CSS` a:

```
a { color: blue; }
```

Посилання може мати різні стани. Для цих станів існують відповідні псевдокласи.

Псевдоклас `:link` використовується для посилань на сторінки, які користувач ще не відвідував.

Псевдоклас `:visited` використовується для посилань на сторінки, які користувач відвідав.

Псевдоклас `:active` використовується для активних посилань. Активним посилання стає при натисненні на нього.

Псевдоклас `:hover` використовується для посилань, над якими знаходиться покажчик миші.

Зазвичай використання псевдокласу `link` має той же ефект, що і застосування стилю до селектора `A`. Тому цей псевдоклас можна опустити.

Це можна використовувати для створення цікавих ефектів. Наприклад, якщо ми хочемо, щоб посилання ставали помаранчевими і курсивними при наведенні покажчика миші на них, то наше CSS правило повинно виглядати так:

```
a: hover {
color: orange;
font - style: italic;
}
```

Одне з найбільш популярних застосувань CSS - це приховування підкреслення у посилань. Часто робиться, що при наведенні курсору, посилання стає підкресленим, міняє свій колір або використовується і той і інший ефект одночасно. Приклад налаштування таких стилів наведено в наступному прикладі:

```
A:link {
 text-decoration: none;
}
A:visited { text-decoration: none; }
A:active { text-decoration: none; }
A:hover {
 text-decoration: underline;
 color: red;
}
```

Властивості списків

За допомогою CSS можна створити марковані й нумеровані списки, а також використовувати в якості маркера певне зображення.

У таблиці перераховані властивості елементів, призначені для налаштування вигляду списків:

Властивість	Значення	Опис	Приклад
list-style-type	disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha none	Вид маркеру. Перші три використовуються для створення маркованого списку, а інші — для нумерованого.	LI {list-style-type: circle} LI {list-style-type: upper-alpha}
list-style-image	none URL	Встановлює символом маркеру любе зображення	LI {list-style-image: url(check.gif)}
list-style-position	outside inside	Вибір положення маркеру відносно блоку тексту	LI {list-style-position: inside}

Скорочені властивості CSS

Деякі властивості CSS прийнято називати скороченнями або скороченими властивостями (shorthand property). Вони надають короткий, компактний спосіб запису для декількох інших, вузькоспеціалізованих властивостей. Наприклад, скорочена властивість border управляє зовнішнім виглядом рамки навколо елементу, дозволяючи визначити в одному рядку відразу кілька атрибутів: колір, стиль і ширину рамки. Інакше для цього знадобилися б властивості border-width, border-style і border-color.

Список скорочених властивостей:

font - скорочена властивість для властивостей шрифта:

* font-style
* font-variant
* font-weight
* font-size
* line-height
* font-family

background - скорочений варіант запису для властивостей фону:

    * background-color
    * background-image
    * background-repeat
    * background-attachment
    * background-position

`margin` - скорочена властивість для управління зовнішніми відступами:

    * margin-top
    * margin-right
    * margin-bottom
    * margin-left

`padding` - скорочена властивість для управління внутрішніми відступами:

    * padding-top
    * padding-right
    * padding-bottom
    * padding-left

`border` - скорочений варіант запису для властивостей бордюру:

    * border-width
    * border-style
    * border-color

`list-style` – скорочена властивість для налаштування вигляду списків:

    * list-style-type
    * list-style-image
    * list-style-position

Використання селекторів класів

Але на цьому можливості гнучкого налаштування візуального оформлення сторінки за допомогою таблиць стилів не закінчуються. Існують ще можливості для різного представлення однотипних елементів за допомогою однієї таблиці стилів - про них ми зараз і поговоримо.

Припустимо, ми хочемо створити сторінку, на якій буде два види абзаців `<p>`, при чому обидва види будуть постійно чередуватися і часто повторюватися. Типовий приклад такої сторінки - інтерв'ю, в якому чергуються питання журналіста і відповіді. При створенні такої сторінки доцільно візуально відокремити питання і відповіді один від одного. Якщо б довелося це робити тими засобами CSS, які були розглянуті вище, потрібно було б створити дві різні таблиці стилів. На щастя, в цьому немає необхідності. Можна створити в одній таблиці стилів два різних класи абзаців за допомогою селектора класу.

Для цього в описанні класу стилю після вказання назви тегу ставиться крапка і вказується назва класу, а в самому `html`-коді для конкретного елементу потрібно задати значення атрибуту `class` рівне назві класу. Це буде виглядати так:

```
<html>
 <head>
 <style>
  p.ask {    /* тег абзацу і селектор класу */
    font-style: italic;    /* курсив */
    font-weight: bold;    /* напівжирний */
    font-family: Arial, sans-serif; /*шрифт без засічок */
    font-size: 10pt;   /* розмір шрифту*/
    color: gray;    /* сірий колір тексту */
    margin-left: 15px    /* відступ зліва */
  }
  p.answer {
    font-family: 'Times New Roman', serif;
    font-size: 12 pt; color: red;
    margin-left: 15px
  }
 </style>
 </head>
 <body>
   <p class="ask">Питання журналіста</p>
   <p class="answer">Відповідь на питання</p>
 </body>
</html>
```

В приведеному прикладі питання журналіста відображатимуться шрифтом Arial сірого кольору, напівжирним курсивом, розміром 10 пунктів, з відступом 15 пікселів від лівого краю сторінки. Відповіді на питання відображатимуться шрифтом Times New Roman Розміром 12 пунктів червоного кольору:

Питання журналіста

Відповідь на питання

Важливо не забувати прописувати атрибут `class` різним класам абзаців безпосередньо в коді `html`. Можна створювати різну кількість класів для любим елементів сторінки.

Використання селекторів ідентифікатори елементів

Розглянемо інший випадок. Припустимо, потрібно створити на сторінці будь-які унікальні елементи, до яких в подальшому буде необхідність звертатися з програм JavaScript. Можливо, ці елементи будуть повторюватися на інших сторінках, і ви хотіли б задати їм єдине оформлення за допомогою CSS. На цей випадок у каскадних таблицях стилів є можливість призначення стилю елементу з унікальним ідентифікатором (необхідно вказати атрибут id). Щоб описати стиль для елементу з певним ідентифікатором потрібно вказати цей ідентифікатор, добавивши перед ним знак # .Найбільш часто ідентифікатори використовуються для елементів форм. Приклад призначення ідентифікатора і правил CSS:

```
<html>
<head>
  <style>
    #green {
      color: green;
    }
    #red {
      color: red;
    }
   </style>
</head>
<body>
  <p id = "green"> Текст, відображений зеленим кольором</p>
  <p id = "red">Текст, відображений червоним кольором</p>
</body>
</html>
```

Текст, відображений зеленим кольором

Текст, відображений червоним кольором

Використання комбінованих селекторів

Селектори нащадків

«Нащадками» елемента HTML називаються будь-які вкладені в нього елементи: це його «діти» (тобто безпосередньо вкладені), діти його дітей, і так далі, углиб ієрархії тегів. Правильно використовуючи селектори, ми можемо варіативно застосувати CSS стилі до потрібних елементів, пославшись на їх

батьківський елемент. Для цього перед селектором шуканого елемента треба вставити селектори його «предків», розділивши їх пробілом.

```
#foter li a {font-weight:bold; }
```

У наведеному прикладі селектор нащадків вказує на всі гіперпосилання (елементи A) вкладені в списки (елементи LI), у свою чергу знаходяться всередині елемента з ідентифікатором footer. У всіх таких гіперпосилань ми міняємо товщину шрифту на bold (напівжирний шрифт).

Селектори дітей

«Дітьми» або «дочірніми елементами» елемента HTML називаються безпосередньо вкладені в нього елементи, він для них є «батьківським» елементом. Елементи, що знаходяться на 2-му і глибших рівнях вкладеності, «дітьми» не є - це діти дітей, тобто «нащадки».

CSS дозволяє нам створити селектор для вибору дочірніх елементів будь-якого елементу і змінити їх властивості, застосувавши CSS стилі. Для цього перед селектором шуканого елемента треба вставити селектори його «предків», розділивши їх знаком >.

```
i > b {font-weight:normal;
```

В наведеному прикладі ми знаходимо всі елементи b, вкладені безпосередньо в елементи i, та вимикаємо для них напівжирний шрифт. Всі інші елементи B в документі залишаться без змін.

Селектори суміжних елементів (Adjacent Sibling Selectors)

Іноді потрібно вибрати елемент, розташований в HTML-документі безпосередньо за певним елементом. Так, наприклад, щоб вибрати всі заголовки H1, наступні за параграфами P, і змінити їхній верхній відступ, ми напишемо наступне правило CSS:

```
p + h1 {margin-top: 20px; }
```

Буде обраний тільки перший заголовок H1, розташований безпосередньо після P. Якщо між елементами P і H1 зустрінеться хоча б один елемент то селектор не спрацює. Селектори суміжних елементів з'явилися в CSS v2 і підтримуються, на жаль, не у всіх браузерах. В Internet Explorer вони з'явилися тільки з 7-ї версії, в Opera - з 5-ї версії.

Контрольні питання

1. Що таке CSS?
2. Що таке властивість стилю? Як визначити цей параметр?
3. Які є способи підключення каскадних таблиць стилів? Наведіть приклади.
4. Назвіть властивості для тексту та шрифту. Наведіть приклади.
5. Назвіть властивості стилів, що використовуються для задання кольорів. Яким чином можна задавати колір? Наведіть приклади.
6. Принципи використання псевдокласів гіперпосилань.
7. Властивості стилів для списків.
8. Скорочені властивості стилів.
9. Типи та синтаксис селекторів. Приклади використання.

Завдання для виконання лабораторної роботи:

Завдання 1. Ознайомитися з теоретичним матеріалом про методи підключення та налаштування каскадних таблиць стилів.

Завдання 2. Вивчити методи підключення каскадних таблиць стилів

Завдання 3. Ознайомитись з синтаксисом CSS, з властивостями стилів, що використовуються для налаштування вигляду тексту, списків, та гіперпосилань. Для отримання додаткової інформації рекомендується використовувати інформаційні ресурси Інтернет.

Завдання 4. Виконати навчальні приклади, результати зберегти.

Завдання 5. Використати власну html-сторінку, яка була створена в ході виконання лабораторної роботи № 1. Підключити до сторінки каскадні таблиці стилів, задати стилі для таких елементів як абзац, заголовки, списки та інші. При цьому використати можливості, описані в лабораторній роботі – скорочені властивості стилів, різноманітні селектори.

Завдання 6. Відповісти на контрольні питання, відповіді включити в звіт по лабораторній роботі.

Лабораторна робота №3

Тема: Компоновка html-сторінок.

Мета: Ознайомитись з способами використання мови розмітки та каскадних таблиць стилів для створення макету сайту. Навчитись використовувати блочні та рядкові елементи. Набути навики налаштування налаштування властивостей позиціонування елементів на сторінці.

Теоретичні відомості

Блочні елементи

Блочним називається елемент, який відображається на веб-сторінці у вигляді прямокутника. Такий елемент займає всю доступну ширину, висота елемента визначається його вмістом, і він завжди починається з нового рядка. До блоковим елементів відносяться теги `<h1>`, `...`, `<h6>`, `<hr>`, `<ol>`, `<p>`, `<pre>`, `<table>`, `<ul>`, `<div>`, `<address>`, `<blockquote>`, `<fieldset>`, `<form>`, і деякі застарілі. Також блоковим стає елемент, якщо в стилі для нього властивість `display` задано як `block`, `list-item`, `table`.

Для блочних елементів характерні наступні особливості:

- Блоки розташовуються по вертикалі один під одним.
- На прилеглих сторонах елементів діє ефект «схлопування» відступів (якщо у верхнього елементу є відступ знизу 15 пікселів, а у нижнього – зверху 10 пікселів, то між двома елементами буде відступ більшого розміру – 15 пікселів)
- Заборонено вставляти блоковий елемент всередину рядкового. Наприклад, код `<a><h1>Заголовок</h1></a>` не пройде валідацію, правильно вкладати теги навпаки - `<h1><a>Тема</a></h1>`.
- По ширині блокові елементи займають весь допуступний простір.
- Якщо задана ширина вмісту блоку (властивість `width`), то загальна ширина блоку складається з суми значень ширини, полів, рамок, відступів зліва і справа.
- Висота блочного елемента обчислюється браузером автоматично, виходячи з вмісту блоку.
- Якщо задана висота контенту (властивість `height`), то висота блоку складається з суми значень висоти, полів, рамок, відступів зверху і

знизу. При перевищенні зазначеної висоти контент відображається поверх блоку (за його межами).

- На блокові елементи не діють властивості, призначені для рядкових елементів, на зразок `vertical-align` (вирівнювання по вертикалі).
- Текст, розміщений всередині блочного елементу за замовчуванням вирівнюється по лівому краю.

Не всі блочні теги допустимо вкладати один в інший, тому при створенні структури сайту активну роль виконує елемент `<div>` як універсальний блок для верстки. Тег `<div>` допустимо вкладати один в інший, інші блокові елементи також можна поміщати всередину `<div>`. У прикладі показаний фрагмент коду сайту, де активно використовуються ці теги для формування необхідної конструкції.

```
<div id="container">
  <div id="intro">
    <div id="pageHeader">
      <h1>css Zen Garden</h1>
      <h2>The Beauty of CSS Design</h2>
    </div>
    <div id="mainContent">
      <p class="p1">A demonstration of what can be
accomplished visually through CSS - based design. Select any
style sheet from the list to load it into this page.</p>
    </div>
  </div>
</div>
```

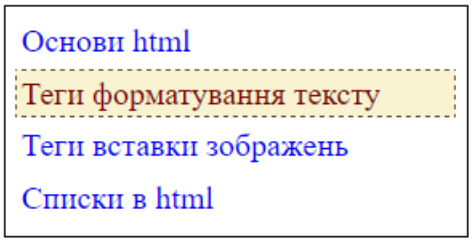
Перетворення в блочний елемент

У деяких випадках потрібно наділити рядковий елемент характеристиками блочного. Так, перетворення посилання в блок дозволяє збільшити корисну площу посилання за рахунок використання властивостей `width` і `height`. У прикладі показано створення меню, в якому посиланнями є вся доступна область.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <style type="text/css">
   .menu {
    width: 220px; /* Ширина меню */
    padding: 5px; /* Відсупи від рамки до пунктів меню */
```

```
    border: 1px solid #000; /* Рамка навколо меню */
   }
   .menu P { margin: 0 0 2px; }
   .menu A { /*стиль для посилання*/
    padding: 2px; /*Відступ від рамки навколо посилання до тексу
*/
    display: block; /* Посилання відобразити як блочний елемент */
    border: 1px solid #fff; /* Приховуємо рамку навколо */
    text-decoration: none; /* Забираємо підкреслення у посилань */
   }
   .menu A:hover { /*стиль для посилання, на яке наводиться
мишка*/
    background: #faf3d2; /* Коір фону під посиланням */
    color: #800000; /* Новий колір посилання */
    border: 1px dashed #634f36 /* Рамка навколо тексту */
   }
  </style>
 </head>
 <body>
  <div class="menu">
   <p><a href="1.html">Основи html</a></p>
   <p><a href="2.html">Теги форматування тексту</a></p>
   <p><a href="3.html">Теги вставки зображень</a></p>
   <p><a href="4.html">Списки в html</a></p>
  </div>
 </body>
</html>
```

Весь рядок виступає посиланням, а не лише сам текст, тому при наведенні на пункт меню курсора відбувається зміна стилю, і чітко видно яка область є посиланням (курсор наведено на 2-й пункт меню).



Рядкові елементи

Рядковими називаються такі елементи документа, які є безпосередньо частиною рядка. До рядкових елементів відносяться теги `<img>`, `<span>`,

<a>, <q>, <code> і ін., а також елементи, у яких властивість display встановлено як inline.

Перерахуємо характерні особливості рядкових елементів:

- Всередину рядкових елементів допустимо поміщати текст або інші рядкові елементи. Вставляти блокові елементи всередину рядкових заборонено.
- Ефект «схлопування» відступів не діє.
- Властивості, пов'язані з розмірами (width, height) не діють для рядкових елементів.
- Ширина дорівнює вмісту плюс значення відступів, полів і рамок.
- Кілька рядкових елементів, що слідують один за одним, розташовуються на одному рядку і переносяться на інший рядок лише при необхідності.
- Можна вирівнювати текст по вертикалі за допомогою властивості vertical-align.

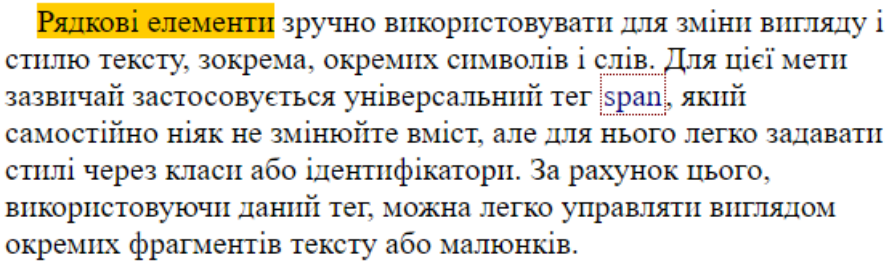
Рядкові елементи зручно використовувати для зміни вигляду і стилю тексту, зокрема, окремих символів і слів. Для цієї мети зазвичай застосовується універсальний тег , який самостійно ніяк не змінюйте вміст, але для нього легко задавати стилі через класи або ідентифікатори. За рахунок цього, використовуючи даний тег, можна легко управляти виглядом окремих фрагментів тексту або малюнків.

Для верстки рядкові елементи застосовуються рідше, ніж блочні. Це пов'язано в основному з тим, що всередину рядкових елементів не допускається вкладати контейнери , <p> і інші широко застосовувані теги. Проте, блокові і рядкові елементи вдало доповнюють один одного, оскільки дозволяють на всіх рівнях міняти вид складових веб-сторінок. У прикладі показано використання тега для виділення окремих слів.

```
<!DOCTYPE html >
<html>
 <head>
  <title>Рядкові елементи</title>
  <style type="text/css">
   .n_name {
    background: #fc0; /* Колір фону */
    margin-left: 1em; /* Відступ зліва */
   }
   .code {
```

```
        padding: 1px; /* поля навколо тексту */
        border: 1px dotted maroon; /* рамка */
        color: navy; /* колір тексту */
       }
      </style>
     </head>
     <body>
       <p><span class="n_name">Рядкові елементи</span> зручно використовувати для зміни вигляду і стилю тексту, зокрема, окремих символів і слів. Для цієї мети зазвичай застосовується універсальний тег <span class="code">span</span>, який самостійно ніяк не змінюйте вміст, але для нього легко задавати стилі через класи або ідентифікатори. За рахунок цього, використовуючи даний тег, можна легко управляти виглядом окремих фрагментів тексту або малюнків.
       </p>
     </body>
    </html>
```

Результат прикладу показано нижче:



В даному прикладі тег `<span>` і стилі використовуються для виділення різними способами фрагментів тексту. Зокрема, виділення відбувається за рахунок фонового кольору, рамки навколо тексту і зміною його кольору.

Перетворення в рядковий елемент

Рядкові елементи можна перетворювати в блокові за допомогою властивості `display` і його значення `block`.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Перетворення в рядкові елементи</title>
  <style type="text/css">
   .notetitle {
    border: 1px solid black; /* рамка */
    border-bottom: none; /* забираємо нижню межу рамки */
```

```
        padding: 3px; /* поля навколо тексту */
        display: inline; /* відобразити як рядковий елемент */
        background: #ffeebf; /* колір фону */
        font-weight: bold; /* виділити жирним  */
        font-size: 0.9em; /* розмір шрифту */
        margin: 0; /* забираємо відступи */
        white-space: nowrap; /* забороняємо перенос тексту */
       }
       .note {
        border: 1px solid #634f36; /* рамка */
        background: #f3f0e9; /* колір фону */
        padding: 7px; /* поля навколо тексту */
        margin: 0 0 1em 0; /* значення відступів */
       }
      </style>
     </head>
     <body>
      <p class="notetitle">Верстка сайтів</p>
      <p class="note">HTML спочатку замислювався як мова, якій
не потрібні кошти оформлення, такі як колір, розмір, рамки або
щось подібне. Розроблений в Європейському інституті фізики
частинок (CERN), HTML був іграшкою вчених, їх, перш за все,
цікавила логіка інформації, а не її візуальне уявлення. Тоді
ще не існувало поняття веб-дизайну і верстки як такої, всі
сайти за своїм оформленням були практично однотипними, в стилі
академічного дизайну. </p>
     </body>
    </html>
```

Верстка сайтів

HTML спочатку замислювався як мова, якій не потрібні кошти оформлення, такі як колір, розмір, рамки або щось подібне. Розроблений в Європейському інституті фізики частинок (CERN), HTML був іграшкою вчених, їх, перш за все, цікавила логіка інформації, а не її візуальне уявлення. Тоді ще не існувало поняття веб-дизайну і верстки як такої, всі сайти за своїм оформленням були практично однотипними, в стилі академічного дизайну.

В даному прикладі блочний тег `<p>` відображається на веб-сторінці як рядковий елемент. Це потрібно для того, щоб ширина фону і рамки дорівнювала ширині самого тексту з урахуванням полів. Як відомо, ширина блокових елементів не залежить від ширини вмісту, тому і доводиться

представляти тег <p> у вигляді рядкового елемента. В принципі, аналогічним рішенням буде використовувати замість <p> тег .

Рядково-блочні елементи

Блочні і рядкові елементи відмінно доповнюють один одного при верстці, займаючи кожен свою певну нішу. Але трапляються випадки, коли характеристик тих чи інших елементів недостатньо.

Галерея фотографій, представлена на рисунку нижче, складається з секцій, в які входить зображення з підписом до нього, при цьому секції розміщуються по горизонталі, займаючи всю доступну ширину. При зменшенні вікна браузера секції переходять на інший рядок.

Якщо для формування секцій використовувати тег , як блочний елемент він буде кожен раз починатися з нового рядка.

В HTML немає тега, який відноситься до рядково-блочним елементів, його можна визначити, задавши елементу властивість display із значенням inline-block.

```
div {
 display: inline-block;
}
```

Характеристики цих елементів наступні:

- Всередину рядково-блочних елементів допустимо поміщати текст, рядкові або блочні елементи.
- Висота елемента обчислюється браузером автоматично, виходячи з вмісту блоку.
- Ширина дорівнює вмісту плюс значення відступів, полів і рамок.

- Кілька елементів, які йдуть підряд, розташовуються на одному рядку і переносяться на інший рядок при необхідності.
- Можна вирівнювати по вертикалі за допомогою властивості `vertical-align`.
- Дозволено встановлювати ширину і висоту.
- Ефект «схлопування» відступів не діє.

Щоб створити галерею для тега `<div>` слід задати значення `display` як `inline-block`, а всередину нього додати зображення і підпис через тег `<p>` :

```
<!DOCTYPE html>
<html>
 <head>
  <title>Галерея</title>
  <style type="text/css">
   .photo {
    background: #d9dabb; /* Колір фону */
    width: 270px; /* Ширина */
    margin: 0 10px 10px 0; /* ВІдступи */
    padding: 10px 0; /* Поля зверху і знизу */
    text-align: center; /* Вирівнювання по центру */
    display: inline-block; /* Рядково-блочний елемент */
   }
   .photo img {
    border: 2px solid #8b8e4b; /* рамка */
   }
   .photo p {
    margin: 0; /* відступи */
   }
  </style>
 </head>
 <body>
  <div class="photo">
   <p><img src="images/001.jpg" alt="" /></p>
   <p class="caption">Кафедральний собор</p>
  </div>
  <div class="photo">
   <p><img src="images/002.jpg" alt="" /></p>
   <p class="caption">Замок</p>
  </div>
  <div class="photo">
   <p><img src="images/003.jpg" alt="" /></p>
   <p class="caption">Пішохідний міст</p>
  </div>
```

```
  <div class="photo">
   <p><img src="images/004.jpg" alt="" /></p>
   <p class="caption">Синагога</p>
  </div>
  <div class="photo">
   <p><img src="images/005.jpg" alt="" /></p>
   <p class="caption">Цвітіння сакур</p>
  </div>
  <div class="photo">
   <p><img src="images/006.jpg" alt="" /></p>
   <p class="caption">Набережна</p>
  </div>
 </body>
</html>
```

Контрольні питання

1. Що таке блочний елемент та які його властивості?
2. Що таке рядковий елемент та які його властивості?
3. Перетворення в рядковий елемент.
4. Перетворення в блочний елемент.
5. Поняття рядково-блочних елементів, їх властивості.

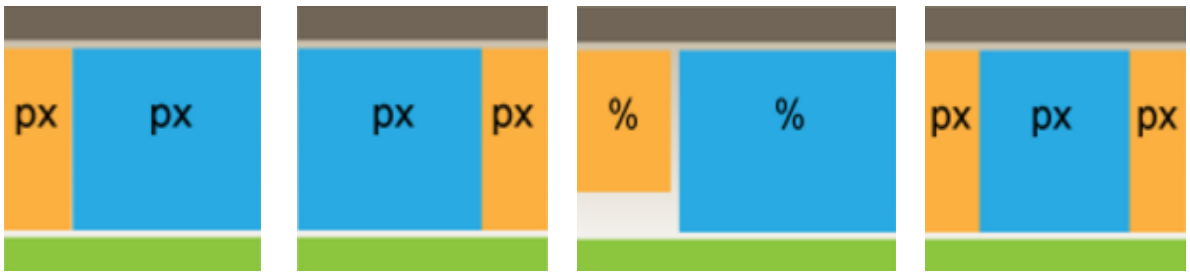
Завдання для виконання лабораторної роботи:

Завдання 1. Ознайомитися з теоретичним матеріалом про властивості блочних та рядкових елементів.

Завдання 2. Дослідити на практиці особливості використання рядкових та блочних елементів, зробити висновки про доцільність їх використання.

Завдання 3. Виконати та проаналізувати навчальні приклади, результати зберегти.

Завдання 4. Використати власну html-сторінку, яка була створена в ході виконання лабораторних робіт № 1 та 2. Використовуючи блочний тег , створити логічну структуру документу, виділивши «шапку», «контент», «бокову панель» та «підвал». Візуально це може виглядати наступним чином:



Завдання 5. Відповісти на контрольні питання, відповіді включити в звіт по лабораторній роботі.

Лабораторна робота №4

Тема: Створення користувацьких форм.

Мета: Ознайомитись з формами як методов взаємодії з користувачем. Навчитись використовувати різні елементи форми. Набути навики сьворення користувацьких форм на сторінці.

Теоретичні відомості

Поняття форми - FORM, ACTION, METHOD

Форма в `HTML`-документі реалізується тегом-контейнером `FORM`, в якому задаються всі керуючі елементи - поля введення, кнопки і.т.д. Якщо керуючі елементи вказані поза вмістом тегу `FORM`, то вони не створюють форму, а використовуються для побудови інтерфейсу користувача на веб-сторінці, тобто для привнесення в неї різних кнопок, прапорців, полів вводу.

Імена елементів форми присвоюються через атрибут `NAME`.

Зокрема `NAME` - визначає ім'я форми, унікальне для даного документа. Використовується тільки, якщо в документі присутні кілька форм.

`ACTION` - обов'язковий атрибут. Вказує шлях до скрипту (або програмі) сервера, який обслуговує дану форму.

`METHOD` - визначає спосіб відправки вмісту `html` форми. Можливі значення `GET` (за замовчуванням) і `POST`.

Метод `GET` використовується для передачі різних змінних, або дуже коротких повідомлень. Інформація передається в явному вигляді через рядок браузера, тобто її можна перехопити. Наприклад якщо ви бачите в рядку набору браузера щось на зразок `http://adress.com/lessons.php?rub=28` це означає, що передається значення змінної `rub` рівне `28`. У `html` формах зазвичай не використовується.

Метод `POST` створений спеціально для передачі текстових повідомлень. Майже завжди застосовується у формах. Передає інформацію в прихованому вигляді.

Однорядкові поля вводу, пароля, прихованого значення –

input: type=text, type=password, type=hidden

Елемент `INPUT` є найбільш вживаним тегом `HTML`-форм. Він дозволяє створювати всередині форми поля введення рядка тексту, імені файлу, пароля і.т.д.

Формат тегу `INPUT` для створення поля введення текстового рядка:

```
<input type=text name=ім'я_параметра [value=значення]
[size=розмір_поля] [maxlen=довжина_поля]>
```

Даний тег створює поле з максимально допустимою довжиною тексту `maxlen` і розміром в `size` знакомісць. Якщо вказаний атрибут `value`, то в полі буде спочатку відображатися значення даного атрибута. У квадратних дужках `[]` позначені необов'язкові атрибути.

Поля введення пароля

Ім'я користувача можна ввести за допомогою звичайного текстового поля. А ось пароль не повинен відображатися на екрані при його введенні. У цьому нам допоможе поле введення пароля:

```
<input type=password name=ім'я_параметра [value=значення]
[size=розмір] [maxlen=довжина]>
```

Принцип роботи даного тега такий же, як і текстового. Різниця полягає в тому, що інформація, що вводиться в поле не відображається, а замінюється "зірочками". Не рекомендується встановлювати значення за замовчуванням з міркувань безпеки (`value`).

Приховане текстове поле

Для передачі службової інформації (про яку користувач навіть не повинен підозрювати) використовуються приховані поля. За допомогою таких полів, наприклад, можуть передаватися параметри налаштувань.

```
<input type=hidden name=ім'я_параметра value=значення>
```

Незалежні та залежні перемикачі - input: type=checkbox, type=radio

Незалежні перемикачі

Дуже часто користувачеві, що заповнює форму в браузері, необхідно дати можливість вказати свої налаштування за допомогою вибору певних значень. При цьому наводяться самі ці значення, а поряд з ними міститься невелике квадратне поле, в якому можна встановити, чи прибрати галочку. При цьому значення, відповідно, буде або вибрано, або ні. Реалізувати це можна знову ж за допомогою тега `INPUT`. Для цього тільки необхідно як значення атрибуту `type` вказати `chechbox`.

```
<input type=checkbox name=імя_параметра value=значення [checked]>
```

Якщо перемикач був включений на момент натискання кнопки відправки даних, то скрипту буде переданий параметр `ім'я = значення`. Якщо ж прапорець вимкнений, то сценарієм взагалі нічого не буде передано - наче перемикача взагалі немає.

Перемикач за замовчуванням або включений, або вимкнений. Щоб перемикач був за замовчуванням включений, необхідно для нього вказати атрибут `checked`. В одній формі може бути одночасно обрано декілька перемикачів.

Залежні перемикачі

Залежні перемикач, так само як і незалежний перемикач, може бути або включений, або вимкнений. При цьому перемикач `radio` є залежним перемикачем, оскільки на формі може бути тільки один включений перемикач типу `radio`. Точніше, якщо в формі присутні кілька однойменних залежних перемикачів, то включений з них може бути тільки один. При виборі одного перемикача всі однойменні залежні перемикачі автоматично вимикаються. Як ім'я перемикачів сприймається значення атрибута `name`. Може бути тільки один активний перемикач. Приклад лістингу форми з залежними перемикачами:

```
<form action="http://localhost/script.php" method="GET">
      <input type=radio name=answer value=yes checked> Так
      <input type=radio name=answer value=no> Ні
      <input type=submit value=Відправити>
</form>
```

Кнопки – input: type=submit, type=reset, type=image

Ще одним елементом управління типу `INPUT` є кнопки. Кнопка відправки служить для відправки скрипту введених у форму параметрів. Синтаксис тега `INPUT` при цьому такий:

```
<input type=submit [name=go] value=Відправити>
```

Атрибут `value` визначає текст, який буде написаний на кнопці відправки. Атрибут `name` визначає ім'я кнопки і є необов'язковим. Якщо значення цього атрибута не вказувати, то скрипту будуть передані введені у форму значення і все. Якщо атрибут `name` для кнопки буде вказаний, то додатково до основних даних форми буде відправлена пара `ім'я = значення` від самої кнопки.

Кнопка скидання параметрів

Крім кнопки `submit` є ще кнопка `reset`, яка скидає параметри форми, а точніше, встановлює для всіх елементів форми значення за замовчуванням. Бажано, щоб на формі була така кнопка, особливо, якщо це велика форма. Наявність даної кнопки забезпечує очищення форми, наприклад, у випадку, коли були введені неправильні параметри. Синтаксис кнопки скидання:

```
<input type=reset value=Скидання>
```

Кнопка відправки з малюнком

Замість кнопки `submit` можна використовувати малюнок для відправки даних. Клік на цьому малюнку дає те ж саме, що й натискання на кнопку `submit`. Однак, крім цього, сценарієм будуть передані координати місця кліка на малюнку. Координати будуть передані у форматі `ім'я.x = коорд_X, y = коорд_Y`. Синтаксис кнопки відправки з малюнком:

```
<input type=image name=імя src=малюнок>
```

Багаторядкові текстові поля - TEXTAREA

Багатострічкові текстові поля створюються за допомогою тегу `TEXTAREA`. Поле, що створюється цим тегом, дозволяє вводити і відправляти не один рядок, а відразу кілька рядків. Синтаксис тега `TEXTAREA`:

```
<textarea name=імя [cols=ширина_в_символах]
rows=висота_в_символах] wrap=тип_переносу>
...текст ....
```

```
</textarea>
```

необов'язкові параметри `cols` і `rows` бажано все-таки вказувати. Перший з них задає кількість символів в рядку, а другий - кількість рядків в області.

Атрибут `wrap` визначає тип перенесення тексту, тобто як виглядатиме текст в полі вводу:

• `Virtual` - праворуч від текстового поля виводиться смуга прокручування. Введений текст виглядає розбитим на рядки, а символ нового рядка вставляється при натисканні клавіші `ENTER`;

• `Physical` - цей тип залежить від типу браузера і виглядає по-різному;

• `None` - текст виглядає в полі в тому вигляді, в якому користувач його вводить. Якщо текст не вміщається в один рядок, з'являється горизонтальна смуга прокручування.

Слід зауважити, що найбільш зручним є тип `Virtual`.

Списки **вибору - SELECT**

Списки з єдиним вибором

Досить часто існує необхідність представити які-небудь дані у вигляді списку і передбачити можливість вибору в цьому списку. В `HTML` списки реалізуються за допомогою тега `SELECT`. Список вибору дозволяє вибрати один варіант з множини. Приклад списку з єдиним вибором:

```
<select name=day size=1>
 <option value=1> понеділок </ option>
 <option value=2> вівторок </ option>
 <option value=3 selected> середа </ option>
 <option value=4> четвер </ option>
 <option value=5> п'ятниця </ option>
 <option value=6> субота </ option>
 <option value=7> неділя </ option>
</select>
```

Атрибут `name` визначає ім'я параметра, який буде переданий скрипту. Якщо атрибут `size` дорівнює 1, то список буде "оснащений" смугою прокручування. Значення, вибране у списку за замовчуванням, можна вказати за допомогою атрибуту `selected` для відповідного тега `option`. У наведеному прикладі день тижня за замовчуванням - середа. Атрибут `value` є

необов'язковим. Якщо його не вказати, то буде передана рядок, укладена в тег option.

Списки множинного вибору

За допомогою тега SELECT можна також створювати списки множинного вибору. У таких списках можна вибрати не одне, а відразу кілька варіантів значень (здійснювати вибір при натисненій клавіші CTRL). Для того, щоб створити список із множинним вибором, необхідно для тега SELECT вказати атрибут multiple. Ось практичний приклад такого списку:

```
<select name=day size=7 multiple>
<option value=1> понеділок </option>
<option value=1> вівторок </option>
<option value=1> середу </option>
<option value=1> четвер </option>
<option value=1> п'ятницю </option>
<option value=1> суботу </option>
<option value=1> неділю </option>
</select>
```

Передавання параметрів форми

Спочатку напишемо HTML-документ, який буде містити практично всі елементи HTML-форми Параметри форми будемо передавати скрипту для наступної обробки. Отже, лістинг HTML-документа send.html:

```
<html>
<head>
     <meta    http-equiv="Content-Type"    content="text/html;
charset=windows-1251">
     <title> Test  </ title>
</ Head>
<body>
     <h3> Тестова форма </ h3>
     <form name="form1" method="post" action="script.php">
          <p> <span> Текстове поле: </span>
               <input type="text" name="textfield">
          </p>
          <p> Поле введення пароля:
               <input type="password" name="pswfield">
          </p>
          <p> Приховане поле hidden
```

```
     <input name="hidden" type="hidden" id="hidden"
     value="Приховане_значення">
</p>
<hr size="1">
<p> Незалежні перемикачі (checkbox): </p>
<p>
     <input     type="checkbox"     name="checkbox1"
     value="1">Варіант перший
     <input     type="checkbox"     name="checkbox2"
     value="1">Варіант другий
     <input     type="checkbox"     name="checkbox3"
     value="1"     checked>Варіант     третій     (за
     замовчуванням)
</p>
<hr size="1">
<p> Залежні перемикачі (radio): </p>
<p>
     <input      name="radiobutton"      type="radio"
     value="yes">Так
     <input      name="radiobutton"      type="radio"
     value="no">Ні
</p>
<hr size="1">
<p> Багаторядкове текстове поле (textarea): </p>
<p>
     <textarea name="textarea" cols="40" rows="10">
     Текст за умовчанням </ textarea>
</p>
<hr size="1">
<p> Список з єдиним вибором: </p>
<p>
     <select name=day_s size=1>
          <option value=1> понеділок </option>
          <option value=2> вівторок </option>
          <option value=3 selected> середа </option>
          <option value=4> четвер </option>
          <option value=5> п'ятниця </option>
          <option value=6> субота </option>
          <option value=7> неділя </option>
     </select>
</p>
<p> Список із множинним вибором (multiple): </p>
<p>
     <select name=day_m[] size=7 multiple>
```

```
                    <option   value=1   selected>   понеділок
                    </option>
                    <option value=2> вівторок </option>
                    <option value=3> середа </option>
                    <option value=4> четвер </option>
                    <option value=5> п'ятниця </option>
                    <option value=6> субота </option>
                    <option value=7> неділя </option>
               </select>
          </p>
          <hr size="1">
          <p>
               <input type="submit" value="Відіслати форму">
               <input type="reset" value="Очистити форму">
          </p>
     </form>
</body>
</html>
```

Коли користувач натискає кнопку "Відіслати форму", браузер передасть скрипту наступні параметри:

• `textfield` - значення текстового поля;

• `pswfield` - значення поля введення пароля;

• `hidden` - значення прихованого поля;

• параметри `checkbox`: `checkbox1`, `checkbox2` і `checkbox3` будуть передані тільки в тому випадку, якщо відповідні їм незалежні перемикачі активні;

• `radiobutton` - значення групи `radio` (буде передано одне із значень: `Yes` або `No`);

• `textarea` - вміст багатострічкової текстової області;

• `day_s` - значення списку з єдиним вибором;

• `day_m` - значення списку із множинним вибором.

Контрольні питання

6. Що таке форма?
7. Основні обов’язкові атрибути форм?
8. Однорядкові поля форми.
9. Залежні та незалежні перемикачі.

10.Кнопки.

11.Списки множинного вибору.

12.Багаторядкове поле введення.

Завдання для виконання лабораторної роботи:

Завдання 1. Ознайомитися з теоретичним матеріалом про створення користувацьких форм.

Завдання 2. Виконати та проаналізувати навчальні приклади, результати зберегти.

Завдання 3. Створіть окрему html-сторінку, використати макет, створений під час виконання лабораторної роботи № 3. Всередині логічного блоку «контент» створіть форму, за допомогою якої користувач зможе залишити відгук про вашу сторінку. Форма повинна містити:

- поле для введення імені
- поле для введення електронної адреси
- поле для відгуку
- можливість виставити оцінку за 5-бальною шкалою
- інше

Завдання 4. Відповісти на контрольні питання, відповіді включити в звіт по лабораторній роботі.

Лабораторна робота №5

Тема: Основи синтаксису PHP.

Мета: Ознайомитись з основами синтаксису PHP. Навчитись використовувати блоки PHP всередині HTML-сторінок. Здобути навики використання керуючих конструкцій мови PHP (розгалуження, цикли).

Теоретичні відомості

Включення в HTML, основи синтаксису, коментарі

За замовчуванням в кінці імен PHP документів ставиться розширення ".php". Коли веб-сервер зустрічає в запиті це розширення, він автоматично передає файл PHP процесору. Можливе і налаштування веб-сервера таким чином, щоб імена файлів закінчувалися розширенням ".htm", ".html".

На виході документа PHP буде отримуватися тільки код html. Команди на мові PHP обрамляються спеціальними дескрипторами - тегами мови PHP. Все, що знаходиться поза цими тегами, ігнорується інтерпретатором. Підтримуються наступні стилі написання тегів:

XML-стиль `<?php код на PHP ?>`

HTML-стиль `<script language="php"> код на PHP </script>`

Короткий стиль `<? код на PHP ?>`

ASP-стиль `<% код на PHP %>`

Кожна команда закінчується крапкою з комою (;). Одну команду можна записувати у кілька рядків або кілька команд в один рядок. PHP чутливий до регістру символів в іменах змінних і функцій.

```
$index = 10;
print($Index);      // Помилка

$index = 10;
print($index);      // Немає помилки
```

PHP нечутливий щодо пробілів, переносів рядка, знаків табуляції.

```
$index = 10;
$index = 10 + 20;
$index = 10+10;
$index =
```

```
10
+
10;
```

PHP підтримує три види коментарів: один багаторядковий і два однорядкових.

```
 /*
     Перший вид коментарів
 */
//   Другий
#    Третій
```

Оголошення змінних, системні змінні

Всі імена змінних повинні починатися зі знака долара ($); Тоді процесор зразу ідентифікує змінну, виділяючи її серед звичайного тексту, що використовується в HTML сторінках. Оголошення не є обов'язковими. Змінна починає існувати з моменту присвоєння їй значення або з моменту першого використання. Якщо використання починається раніше присвоєння, то змінна буде містити значення за замовчуванням. Змінній не призначається певний тип. Тип визначається значенням, що зберігаэться в ній та поточними операціями над нею.

Першим символом після $ повинна бути буква або символ підкреслення. Далі в імені змінної можуть бути присутніми букви, цифри та символ підкреслення.

```
$I;
$1;
$_1  ;
$firstName;
$7Lucky;
$~password;
$Last!Visit;
$Compute-Mean;
```

В PHP існує декілька способів вивести що-небудь в веб сторінку:

Оператори `echo` і `print` виводять значення аргумента, але `print` завжди повертає 1, а `echo` нічого не повертає.

Оператор `print_r()` дозволяє коректно виводити масиви.

Оператор `var_dump()` виводить змінну разом з типом, що зручно при відлагодженні.

Ідентифікатори чисел, рядків чи масивів відносяться до змінних і записуються однаково:

```
$mycounter = 1;
$mystring   = "Hello";
$myarray     = array("One",  "Two",  "Three");
```

```
$foo = 'Bob';  // Присвоєння $foo значеннння 'Bob'
$foo = "My name is Mike";     // Зміна $foo
$bar = 25;          // Присвоєння $bar значення 25
$bar = 2 + 2;       // Присвоєння $bar 4
$tmp = $foo;        // Присвоєння $tmp значення $foo
$tmp = &$foo;       // Посилання на $foo через $tmp
$foo = "John";      // Зміна $foo
echo $tmp;          // Вивід на екран "John"
$foo = "Mike";      // Зміна $foo
unset($foo);        // Усуваємо $foo
echo $tmp;          // Вивід на екран "Mike"
```

Системні змінні

`$GLOBALS` - Масив, що містить всі глобальні змінні.

`$_ENV` - Масив змінних оточення.

`$_COOKIE` - Масив файлів cookie, відправлених на сервер.

`$_GET` - Масив змінних, відправлених методом GET.

`$_POST` - Масив змінних, відправлених методом POST.

`$_FILES` - Масив, що містить інформацію про завантажені файли.

`$_REQUEST` - Масив, що містить $ _GET, $ _POST, $ _FILES, $ _COOKIE.

`$_SESSION` - Масив змінних, розміщених у сесіях PHP.

`$_SERVER` - Масив, що містить інформацію про сервер.

Типи змінних

PHP підтримує наступні основні типи даних:

1. Чотири скалярних типи:

boolean - логічний;
integer - ціле число;
float (double) - число з плаваючою крапкою;
string – стрічка;

2. Два змішаних типи:

   array - масив;
   object - екземпляр класу.

3. Два спеціальних типи:

   resource - посилання на зовнішнє по відношенню до скрипта джерело даних (файл на диску);
   NULL - відсутність якого-небудь значення.

Основні функції над типами змінних

isset (ім'я_змінної) - повідомляє, чи існує змінна.

unset (ім'я_змінної) - знищує зазначену змінну

empty (ім'я_змінної) - повідомляє, чи присвоєно змінній якесь значення.

gettype (ім'я_змінної) - повертає тип вказаної змінної

settype (ім'я_змінної, тип) - конвертує змінну в інший тип.

is_bool (ім'я_змінної) - перевіряє чи є тип змінної логічним.

Функції is_numeric (), is_float (), is_int (), is_string (), is_object (), is_array () працюють за аналогією.

Константи, системні константи

Для значень, які не будуть змінюватися в ході виконання сценарію використовують константи. Як і змінні, константи можуть бути визначені і доступні в будь-якому місці сценарію, але у них є і ряд особливостей:

У констант немає префіксу у вигляді знака долара. Константам можна присвоювати значення, їх можна визначати викликом функції define (). Константи не можуть бути визначені або анульовані після первинного оголошення.

```
define ('PI', 3.14);
$index = 10 * PI;
```

```
PI = 10 * 3.14;               // Помилка!
define("CONSTANT", "Привіт.");
echo CONSTANT;           // Виведе "Привіт"
echo Constant; // Виведе "Constant" та попередження
```

Системні константи

```
__ LINE_ _         //Номер поточного рядка.
__ FILE_ _         //Повний шлях та ім'я поточного файлу.
__ FUNCTION_ _     //Ім'я поточної функції.
__ CLASS_ _        //Ім'я поточного класу.
PHP_EXTENSION_DIR   // Каталог розширень PHP
PHP_OS             // Операційна система
PHP_VERSION        // Версія PHP
PHP_CONFIG_FILE_PATH // Каталог розміщення php.ini
```

Масиви, асоціативні масиви

Масив - це структура, в якій зберігається упорядкований набір даних. Ці дані називаються елементами масиву. Кожен елемент масиву має свій унікальний індекс.

В PHP масив можна створити такими способами:

```
$zoo[0] = 'слон';
$zoo[6] = 'крокодил';
$zoo[4] = 'жираф';
$zoo[] = 'осел';         // Індекс дорівнює 7
// або
$zoo = array ('лев', 'медвідь', 'мавпа');
echo count ($zoo);  // Кількість елементів масиву
```

Асоціативні масиви. В асоціативних масивах використовується стрічковий індекс, а не числовий:

```
$pets['dog'] = 'Бульдог';
$pets['cat'] = 'Сибірський';
$pets['fish'] = 'Окунь';
// або
$pets = array ('lizard' => 'Ігуана',
            'spider' => 'Тарантул',
            'parrot' => 'Какаду');
print_r ($pets);         // Друк масиву
```

Багатовимірні масиви

Масив називається багатовимірним тоді, коли в якості його елементів виступають не тільки скалярні величини, але і самі масиви.

```
$users = array (0 => array ('login' => 'admin',
                    'password' => 'hskdfuegefdjfdg'),
                1 => array ('login' => 'telo',
                    'password' => 'ppqmcnvkfghye')
                    );
echo $users[0]['login']; // admin
```

Умовні оператори if, else, elseif, switch

Конструкція if

Вказані дії виконуються тільки тоді, коли умова є істинною

```
if (умова) {
      Дія;
}
```

```
if($index > 0){
     echo 'Index > 0';
}
```

Конструкція if...else

Якщо умова істинна, виконується дія із блоку if, в протилежному випадку — з блоку else.

```
if(умова){
   Дія;
}else{
   Дія;
}
```

```
if($index > 0){
   echo 'Так';
        } else {
   echo 'Ні';
 }
```

Конструкція elseif

Якщо умова блоку if істина, виконаються дії блоку if. В іншому випадку, якщо умова блоку elseif істина, виконаються дії блоку elseif. У всіх інших випадках виконається дії з блоку else.

```
if(умова){
  Дія;
}else{
  Дія;
}
```

```
if($numb= 5 && $numb = 10)
          $discount = 5;
      }else{
          $discount =10;
       }
```

Конструкція switch

Якщо значення змінної відповідає значенню одного з блоків case, виконаються дії з цього блоку. В іншому випадку - з блоку default.

```
switch(Змінна){
 case Значення 1:
      Дія 1;
      [break;]
 case Значеннння 2:
      Дія 2;
      [break;]
 [default: Дія;]
}
```

```
switch($day){
 case 1:
    echo 'Понеділок';break;
 case 2:
    echo 'Вівторок'; break;
 case 3:
    echo 'Середа'; break;
 case 4:
    echo 'Четвер'; break;
 case 5:
    echo  'П’ятниця'; break;
 case 6:
    echo 'Субота'; break;
 case 7:
    echo ‘Неділя'; break;
 default:
    echo 'Немає такого дня';
}
```

Оператори циклу for, while

Цикли призначені для багаторазового виконання набору інструкцій. У циклі for вказується початкове і кінцеве значення лічильника, а також крок, з яким лічильник буде змінюватися. Змінюватися лічильник може як в позитивну, так і негативну сторону. Дії виконаються стільки разів, скільки ітерацій пройде від початкового значення лічильника до досягнення кінцевого, з вказаним кроком.

```
for(початок;кінець;крок){
   Дія;
```

```
for($i = 1; $i = 5; $i++) {
   $sum += $i;
```

```
   ...
}
```

```
   eho $sum;
}
```

Цикл while. Дії будуть виконуватися до тих пір, поки умова істинна. Цикл while є циклом з передумовою.

```
while (умова) {
  Дія;
  ...
}
```

```
while ($state == 'Ранок') {
   echo 'Робочий день починається';
   $state = 'Обід';
}
```

Цикл do ... while Цикл do ... while є циклом з післяумовою. Це означає, що спочатку буде виконуватися дія, а потім перевірятися умова. Таким чином дія завжди виконається мінімум один раз.

```
do{
   Дія;
   ...
} while (умова);
```

```
do{
   echo 'Удар';
     } while ($state == 'Бокс');
```

Оператори управління циклами break, continue

Break перериває роботу циклу. Інтерпретатор перейде до виконання інструкцій, наступних за циклом. Continue перериває виконання поточної ітерації циклу. Цикл продовжить виконуватися з наступної ітерації

```
$index = 1;
while ($index < 10){
   echo "$index ";
   $index++;
   if ($index == 5)
     break;
}

$index = 0;
while ($index < 10){
   $index++;
   if ($index == 5)
     continue;
   echo "$index";
}
```

Оператор циклу foreach

Цикл foreach Дуже зручний при роботі з масивами. Зазначені дії виконуються для кожного елемента масиву $array, при цьому $key - номер елемента масиву $array, $ value - значення цього елемента.

```
foreach ($array as [ $key => ] $value){
  Дія;
  ...
}
```

```
$pets[] = 'Собака';
$pets[] = 'Кіт';
$pets[] = 'Риба';
foreach ($pets as $index => $value) {
   echo "Елемент № $index має значення: \"$value\"<br>";
}
```

Управління масивами

Сортування - sort

Сортувати можна як прості, так і асоціативні масиви. Для сортування масивів в PHP існують певні функції:

• sort() - сортує масив в алфавітному порядку, якщо хоч би один з його елементів є рядком, і в числовому порядку, якщо усі його елементи - числа.

• rsort() - працює як sort( ), але в зворотному порядку.

• asort() - сортує асоціативний масив; працює як sort( ), але зберігає імена елементів.

• arsort()- працює як asort( ), але в зворотному порядку.

• ksort() - сортує асоціативний масив по іменах елементів.

• krsort()- працює як ksort( ), але в зворотному порядку.

```
<html>
<head>
<title>Сортування асоціативного масиву</title>
</head>
<body>
```

```
<?php
$a = array("перший" => 6, "другий" => 2, "третій" => 1);
echo  "\$a:<br>"
asort ($a);
echo  "asort (\$a) :<br>"
foreach ($a as $k => $t) echo " $k = $t<br>";
ksort ($a);
echo  "ksort (\$a) :<br>"
foreach ($a as $k => $t) echo "$k = $t<br>";
?>
</body>
</html>
```

РЕЗУЛЬТАТ ПРИКЛАДУ :

$a: перший = 6 другий = 2 третій = 1

asort ($a) : третій = 1 другий = 2 перший = 6

ksort ($a) : другий = 2 перший = 6 третій = 1

Пошук елементу

Для перевірки наявності елементу в масиві існують функції:

• in_array() - якщо елемент знайдений, повертає true, інакше - false.

• array_search() - якщо елемент знайдений, повертає його ключ, інакше - false.

```
<html>
<head>
<title>Пошук елементу в масиві</title>
</head>
<body>
<?php
$a = array("перший" => 6, "другий" => 2, "третій" => 1);
if (in_array (2, $a)) echo "значення 2 знайшли!<br>";
echo "ключ знайденого елементу - ".array_search(2, $a);
?>
</body>
</html>
```

РЕЗУЛЬТАТ ПРИКЛАДУ:

значення 2 знайшли!

ключ знайденого елементу – другий

ПРИКЛАД 1

```
<html>
<head>
<title>Перегляд асоціативного масиву</title>
</head>
<body>
<?php
$ціна = array ("помідори" => 15, "огірки" => 12);
foreach ($ціна as $овочі => $грн)
  {
  echo  "$овочі коштують $грн грн.<br>";
  }
?>
</body>
</html>
```

РЕЗУЛЬТАТ ПРИКЛАДУ 1:

помідори коштують 15 грн. огірки коштують 12 грн.

ПРИКЛАД 2

```
<html>
<head>
<title>Перегляд двовимірного масиву</title>
</head>
<body>
<?php
$дані = array (
         "Іванов" => array ("ріст" => 174, "вага"  => 68)
         "Петров" => array ("ріст" => 181, "вага"  => 90),
         "Сидоров" => array ("ріст" => 166, "вага"  => 73));
foreach ($дані as $прізвище => $дані1)
  {
  echo  "<br>$прізвище:<br>";
  foreach ($дані1 as $параметр => $pp)
    {
    echo  "$параметр = $pp<br>";
    }
  }
?>
</body>
</html>
```

РЕЗУЛЬТАТ ПРИКЛАДУ 2:

Іванов: ріст = 174 вага = 68

Петров: ріст = 181 вага = 90

Сидоров: ріст = 166 вага = 73

ПРИКЛАД 3

```
<?php
$a = array ('a' => 'apple ', 'b' => 'banana ',
            'c' => array ('x ',' y ',' z'));
print_r ($a);
?>
```

РЕЗУЛЬТАТ ПРИКЛАДУ 3 :

Array ([a] => apple [b] => banana [c] => Array ([0] => x [1] => y [2] => z))

Контрольні питання:

1. Наведіть приклади включення PHP в HTML-сторінку.
2. Які правила використовуються в PHP для оголошення змінних та констант?
3. Які типи даних можна виділити в PHP?
4. Які функції можна використовувати для роботи з змінними та їх типами?
5. Опишіть масиви та асоціативні масиви на PHP.
6. Які управляючі конструкції є в мові PHP?
7. Які конструкції використовуються для реалізації розгалужених алгоритмів?
8. Які конструкції мови PHP використовуються для реалізації циклічних алгоритмів?
9. Які функції для роботи з масивами передбачені в PHP?

Завдання для виконання лабораторної роботи:

Завдання 1. Ознайомитися з теоретичним матеріалом про створення користувацьких форм.

Завдання 2. Виконати та проаналізувати навчальні приклади, результати зберегти.

Завдання 3. В боковій панелі макету сторінки реалізуйте виведення поточної дати.

Завдання 4. Реалізуйте обробку даних форми, яку ви створили в лабораторній роботі №5. Виведіть інформацію, яку внів користувач в форму, на веб-сторінку.

Список використаної літератури

1. Антоненко В.М., Терейковський І.А., Терейковська Л.О. «Сучасні Internet техно-логії. Курс лекцій та лабораторний практикум. Частина І. Основи Web – дизайну». – Ірпінь: Академія ДПС України, 2005. – 231 с.
2. Пасічнік О.Г., Пасічнік О.В., Стеценко І.В. Основи веб-дизайну – Вид. група BHV , 2009 – 336 с.
3. Дунаев В.. HTML, скрипты и стили. СПб: БХВ-Петербург, 2011- 816с.
4. Браун М.Р., Хоникарт Д. Использование HTML 4.0 – С-Пб: Вильямс, 1999.
5. Глинський Я.М, Ряжська В.А. Інтернет: Сервіси, HTML; Web-дизайн: Навч.посіб. – Л: Деол, 2003.
6. Паттерсон Л. и др. Использование HTML 4.0. - К.-М.-СПб.: Издат. Дом “Вильямс”. – 1998 – 384с.
7. Спейнаур С., Куэрсиа В. Справочник Web-мастера – К.: BHV, 1997
8. Веллинг Л. Разработка Web-приложений с помощью PHP и MySQL / Л. Веллинг, Л. Томсон, пер. с англ. – 2-е изд. – М. : Издательский дом «Вильямс», 2004. – 800 с.
9. Дюбуа П. MySQL : учебное пособие / П. Дюбуа ; пер. с англ. — М. : Издательский дом «Вильямс», 2001. – 816 с.
10. Кухарчик А. PHP: обучение на примерах / А. Кухарчик – Мн. : Новое знание, 2004. – 237 с.

Гарнітура Times New Roman. Папір офсетний.
Формат видання 60х84/16.
Умов. друк. арк.4,0 Зам. № 39. Наклад 25 прим.

Видруковано ПП «АУДИТОР-ШАРК»
88000, м. Ужгород, пл. Жупанатська, 15/1. Тел.: 3-51-25.
E-mail: office@shark.com.ua

Свідоцтво про внесення суб'єкта видавничої справи
до державного реєстру видавців,
виготівників і розповсюджувачів
видавничої продукції
Серія 3т № 40 від 29 жовтня 2012 року