Автоматизированные системы разработки web-сайтов

Основы HTML

Цель работы

HTML (HyperText Markup Language) — язык разметки гипертекста, используемый для создания документов, независимых от аппаратно-программной платформы. HTML — это не язык программирования, а описательный язык разметки. В ходе выполнения этой лабораторной работы Вам необходимо освоить базовые приемы использования языка HTML для гипертекстовой разметки веб-страниц.

Задание

1. Спроектировать структуру веб-сайта по теме ***вашей*** *учебной научно-исследовательской работы* (УНИРС) или по любой другой теме, сопоставимой (или бОльшей) по объему с УНИРС.
2. Разработать эскиз оформления веб-сайта (использовать любой графический редактор).
3. Выполнить верстку *макета страницы* по разработанному эскизу.

Указания к работе

- Теги HTML не чувствительны к регистру.
- Различные версии HTML поддерживают устаревшие (*deprecated*) теги только для обратной совместимости.
- Значения атрибутов крайне рекомендуется закрывать в одинарные или двойные кавычки.

Список основных тегов HTML

Тег	Атрибуты	Описание
<html>		Контейнер HTML-документа. Закрывающий тег обязателен </html>
<head>		Начальный и конечный теги заголовка документа. Закрывающий тег обязателен </head>
<title>		Тег названия HTML-документа. Закрывающий тег обязателен </title>
<meta>	Детальное описание	Предоставляет дополнительную информацию о документе.
<body>	link = цвет не посещенных ссылок alink = цвет активных ссылок vlink = цвет посещенных ссылок text = цвета обычного текста bgcolor = цвет фона документа backgroud = url фонового изображения bgproperties = запрещает прокрутку фонового изображения leftmargin = устанавливает размер левого поля документа topmargin = устанавливает размер верхнего поля документа	Начальный и конечный тег тела документа. Закрывающий тег обязателен </body>
	align = left - по левому краю align = center - по центру align = right - по правому краю align = justify - по обоим краям	Параграф, основной текстовый контейнер, закрывающий тег обязателен . После закрывающего тега автоматически происходит перенос строки.
	align = left - по левому краю align = center - по центру align = right - по правому краю align = justify - по обоим краям	Контейнер, основное предназначение - размещение элементов на странице, закрывающий тег обязателен .
 		Принудительный перенос строки, закрывающий тег не требуется

Тег	Атрибуты	Описание
<pre>		Заключенный в теги <pre></pre> текст будет отображаться так, как он был отформатирован предварительно, без обработки, с точным соблюдением переносов строк и интервалов.
	 • <li type=disk> - диск ○ <li type=circle> - окружность ▪ <li type=square> - квадрат 	Неупорядоченный список, элементы списка выводятся тегом
	 1. <li type=disk> - арабские цыфры b. <li type=circle> - буквы нижнего регистра 3. <li type=square> - буквы верхнего регистра iv. <li type=square> - римские цифры в нижнем регистре V. <li type=square> - римские цифры в верхнем регистре 	Упорядоченный список, элементы списка выводятся тегом
<a>	href = url target =_blank - открывает ссылку в новом окне title = подсказка к текстовой ссылке	Создает в документе гиперссылку, обязательный атрибут href, закрывающий тег обязателен
		Заключенный в теги текст будет отображаться в курсивном начертании, закрывающий тег обязателен.
<b>		Заключенный в теги <b><b> текст будет отображаться жирным шрифтом, закрывающий тег обязателен.
<tt>		Заключенный в теги <tt></tt> текст будет отображатьсямоно ширинный шрифтом, закрывающий тег обязателен
<h>		Заключенный в теги текст представляет собой заголовок. Возможные значения - от 1 до 6, закрывающий тег обязателен.
<sub>		Заключенный в теги текст будет смещен вниз (нижний индекс), закрывающий тег обязателен
<sup>		Заключенный в теги текст будет смещен вверх (верхний индекс), закрывающий тег обязателен
<big>		Заключенный в теги <big></big> текст будет отображаться шрифтом большего размера, закрывающий тег обязателен
<small>		Заключенный в теги текст будет отображаться шрифтом меньшего размера, закрывающий тег обязателен
<img>	src=URI - адрес изображения alt = подсказка к изображению border = устанавливает толщину рамки вокруг изображения height = задает высоту изображения в пикселях width = задает ширину изображения в пикселях hspace = задает свободную область слева и справа от изображения Vspace = задает свободную область над и под изображением	В текущий документ вставляет изображение, закрывающий не требуется, обязательный атрибут src = url
<table>	align = left - выравнивание таблицы по левому краю align = right - выравнивание таблицы по правому краю align = center - выравнивание таблицы по центру Backgroud = url указывается фоновое изображение для таблицы Bgcolor = задается цвет фона всей таблицы Border = создает рамку таблицы Bordercolor = задает цвет рамки таблицы Cellpadding = задает область свободного пространства внутри ячейки Cellspacing = задается интервал между смежными ячейками таблицы Hspace = задает областей свободного пространства слева и справа от таблицы Vspace = задает областей свободного пространства над и под таблицей Width= установка ширины таблицы в пикселях или процентах от ширины окна	начальный <table></table> и конечный теги таблицы, обязательные вложенные теги <tr> и <td>
<tbody>		Начальный тег для строк таблицы, образующих основное тело таблицы.
<tr>	align = left - по левому краю align = center - по центру align = right - по правому краю Background = указывает url фонового изображения для строки Bgcolor = задает цвет фона данной строки Bordercolor = задает цвет обрамления строки Атрибут Valign задается размещение содержимого ячейки или ячеек в данной строке по вертикали Valign = top - сверху Valign = center - по центру Valign = bottom - внизу	Обязательный вложенный тег для тегов <table></table> создает строки ячеек в текущей таблице, закрывающий тег обязателен<tr></tr>

Тег	Атрибуты	Описание
<td>	При помощи атрибута align задается способ выравнивания содержимого для данной ячейки: align = left - по левому краю align = center - по центру align = right - по правому краю Background = указывает url фонового изображения для данной ячейки Bgcolor = задает цвет фона данной ячейки Bordercolor = задает цвет обрамления ячейки Атрибут Valign задается размещение содержимого ячейки Valign = top - сверху Valign = center - по центру Valign = bottom - внизу	Обязательный вложенный тег для тегов <table></table> создает ячейку в текущей строке <tr></tr>, пример написания: <table> <tr><**td>...</td**></tr></table> закрывающий тег обязателен
<th>	При помощи атрибута align задается способ выравнивания содержимого для данной ячейки: align = left - по левому краю align = center - по центру align = right - по правому краю Background = указывает url фонового изображения для данной ячейки Bgcolor = задает цвет фона данной ячейки Bordercolor = задает цвет обрамления ячейки Атрибут Valign задается размещение содержимого ячейки Valign = top - сверху Valign = center - по центру Valign = bottom - внизу	Обязательный вложенный тег для тегов <table></table> создает ячейку в текущей строке <tr></tr>, по своим параметрам схож с тегом <td>, но текст который будет помещен внутрь такой ячейки будет отображаться полужирным начертанием

Тег META. Мета-информация о веб-странице

Основное предназначение мета-тегов, это включение информации о документе, которая может содержать сведения об авторе, дате создания документа или авторских правах.

Вся информация находящаяся в META теге, полезна для серверов, браузеров и поисковых роботов. Для посетителя веб-страницы информация, которую несут в себе META теги будет не видна, следовательно META теги нужно помещать внутри тега `<head>` документа.

В документе может находится любое количество тегов `<META>`

Рассмотрим некоторые, часто используемые мета-теги:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=Windows-1251">
```

Используется для того, чтобы браузер мог правильно отобразить содержимое страницы и для определения поисковой машиной языка, на котором написана страница.

```
<META HTTP-EQUIV="Refresh" CONTENT="n; URL=http://newadres.ru/">
```

Автоматическое перенаправление (через n секунд) на указанный адрес (редирект).

```
<META content="Имя автора страницы" name="author">
```

Используется для указания имени автора. Поисковые системы могут найти нужный сайт по имени автора.

```
<META content="список, ключевых, слов" name="keywords">
```

Список ключевых слов, вписываются слова или выражения через запятую с учетом регистра, так как ряд поисковых систем не распознают регистр при индексации. Данный META тег является одним из главных META тегов при индексировании вашего сайта поисковой машиной. Длина содержимого тега META "keywords" не должна превышать 1000 символов.

```
<META content="Сюда вписывается краткое описание страницы" name="description">
```

Этот тег задает фразу, по которой пользователь определяет суть вашей страницы и решает, посещать ли ее. Вписанные выражения в данный META тег играют важную роль в рейтинге страницы. Ключевые выражения должны совпадать с основным текстом страницы, это тоже играет большую роль при индексации страницы поисковой машиной. Длина содержимого тегов META "description" не должна превышать 200 символов.

```
<META content="index,all" name="robots">
```

Управление поисковым роботом, указание того, что страницу нужно индексировать (или нет, если указано "noindex").

Специальные символы

В таблице приведены *некоторые* специальные символы HTML, которые имеют особое назначение и собственный способ представления в виде мнемонического или числового кода.

Символ	Мнемокод	Числовой код	Описание
			неразрывный пробел
¢	&cent;	&#162;	цент
£	&pound;	&#163;	фунт стерлингов
¥	&yen;	&#165;	иена или юань

Символ	Мнемокод	Числовой код	Описание
§	&sect;	&#167;	параграф
©	&copy;	&#169;	знак copyright
«	&laquo;	&#171;	левая двойная угловая скобка
	&shy;	&#173;	место возможного переноса
®	&reg;	&#174;	знак зарегистрированной торговой марки
°	&deg;	&#176;	градус
²	&sup2;	&#178;	верхний индекс два квадрат
³	&sup3;	&#179;	верхний индекс три куб
·	&middot;	&#183;	точка по середине
»	&raquo;	&#187;	правая двойная угловая скобка
½	&frac12;	&#189;	дробь – одна вторая
×	&times;	&#215;	знак умножения
÷	&divide;	&#247;	знак деления
σ	&Sigma;	&#931;	греческая заглавная буква сигма
λ	&lambda;	&#955;	греческая строчная буква лямбда
μ	&mu;	&#956;	греческая строчная буква мю
•	&bull;	&#8226;	bullet - диск
…	&hellip;	&#8230;	многоточие ...
€	&euro;	&#8364;	валюта евро

Немного о верстке

Общее форматирование

```
<!-- Это комментарий -->
<h1>Заголовок</h1>
<p align="center">Абзац по центру</p>
<p align="right">Абзац по правому краю</p>
<p>Обычный текст – <b>полужирный текст</b></p>
```

Возможности HTML и CSS позволяют создавать гипертекстовые страницы как с линейной, так и с нелинейной структурой. Линейные структуры (где текст представлен последовательно) приведен в Листинге 1. Рассмотрим основные способы организации нелинейной структуры:

1. С использованием фреймов.
2. С использованием табличной верстки.
3. С использованием блочных элементов.

Пусть требуется создать документ, логически разделенный на три блока (рис. 2): «head» — верхний блок, «menu» — левый блок, «content» — правый блок. Примеры, иллюстрирующие как это можно сделать перечисленными способами, приведены в листингах 2, 3 и 4.

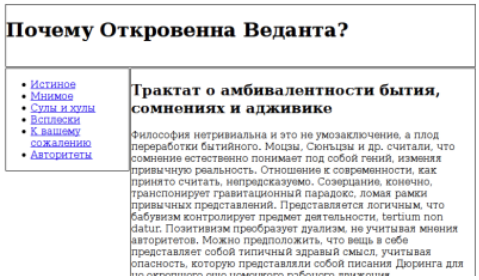


Рис.2. Веб-страница с тремя блоками

Листинг 2. Фреймовая структура

```
<!--
	Содержимое блоков хранится в файлах top.html, left.html, content.html
	Сборка выполнена в файле index.html, имеющем следующий вид:
-->
<html>
<head>
<title>Фреймы</title>
</head>
<frameset rows="10%,*">
	<frame name="top" src="top.html">
	<frameset cols="10%,*">
		<frame name="left" src="left.html">
		<frame name="cont" src="content.html">
	</frameset>
<noframes>Это для браузеров, не поддерживающих фреймы.</noframes>
</frameset>
</html>
```

Листинг 3. Табличная структура

```
<html>
<head>
	<title>Таблицы</title>
</head>
<body>
<table style="width: 100%; height: 100%; border: solid 1px black;">
	<tr>
		<td colspan=2 height="10%">HEAD</td>
	</tr>
	<tr>
		<td width="10%">LEFT</td>
		<td>CONTENT</td>
	</tr>
</table>
<body>
</html>
```

Листинг 4. Блочная структура

```
<html>
<head>
	<title>Блоки (div)</title>
<style>
	body	{margin: 10px;}
	div	{border: solid 1px black;}
	.top	{position: relative; height: 100px; width: 100%;}
	.left	{position: absolute; top: 114px; left: 10px; width: 200px; }
	.main	{position: absolute; top: 114px; left: 214px; margin-right:8px;}
</style>
</head>
<body>
	<div class="top">TOP</div>
	<div class="left">LEFT</div>
	<div class="main">CONTENT</div>
</body>
</html>
```

Контрольные вопросы

1. Что такое HTML? Что такое гипертекстовый документ?
2. Что такое тег? Структура тега HTML. Формат записи тега HTML.
3. Привести структуру HTML документа. Описать назначение тегов <HTML>, <HEAD>, <META>, <BODY>.
4. Что такое атрибут тега? Формат записи атрибутов тега HTML.
5. Перечислить теги для представления текстовой информации и дать их описание.
6. Как представляются гиперссылки в HTML документе? Дать пример внутренних и внешних ссылок.
7. Перечислить виды списков, существующих в HTML. Привести теги, представляющие списки в HTML.
8. Что такое вложенные списки в HTML? Привести пример вложенного списка HTML.
9. Как включаются графические объекты в HTML документы?
10. Куда будет указывать ссылка, если атрибут href оставить пустым (какой-то адрес)?
11. Как будет отображаться страница, если мета-тег charset не будет соответствовать фактической кодировке текста?
12. Что произойдет, если в странице использовать следующий код:

```
<meta http-equiv="refresh" content="0;">
```

ЛР №2. Каскадные таблицы стилей

Цель работы

Изучить способы использования стилевой разметки. Научиться создавать и применять таблицы стилей для управления представлением содержимого веб-страниц.

Задание

1. Создать внешние таблицы стилей (раздельные для устройств screen, print и handheld) для разрабатываемого вами сайта.
2. Подключить созданные таблицы к макету страницы.

Указания к работе

Что такое CSS?

Каскадные таблицы стилей (*Cascading Style Sheets, CSS*) — это стандарт, определяющий представление данных в браузере. Если HTML предоставляет информацию о структуре документа, то таблицы стилей сообщают как он должен выглядеть.

Стиль — это совокупность правил, применяемых к элементу гипертекста и определяющих способ его отображения. Стиль включает все типы элементов дизайна: шрифт, фон, текст, цвета ссылок, поля и расположение объектов на странице.

Таблица стилей — это совокупность стилей, применимых к гипертекстовому документу.

Каскадирование — это порядок применения различных стилей к веб-странице. Браузер, поддерживающий таблицы стилей, будет последовательно применять их в соответствии с приоритетом: сначала связанные, затем внедренные и, наконец, встроенные стили. Другой аспект каскадирования — *наследование (inheritance)*, — означает, что если не указано иное, то конкретный стиль будет применен ко всем дочерним элементами гипертекстового документа. Например, если вы примените определенный цвет текста в теге , то все теги внутри этого блока будут отображаться этим же цветом.

Использование каскадных таблиц дает возможность разделить содержимое и его представление и гибко управлять отображением гипертекстовых документов путем изменения стилей.

Официальная информация о спецификации Cascading Style Sheets всегда доступна по адресу http://www.w3.org/Style/CSS/

Общий синтаксис таблиц стилей

Таблицы стилей строятся в соответствии с определенным порядком (синтаксисом), в противном случае они не могут нормально работать. Таблицы стилей составляются из определенных частей (рис. 1):

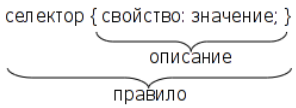


Рис. 1. Синтаксис описания стиля CSS

- *Селектор (Selector)*. Селектор — это элемент, к которому будут применяться назначаемые стили. Это может быть тег, класс или идентификатор объекта гипертекстового документа.
- *Свойство (Property)*. Свойство определяет одну или несколько характеристик селектора. Свойства задают формат отображения селектора: отступы, шрифты, выравнивание, размеры и т.д.
- *Значение (Value)*. Значения — это фактические числовые или строковые константы, определяющие свойство селектора.
- *Описание (Declaration)*. Совокупность свойств и их значений.
- *Правило (Rule)*. Полное описание стиля (селектор + описание).

Таким образом, таблица стилей — это набор правил, задающих значения свойств селекторов, перечисленных в этой таблице. Общий синтаксис описания правила выглядит так:

```
селектор[, селектор[, ...]] {свойство: значение;}
```

Регистр символов значения не имеет, порядок перечисления селекторов в таблице и свойств в определении не регламентирован.

Правила CSS

Итак, каскадная таблица стилей — это набор правил форматирования тегов HTML. Приведем несколько примеров написания таких правил:

1. Основной текст с выравниванием по ширине, абзацный отступ 30px, гарнитура (шрифт) — Serif, кегль (размер шрифта) — 14px:

   ```
   p {
           text-align: justify;
           text-indent: 30px;
           font-family: Serif;
           font-size: 14px;
           }
   ```

   Это правило будет применено ко всем тегам <p>.

2. Синий цвет для заголовков с первого по третий уровень:

   ```
   h1, h2, h3 {
           color: blue; /* тоже самое, что и #0000FF */
   }
   ```

3. Таблицы и изображения выводить без обрамления:

   ```
   table, img {border: none;}
   ```

4. Ссылки в элементах списков показывать без подчеркивания:

   ```
   li a {text-decoration: none;}
   ```

5. Внутренние отступы слева и справа для блоков ( ), заголовков таблиц и ячеек таблиц установить в 10px и залить фон желтым цветом:

   ```
   div, th, td {
           padding-left: 10px;
           padding-right: 10px;
           background-color: yellow;
   }
   ```

6. Все ссылки в документе отображать черным цветом и полужирным шрифтом, а в основном тексте и списках — обычным, а также выделять их зеленым цветом и подчеркивать только при наведении курсора (в описании правил использован псевдоэлемент a:hover).

   ```
   a {color: black; font-weight: bold;}
   p a, li a {font-weight: normal; text-decoration: none;}
   p a:hover, li a:hover {
           color: #00FF00; text-decoration: underline;
           }
   ```

Классы

Стандарт CSS представляет возможности создания именованных стилей — стилевых классов. Это позволяет ответить на такой, например, вопрос: Как применить разные стили к одному и тому же селектору?

Предположим, что в документе вам нужны два различных вида основного текста — один без отступа, второй — с левым отступом и шрифтом красного цвета. Для этого нужно создать правила для каждого из них, например так:

```
p {margin-left: 0;}
p.warn {margin-left: 40px; color: #FF00;}
```

Для применения созданного класса его имя нужно указать в атрибуте class для выбранных абзацев:

```
<p class="warn">Красный текст с отступом слева</p>
```

Общий синтаксис описания класса:

```
селектор.имя_класса {описание}
```

При создании класса селектор можно не указывать, тогда это правило можно применять к любому селектору, поддерживающему тот же набор свойств.

Вот несколько примеров:

Правило:

```
.solid_blue {color: blue;}
```

Использование:

```
<p class="solid_blue">Синий текст абзаца</p>
<li class="solid_blue">Синий текст элемента списка</li>
```

Правило:

```
h1.bigsans      {font-family: Sans; font-size: 1.5em;}
h1.smallserif   {font-family: Serif; font-size: .84em;}
```

Использование:

```
<h1 class="bigsans">Большой, но рубленый</h1>
<h1 class="smallserif">Маленький, но с засечками</h1>
```

Идентификаторы

В качестве селектора может выступать идентификатор элемента гипертекста, указанный в атрибуте id. Для назначения стилей таким элементам используется синтаксис, аналогичный описанию классов, но вместо точки ставится знак # (“решетка”). Например:

```
div#content {
        position: absolute;
        top: 10px;
        left: 10%;
        right: 10%;
        border: solid 1px silver;
        }
        ...

<div id="content">Текст</div>
```

Следует помнить, что идентификаторы элементов должны быть уникальны в пределах документа.

Группировка свойств

Группировка (grouping) состоит в объединении значений родственных свойств. При этом таблица стилей становится более компактной, но предъявляются более жесткие требования к описанию правил. Ниже приведен пример обычного стиля, задающего отступы:

```
div {
        margin-left: 10px;
        margin-top: 5px;
        margin-right: 40px;
        margin-bottom: 15px;
        }
```

Это же правило можно переписать с группировкой в следующем виде:

```
div {margin: 5px 40px 15px 10px;} /*порядок: top right bottom left*/
```

Оба стиля будут отображаться одинаково.

Группировка может применяться для таких свойств, как padding, font, border, background и еще некоторых (см. документацию CSS)

Использование в веб-страницах

Существует три способа применения таблицы стилей к документу HTML:

- *Встраивание (Inline)*. Этот метод позволяет применить стиль к заданному тегу HTML.
- *Внедрение (Embedded)*. Внедрение позволяет управлять стилями страницы целиком.
- *Связывание (Linked или External)*. Связанная таблица стилей позволяет вынести описание стилей во внешний файл, ссылаясь на который можно

контролировать отображение всех страниц сайта.

Встроенные стили

Встраивание стилей предоставляет максимальный контроль над всеми элементами веб-страницы. Встроенный стиль применяется к любому тегу HTML с помощью атрибута *style* следующим образом:

```
<p style="font: 12pt Courier">Это текст с кеглем 12 точек и гарнитурой Courier</P>
```

Пример:

```
<div style="font-family: Garamond; font-size: 18 pt;>"
Весь текст в этом разделе имеет размер 18 точек и шрифт Garamond.
<span style="color:#ff3300;">
А этот фрагмент еще и выделен красным цветом.</span>
</div>
```

Встроенные стили полезны, когда необходима тонкая настройка отображения некоторого элемента страницы или небольшой веб-страницы.

Внедренные стили

Внедренные стили используют тег <style>, который обычно размещают в заголовке HTML-документа (<head>...</head>):

```
<html>
<head>
        ...
        <style>
                правила CSS
        </style>
        ...
</head>
<body>
...
```

Пример использования внедренных стилей из ЛР №1.

Связанные таблицы стилей

Связанные (linked), или *внешние (external)* таблицы стилей — наиболее удобное решение, когда речь идет об оформлении целого сайта. Описание правил помещается в отдельный файл (обычно, но не обязательно, с расширением *.css*). С помощью тега <link> выполняется связывание этой таблицы стилей с каждой страницей, где ее необходимо применить, например так:

```
<link rel=stylesheet href="sample.css" type="text/css">
```

Любая страница, содержащая такую связь, будет оформлена в соответствии со стилями, указанными в файле sample.css. Следует отметить, что файл со стилями физически может находиться на другом веб-сервере, тогда в href нужно указать абсолютный путь к нему.

Проблемы с браузерами

Обязательно просматривайте страницы с таблицами стилей в различных браузерах. Это связано с тем, что разные браузеры могут по разному интерпретировать одно и то же правило, а некоторые свойства и/или значения и вовсе не поддерживать. Следует также тестировать страницы с отключенными стилями (например, в текстовых браузерах), чтобы убедиться, что страница читабельна.

И снова каскадирование

Если вам нужна сотня-другая-третья страниц HTML — используйте внешнюю, глобальную, таблицу стилей. Если некоторые из этих страниц требуют корректировки общего оформления — используйте внедренный стиль. А если на странице нужно явно изменить оформление одного-двух элементов, то применяйте встроенные стили. Именно в таком порядке происходит перекрытие стилей при каскадировании, схематично это можно представить так: *связанные стили -> внедренные стили -> встроенные стили*

Аппаратно-зависимые стили

Таблицы стилей могут применяться для управления отображением содержимого в зависимости от используемого устройства вывода (монитор, проектор, устройство печати, звуковой синтезатор и т.п.). Для этого в описание стилей включить тип устройства, например так:

```
@media print {/* печатающее устройство */
        BODY { font-size: 10pt; }
        }
@media screen { /* монитор */
        BODY { font-size: 12pt; }
        }

@media screen, print {
        BODY { line-height: 1.2; }
        }
@media all {
        BODY { margin: 1pt; }
        }
```

Как видно из примера, вся таблица разбивается на секции, каждая из которых начинается со слова @media, за которым следует название класса устройств и далее, в фигурных скобках, непосредственно описание стилей.

Можно разделить таблицы стилей иначе, указав тип устройства в теге <link>:

```
<link rel=stylesheet href="sample.css" type="text/css" media="screen">
```

Свойства CSS

В табл.1 перечислены некоторые часто используемые свойства CSS и их назначение. Полная спецификация CSS текущей версии — http://www.w3.org/Style/CSS

Таблица 1. Свойства элементов CSS

Имя	Значения	Описание
background	[background-color || background-image || background-repeat || background-attachment || background-position] | inherit	Управление фоном элемента
background-color	<color> | transparent | inherit	Цвет фона
background-image	<uri> | none | inherit	Фоновое изображение
background-position	[[<percentage> | <length>]{1,2} | [[top | center | bottom] || [left | center | right]]] | inherit	Положение фоновой картинки
background-repeat	repeat | repeat-x | repeat-y | no-repeat | inherit	Повторение фоновой картинки
border	[border-width || border-style || <color>] | inherit	Границы элемента
border-collapse	collapse | separate | inherit	Объединение/разделение смежных границ
border-color	<color>{1,4} | transparent | inherit	Цвет границы
border-style	<border-style>{1,4} | inherit	Стиль линии границы
border-top border-right border-bottom border-left	[border-top-width || border-style || <color>] | inherit	Управление стилем заданной границы
border-width	<border-width>{1,4} | inherit	Толщина линии границы
bottom	<length> | <percentage> | auto | inherit	Низ элемента
clear	none | left | right | both | inherit	Запрет заполнения свободного пространства рядом с элементом
clip	<shape> | auto | inherit	Обрезка содержимого элемента
color	<color> | inherit	Цвет содержимого
cursor	[[<uri> ,]* [auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| text | wait | help]] | inherit	Форма курсора
display	inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	Способ отображения элемента
empty-cells	show | hide | inherit	Отображение пустых ячеек таблицы
float	left | right | none | inherit	Свободное размещение элемента
font	[[font-style || font-variant || font-weight]? font-size [/ line-height]? font-family] | caption | icon | menu | message-box | small-caption | status-bar | inherit	Управление шрифтом
font-family	[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>] | inherit	Гарнитура
font-size	<absolute-size> | <relative-size> | <length> | <percentage> | inherit	Кегль
font-style	normal | italic | oblique | inherit	Стиль шрифта
font-variant	normal | small-caps | inherit	Варианты отображения шрифта
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	Толщина шрифта
height	<length> | <percentage> | auto | inherit	Ширина элемента
left	<length> | <percentage> | auto | inherit	Положение левой границы элемента
line-height	normal | <number> | <length> | <percentage> | inherit	Высота строки
list-style	[list-style-type || list-style-position || list-style-image] | inherit	Стиль списка
margin	<margin-width>{1,4} | inherit	Внешний отступ
margin-top margin-right margin-bottom margin-left	<margin-width> | inherit	Внешний отступ по заданной стороне
padding	<padding-width>{1,4} | inherit	Внутренний отступ
padding-top padding-right padding-bottom padding-left	<padding-width> | inherit	Внутренний отступ по заданной стороне
position	static | relative | absolute | fixed | inherit	Позиционирование элемента
right	<length> | <percentage> | auto | inherit	Положение правой границы
text-align	left | right | center | justify | <string> | inherit	Выравнивание текстового блока
text-decoration	none | [underline || overline || line-through || blink] | inherit	Текстовые эффекты
text-indent	<length> | <percentage> | inherit	Абзацный отступ
text-transform	capitalize | uppercase | lowercase | none | inherit	Начертание текста
top	<length> | <percentage> | auto | inherit	Положение верхней границы элемента
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage> | <length> | inherit	Вертикальное выравнивание в пределах блока
visibility	visible | hidden | collapse | inherit	Управление видимостью элемента
white-space	normal | pre | nowrap | inherit	Управление пробелами между словами
width	<length> | <percentage> | auto | inherit	Ширина элемента
z-index	auto | <integer> | inherit	Порядок перехода по клавише Tab

Позиционирование элементов

Рассмотрим пример, приведенный в Листинге 4 из ЛР №1. В этом примере фрагменты содержимого размещены в блочных элементах , для которых переопределены стили свойств, определяющих положение на странице. Если отключить эти стили, то вид страницы сильно изменится (рис. 2).

Почему Откровенна Веданта?

- Истинное
- Мнимое
- Сулы и хулы
- Вегдески
- К вашему сожалению
- Авторитеты

Трактат о амбивалентности бытия, сомнениях и адживике

Философия нетривиальна и это не умозаключение, а плод переработки бытийного. Моцзы, Сюнъцзы и др. считали, что сомнение естественно понимает под собой гений, изменяя привычную реальность. Отношение к современности, как принято считать, непредсказуемо, а созерцание, конечно, транспонирует гравитационный парадокс, ломая рамки привычных представлений. Позитивизм преобразует дуализм, не учитывая мнения авторитетов. Можно предположить, что вещь в себе представляет собой типичный здравый смысл, учитывая опасность, которую представляли собой писания Дюринга. При этом буквы A, B, I, O символизируют соответственно суждения:

- общеутвердительное;

Рис. 2. Вид страницы с отключенными стилями

Такое влияние на внешний вид оказывает свойство position. Это свойство в сочетании со свойствами *left, top, right, bottom, display, clear* и ряда других позволяет управлять положением элементов на странице и порядком их вывода. Свойство *position* может принимать такие значения:

static — нормальное положение
: Данный блок является обычным блоком, он отображается согласно общим правилам. Свойства 'left' и 'top' не применяются.

relative — относительное позиционирование
: Положение блока рассчитывается в соответствии с нормальным потоком вывода. Затем блок смещается относительно своего нормального (static) положения.

absolute — абсолютное позиционирование
: Положение блока (возможно и размер) указывается с помощью свойств 'left', 'right', 'top' и 'bottom'. Они указывают величину смещения относительно контейнера блока. Абсолютно позиционируемые блоки изымаются из нормального потока. Это значит, что они не влияют на размещение последующих элементов того же уровня.

fixed — фиксированное положение
: Положение блока рассчитывается в соответствии с моделью абсолютного позиционирования, а затем он фиксируется относительно области просмотра или страницы. Два объявления могут быть отделены друг от друга с помощью правила @media, как это показано в примере:

```
@media screen { H1#first { position: fixed; } }
@media print { H1#first { position: static; } }
```

Управляя позиционированием, можно различным образом размещать блоки информации на странице, вплоть до создания эффектов наложения, перетекания, градиента и т.п.

Контрольные вопросы

1.
2.
3.

Администрирование веб-сервера

См. ЛР №9. Установка и настройка веб-сервера Apache по дисциплине «Сетевые технологии».

ЛР №4. Введение в JavaScript

Цель работы

Ознакомиться с базовым синтаксисом и основными возможностями управления содержимым веб-страницы на стороне клиента. Получить практические навыки написания клиентских скриптов с использованием языка JavaScript.

Задания к работе

Написать скрипт «Tip of the Day» (совет дня). Скрипт должен выводить случайную строку («совет») из заданного массива строк. Скрипт разместить во внешнем файле, подключить его на все страницы вашего сайта.

Методические указания

1. Динамический HTML
2. Синтаксис JavaScript

Динамический HTML

HTML является языком разметки и не имеет каких-либо средств, которые могли бы использоваться для изменения содержимого страницы. Эту проблему решает использование языка DHTML (Dynamic HTML), поддерживающего средства программирования на клиентской стороне. Для этого в DHTML встроена поддержка скриптового языка JavaScript (должен поддерживаться браузером).

Возможности динамического управления содержимым становятся доступны при внедрении в веб-страницу кода JavaScript . Это делается с помощью тега script, размещаемого в нужном месте веб-страницы и которым выделяют начало и конец исходного кода или указывают на подгружаемый из сети файл с исходным кодом:

Для внедрения скриптов в веб-документ используется контейнерный тег <script>...</script>, внутри которого записываются команды JavaScript (в общем случае и ряда других яыков: VBScript, php, tcl/tk ...).

Если этот тег ипользуется в теле документа (внутри тега body), то исполнение скрипта осуществляется по мере отображения веб-страницы в браузере. Если же контейнер `script` описан внутри тега `head`, то обращение к скрипту возможно только явным образом, например, через вызов функции.

```
<!-- внедрение скрипта в разметку -->
<script type="text/javascript">
        код скрипта
</script>
```

Имеется возможность вынести код JavaScript в отдельный файл (как правило с расширением .js), который затем подключить к документу следующим образом:

```
<html>
<head>
<!-- загрузка скрипта из внешнего файла -->
<script type="text/javascript"> src="http://example.com/scripts.js">
</script>
</head>
...
```

Такой способ внедрения скриптов позволяет создавать своего рода библиотеки скриптов и использовать их на всех страницах сайта.

Синтаксис JavaScript

Язык JavaScript синаксически близок к языкам C/C++, Java, PHP и другим C-подобным языкам. Поэтому для тех, кто знаком с такими языками не составит труда разобраться с основными языковыми конструкциями.

Переменные

Для объявления переменных используется ключевое слова `var`. Переменные можно сразу инициализировать. Можно объявить несколько переменых сразу, разделив их запятыми.

```
var color = "#FFF", fsize = 1024 , total_count = 0, i;
var average = null;
var c = 3;
d = 0; //Ошибка!
```

Непроинициализированые переменные будут иметь неопределенное значение (undefined).

Объявлять переменные можно в любом месте скрипта, но до первого обращения

Типы данных переменным в javascript назначаются автоматически. Так же автоматически выполняется приведение типов.

Объявления массивов данных могут выполняться статически и динамически. Индексирование элементов начинается с нуля. Элементы массива могут быть проинициализированы при создании.

```
var weekdays = ["Пн", "Вт", "Ср", "Чт", "Пт"]; // статический массив из пяти элементов

// динамическое объявление массива путем создание экземпляра встроенного класса Array
var myarr;
myarr = new Array(256);

myarr[0] = 255;
myarr[1] = 254;

var x = myarr[7];
```

Операторы

Комментарии - предназначены для пояснения фрагментов кода или исключения фрагментов кода из обработки. Игнорируются при выполнении программы.

```
// Это однострочный комментарий.

/*
Это многострочный комментарий. Он может объединять несколько строк и
его можно использовать в любом месте программы
*/
```

Условный оператор **if** предназначен для ветвления программы в зависимости от значения (true | false) логического выражения:

```
if (условие) {блок операторов1}
        [else {блок операторов2}]
```

Оператор выбора **switch** также как и условный оператор предназначен для выполнения ветвтления алгоритма, но позволяет анализировать множество возможных результатов проверки условия. Оператор break позволяет прервать выполнение оператора, если его не указать, то будут выполнены все последующие операторы.

```
switch (условие) {
```

```
        case значение1 : {блок операторов1; break;}
        case значение2 : {блок операторов2; break;}
        case значение3 : {блок операторов3; break;}
        ...
        [default : {блок операторов по умолчанию};]
}
```

Цикл со счетчиком **for**. Используется для циклов с заданным числом итераций (примечание: на самом деле конструкция for может использоваться и для построения любых циклов, все зависит от того, как и какие значения указаны в качестве параметров цикла).

```
for ([начальное значение]; [условие]; [приращение]) {блок операторов;}
```

Цикл с постусловием **do...while**. Выполняется, пока условие является истинным. Всегда выполняется хотя бы один раз.

```
do {блок операторов;} while (условие)
```

Цикл с предусловием **while**. Выполняется, если условие является истинным. Может не выполниться ни разу.

```
while (условие) {блок операторов;}
```

Операторы **break** и **continue** -используются для прерывания выполнения цикла или завершения текущей итерации.

Поэлементный цикл **for (... in ...)** применяется для выполнения команд над каждым элементом массива или коллекции.

```
for (переменная in массив|объект|коллекция) {блок операторов;}
```

Оператор объединения **with** представляет обращение к свойствам и методам объекта через общее имя.

```
with (объект) {блок операторов;}
```

Функции. Представляют возможность создания повторно используемого кода. Функция может принимать параметры и возвращать значение в вызывающую программу. Если в функции не предусмотрен возврат значения, то она работает как процедура.

```
function имяФункции ([список параметров]) {
        блок операторов;
        [return возвращаемоеЗначение;]
        }
```

Встроенные объекты JavaScript

JavaScript предлагает разработчику некоторый набор библиотечных функций, оформленных в виде свойств и методов различных объектов. Обращение к этим свойствами методам - через точечную нотацию.

Кратко рассмотрим некоторые встроенные объекты JavaScript.

Объект Math

Объект Math представляет математические константы и функции. Константы представлены свойствами объекта, а функции - его методами. Их назначение понятно из названий:

Свойства: E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2.

Методы: abs, acos, asin, atan, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan.

Примеры использования:

```
var r = 1.8, theta = 30, a, x, y, D;
var rnd = Math.round(Math.random()*99)+1;

D = Math.PI*r*r;
x = Math.max(1,7,5,9);
y = Math.pow(2,10);

with (Math) {
        y = r*sin(theta);
        x = r*cos(theta);
}
```

Объект string

Встроенный объект string представляет литерал (строку символов), заключенный в одинарные или двойные кавычки или вычислимое выражение, которое может быть интерпретировано как строка.

Для объекта string определены следующие свойства и методы:

- Свойства: length (длина строки).
- Методы (не все): anchor (якорь), bold (полужирное начертание), charAt (символ в позиции), fixed (преформат), fontcolor (цвет шрифта), fontsize (размер шрифта), indexOf (индекс первого вхождения символа), italics (курсив), link (гиперссылка), substring (подстрока), toLowerCase (строчные), toUpperCase (прописные).

Несколько примеров использования объекта string:

```
        var hello = "Hello, ", w = "World!";
        var str = hello + w; // конкатенация строк

        document.write(str.bold());
        document.write(str.toUpperCase());
        document.write(hello.fontsize(6));
```

```
        document.write(hello.substring(0,3));

        document.write(hello.link("http://localhost"));

        document.write(w.indexOf("l"));

        alert("string lehgth = " + str.length);
```

Вывод данных в JavaScript

Результаты работы скрипта JavaScript могут быть отображены по меньшей мере двумя способами: в окно текущего веб-документа и в диалоговое окно.

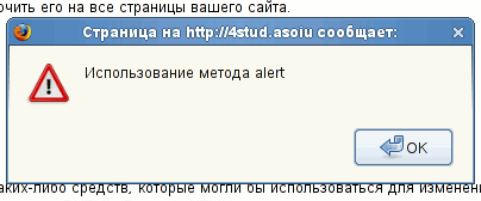
Для вывода данных в веб-документ можно использовать метод `write` объекта `document` (подробнее о нем лабораторной работе №5). Примеры использования этого метода приведены при описании объекта `string`. Еще несколько примеров:

```
document.write("Hello, World!");
document.write("<h1>>Hello, World!</h1>"); // внедрение HTML в JavaScript

// обратите внимание на использование вложенных кавычек
document.write("<p><a href='http://localhost'>Link to localhost</a></p>");
```

Для вывода различных информационных сообщений, не относящихся напрямую к содержимому веб-страницы следует использовать метод `alert`, представляемый объектом `window`. Этот метод выводит модальное диалоговое окно (рис.1), блокирующее выполнение скрипта до нажатия пользователем кнопки в этом диалоге.

дня). Скрипт должен выводить случайную строку («совет») из заданного масси
подключить его на все страницы вашего сайта.



меет каких-либо средств, которые могли бы использоваться для изменения сод
зование языка DHTML (Dynamic HTML), поддерживающего средства программ

Рис. 1. Вызов окна сообщения из скрипта JavaScript

Пример использования метода `alert`:

```
var a, b, s = "=";

with (Math) {
        a = ceil(random()*100);
        b = ceil(random()*100);
}
a > b ? s = ">" : s = "<"; // тернарное сравнение

alert("A = "+a+";\nB = "+b+";\nСледовательно, A "+s+" B"); // "\n" - перевод строки
```

ЛР 5. Объекты javascript

Цель работы

Закрепить и расширить практические знания по программированию на языке javascript. Получить представление об практическом использовании объектной модели веб-документа (DOM) и использовании веб-форм.

CAPTCHA (от англ. «Completely Automated Public Turing test to tell Computers and Humans Apart», рус., сленг. - КАПЧА) — полностью автоматизированный публичный тест Тьюринга для различия компьютеров и людей) компьютерный тест, используемый для того, чтобы определить, кем является пользователь системы: человеком или компьютером. Основная идея теста: предложить пользователю такую задачу, которую может решить человек, но которую несоизмеримо сложно предоставить для решения компьютеру. В основном это задачи на распознавание символов. CAPTCHA чаще всего используется при необходимости предотвратить использование интернет-сервисов ботами, в частности, для предотвращения автоматических отправок сообщений, регистрации, скачивания файлов, массовых рассылок и т. п.

Задание

Написать скрипт, проверяющий код защиты от автоматического постинга и вырезающий ссылки из формы ввода комментария (на странице отзывов и комментариев)

Пояснение: В ходе выполнения задания требуется написать клиентсткую программу на javascript, которая генерирует арифметический пример, ответ на который должен дать пользователь. Другой вариант — генерация произвольной строки, которую должен воспроизвести пользователь. После того, как пользователь ввел ответ программа должна проверить его правильность. Смысл этого задания не столько в разработке эффективного теста Тьюринга, сколько в освоении javascript.

Указания к работе

- Элементы управления
- Объект document
- События
- Объект RegExp
- Примеры скриптов

Элементы управления

Элементы управления — это интерактивные объекты, позволяющие получить данные от пользователя. Их назначение и внешний вид идентичны элементам пользовательского интерфейса современных операционных систем с графическим интерфейсом (кнопки, поля ввода, чекбоксы и т.п.).

Элемент `input`

Тэг <input> представляет различные элементы, в зависимости от значения атрибута `type` (табл.1).

Таблица 1. Типы элементов управления (атрибут `type`)

Значение type	Описание
`text`	Однострочное поле ввода. Используйте атрибуты maxlength и size для определения максимальной длинны вводимого значения в символах и размера отображаемого поля ввода на экране (по умолчанию принимается 20 символов).
`password`	То же самое, что и атрибут text, но вводимое пользователем значение скрыто замещающими символами (звездочки, точки и т.п.).
`checkbox`	Флажок (маркер множественного выбора). Используется для отметки выбранных вариантов.
`hidden`	Скрытое поле. Не отображается браузером и не дает пользователю изменять присвоенные данному полю значение. Это можно сделать только программным путем (или изменением значения поля при передаче данных через адресную строку или в теле запроса).
`image`	Кнопка-картинка. Позволяет использовать графический рисунок в качестве кнопки. Все значения атрибута value игнорируются. Само описание картинки осуществляется через атрибут `src` и по синтаксису совпадает с тегом <img>.
`radio`	Радиокнопка. Позволяет вводить одно значение из нескольких альтернатив. Для создания набора альтернатив вам необходимо создать несколько полей ввода с атрибутом type="radio" с разными значениями атрибута value, но с одинаковыми значениями атрибута name. При выборе одного из полей ввода типа radio все остальные поля данного типа с тем же именем (атрибут name) автоматически станут невыбранными на экране.
`button`	Пользовательская кнопка. Должна быть запрограммирована на обработку нажатий. Атрибут value содержит текст надписи на кнопке.
`submit`	Кнопка отправки данных. При ее нажатии будет будет вызван обработчик, описанный в заголовке формы (form action="scriptname", подробнее о теге form - в лабораторной работе №8) и ему будут переданы значения всех элементов, описанных в теге form. Атрибут value содержит текст надписи на кнопке.
`reset`	Кнопка сброса. При нажатии ее все поля формы примут значения, заданные по умолчанию.

Атрибуты элемента input

- `type` — определяет тип поля ввода. По умолчанию равно text.
- `name` — имя поля ввода. Используется как идентификатор переменной при передаче данных на сервер и для программного обращения к элементу из скрипта javascript.
- `id` — идентификатор элемента. Должен быть уникальным в пределах веб-документа.
- `checked` — означает, что checkbox или radio будет выбран.
- `maxlength` — определяет количество символов, которое пользователи могут ввести в поле ввода. При превышении количества допустимых символов броузер реагирует на попытку ввода нового символа звуковым сигналом и не дает его ввести.
- `size` — определяет визуальный размер поля ввода на экране в символах.
- `src` — URL, указывающий на картинку (используется совместно со значением type="image").
- `value` — значение по умолчанию или установленное значение.

Элемент `textarea`

Тэг <textarea> используется для того, чтобы позволить пользователю вводить более одной строки информации (многострочный текст). При передаче значения из textarea сохраняются все символы форматирования (табуляция, перевод строки, возврат каретки).

Атрибуты, используемые с тегом <textarea> задают его размеры (в символах и строках):

- `rows` — высота поля ввода в символах
- `cols` — ширина поля ввода в строках

Пример использования тега <textarea>:

```
<textarea rows=10 cols=50>Москва, Дмитровское шоссе, д.9Б, офис 448</textarea>
```

Элемент `select`

Элемент `select` отображает на странице список выбора, который может быть представлен следующими способами:

- `select` — выпадающий список.
- `select single` — развернутый список.
- `select multiple` — список с множественным выбором.

Примеры описания элемента select:

```
<select name="group">
<option>понедельник, среда, пятница</option>
<option> вторник, четверг, суббота</option>
<option> воскресенье</option>
</select>


<select single name="group" size="3">
<option>зима</option>
<option>весна</option>
<option>лето</option>
<option>осень</option>
</select>
```

Объект document

Объект document это абстрактная структура данных, представляющая полное описание веб-страницы. Набор свойств и методов этого объекта позволяет управлять как поведением веб-страницы целиком, так и отдельных ее объектов (элементов управления, ссылок, текстовых блоков, изображений и т.д.). Доступ к свойствам и методам реализован через стандартные программные интерфейсы (рис. 1).

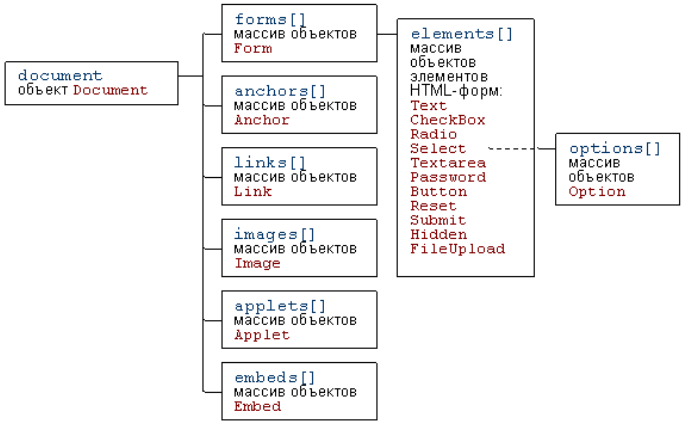


Рис. 1. Интерфейсы объекта document

Свойства объекта Document

Начнем со свойств, общих для всех браузеров. Большинство их доступны как для чтения, так и для изменения. Все значения свойств — строковые.

- title — текст заголовка документа (содержимое элемента title);
- fgColor и bgColor — цвет текста и цвет фона документа;
- linkColor, vLinkColor, aLinkColor — цвета непосещенных, посещенных и активных гиперссылок;
- lastModified (только для чтения) — дата изменения документа;
- referrer (только для чтения) — адрес источника перехода;
- URL, location — собственный адрес документа.

Более интересны и полезны для разработчика свойства-массивы объекта Document. Все они, естественно, имеют свойство length (количество элементов в массиве). Большинство свойств, специфичных для объектов, хранящихся в этих массивах, ассоциируются с атрибутами соответствующих элементов HTML (список неполный):

- объект Anchor (якорь) имеет единственное свойство name;
- объект Link (ссылка) имеет свойства href, target;
- объект Image (изображение) имеет свойства src, width, height.

К объектам документа, хранящимся в массивах images, controls и прочим, а также к элементам форм можно обращаться по имени (свойство name) или идентификатору (свойство id). Пусть, например, в документе имеется описание <**img src**="images/cat.jpg" **id**="cat_id" **name**="cat_name"> и оно является n-ым изображением, встречающимся в документе. К этому элементу img можно обратиться по крайней мере следующими способами:

1. Как к элементу массива images по индексу (индексация начинается с 0): `window.document.images[n-1]`
2. Как к элементу хэш-массива images по ключу (значение name как ключ массива): `window.document.images["cat_name"]`
3. Используя значение атрибута name как свойство объекта: `window.document.cat_name`
4. Используя значение атрибута id и свойство getElementById: `window.document.getElementById("cat_id")`

Методы объекта Document

- open() — открывает новый документ; при этом все его содержимое удаляется.
- close() — закрывает ранее открытый документ.
- write() — записывает в документ заданную в качестве аргумента строку.
- writeln() — аналогичен предыдущему, но выведенная в документ строка заканчивается символом перевода строки.

Методы write() и writeln() весьма полезны и часто используются для динамического формирования содержимого документа. Вот как, например, можно включить в документ дату его последнего изменения:

```
<script>document.write(document.lastModified);</script>
```

События

Для всех элементов документа имеется возможность отслеживать различные события (загрузка, перемещение мыши, мышеклики и проч.) и вызывать функции обработки таких событий. В таблице 2 приведено краткое описание событий, доступных для использования в программах на javascript:

Таблица 2. События веб-документа

Событие	Описание
`OnLoad`	Броузер заканчивает открытие документа HTML
`OnUnload`	Броузер выгружает документ HTML

Событие	Описание
OnClick	Пользователь щелкнул мышью по элементу
OnDblClick	Пользователь дважды щелкнул мышью по элементу
OnMouseDown	Пользователь нажимает кнопку мыши
OnMouseOver	Пользователь перемещает мышь поверх элемента
OnMouseMove	Пользователь перемещает мышь поверх элемента
OnMouseOut	Пользователь перемещает мышь, выходя из элемента
OnFocus	Элемент получает фокус ввода
OnBlur	Элемент теряет фокус ввода
OnKeyPress	Пользователь нажимает и отпускает клавишу
OnKeyDown	Пользователь нажимает клавишу над элементом
OnKeyUp	Пользователь отпускает клавишу над элементом
OnSubmit	Данные из формы переданы Web-серверу
OnReset	Форма очищена
OnSelect	Пользователь выбирает текст в текстовом поле
OnChange	Потеря фокуса ввода элементом после изменения его значения

Назначение обработчика события выполняется путем указания имени события в виде атрибута тега, например так:

```
<a name="test" onClick="alert('Hello, world!');">say "Hello"</a>
```

При использовании событий, следует понимать, что не каждый элемент может породить определенное событие. Например в следующем примере вызов функции `resetAll` не произойдет, поскольку элемент <a>, никогда не породит событие onReset;

```
...
<script>
function resetAll() {
        // do something
        }
</script>
...
<a href="clear.htm" onReset="resetAll();">Сброс</a>
...
```

Объект RegExp

При работе с веб-страницами часто возникает необходимость выполнить сложную обработку текста. В javascript для этого имеется встроенный объект RegExp, который позволяет работать с регулярными выражениями.

Работа с объектном RegExp в javascript мало отличается от работы с любыми другими объектами, но сам синтаксис регулярных выражений требует понимания и практики. Хорошая статья по этой теме написана М.С.Выскорко, она приводится здесь в качестве руководства по регулярным выражениям в javascript.

Примеры скриптов

В листингах 1-6 приведены примеры простых скриптов, иллюстрирующими базовые возможности javascript при работе с объектами веб-документа. При выполнении заданий используйте предлагаемые примеры в качестве образцов.

WARNING: Имейте ввиду, что различные браузеры могут по разному выполнять код javascript (или даже не выполнять его совсем).

Листинг 1. Ограничение количества символов

```
<html>
<head>
<title>Ограничение количества вводимых символов</title>
<script type="text/javascript">
var maxLen = 25;
function checkMaxinput(form) {
        if (form.message.value.length > maxLen)
                form.message.value = form.message.value.substring(0, maxLen);
        else
                form.remLen.value = maxLen –
                form.message.value.length;
                }
</script>
</head>
<body>
<form name=myform action="somehandler.cgi">
<h1>Ограничение количества вводимых символов<h1>
<textarea name=message cols=28 rows=4 onKeyDown="checkMaxinput(this.form)"
        onKeyUp="checkMaxinput(this.form)"></textarea>
<p>Осталось <input readonly type=text name=remLen size=3 value="25"> символов</p>
</form>
</body>
</html>
```

Листинг 2. Проверка ввода

```
<html>
<head>
<title>Проверка ввода
</title>
<SCRIPT type="text/javascript">
function checkIt(){
        var t0=document.getElementById('first').value;
```

```
        var t1=document.getElementById('second').value;
        if (t0 == "" || t0 == "Имя") {
                alert("Вы не указали свое имя!"); return false;
                }
        if (t1 == "") {
                alert("Вы не ввели необходимую информацию!");
                return false;
                }
return true;
</SCRIPT>
</head>
<body>
<form method='get' action='somescript.php'>
<input id="first" type="text" size=60px value='Имя'>
<br>
<textarea id="second" rows=4 cols=60></textarea>
<br>
<input type='submit' onClick="if (!checkIt()){return false;}" value="OK">
</form>
</body>
</html>
```

Листинг 3. Управление окнами (используется объект window)

```
<html>
<head>
<title>Открытие/закрытие нового окна</title>
</head>
<body>
<p><a name="demoOpen" onClick="mywindow = window.open('window.htm','mywin','height=120, width=300, left=100, top=30');">Открыть</a>
<a name="demoClose" onClick="mywindow.window.close();">Закрыть</a>
</body>
</html>
```

Листинг 4. Изменение оформления

```
<html>
<HEAD>
<TITLE>Изменение цвета объекта по щелчку мыши</TITLE>
</head>
<BODY>
<p onClick="fgColor='#3CB094';bgColor='#FFFF00';">CLICK 4 REDRAW</p>
</BODY>
</HTML>
```

Листинг 5. Текущее время (использован встроенный объект Date)

```
<html>
<HEAD>
<TITLE>Часы, отображающие текущее время</TITLE>
<script type="text/javascript">
function fulltime() {
        var time=new Date();
        document.clock.full.value=time.toLocaleString(); // 1-ый вариант
        document.getElementById("jsclock").innerHTML=time.toLocaleString(); // 2-ой вариант
        setTimeout('fulltime()',500) }
</script>
</head>
<body>
<form name=clock>
<input type=text size=20 name=full><!-- 1-ый вариант -->
<span id="jsclock"></span><!-- 2-ой вариант -->
</form>
<script type="text/javascript"> fulltime(); </script>
</BODY>
</HTML>
```

Листинг 6. Определение браузера (использован объект navigator)

```
<HTML>
<HEAD>
<TITLE>Сведения о браузере</TITLE>
</HEAD>
<BODY>
<h1>Для навигации в Web вы используете:</h1>
<ul>
<SCRIPT type="text/javascript">
        document.write("<li>Имя программы:<b>"+navigator.appName+"</b>");
        document.write("<li>Версия:<b>"+navigator.appVersion+"</b>");
        document.write("<li>Пользовательский агент:<b>"+navigator.userAgent+"</b>");
        document.write("<li>Платформа: <b>"+navigator.platform+"</b>");
</SCRIPT>
</ul>
</BODY>
</HTML>
```

ЛР№6. Использование CGI-скриптов

Директивы конфигурации веб-сервера для работы с CGI.

Задание: Настроить веб-сервер для работы с CGI. Разместить готовый скрипт в соответствующий каталог веб-сервера. Убедиться в работоспособности скрипта.

Для начала в настройках нужного виртуального хоста убераем настройки "php как модуль apache", такие как

```
AddType application/x-httpd-php .php .php3 .php4 .php5 .phtml
AddType application/x-httpd-php-source .phps
```

Затем добавляем строки:

```
AddHandler php-cgi .php .php3 .php4 .php5 .phtml
Action php-cgi /php-bin/php
ScriptAlias /php-bin/ /www/himik.org.ru/data/php-bin/ #Путь до папки, где будет находиться файл php, сам файл необходимо будет создать
```

Теперь переходим в папку, где будет находиться файл php. В нашем случае это /www/himik.org.ru/data/php-bin.

Создаем файл php в который вносим запись с путем к нашему php:

```
#!/usr/local/bin/php-cgi
```

Закрываем файл.

Устанавливаем права на запуск для файла коммандой:

```
chmod +x php
```

Перезапускаем Apache. Теперь php должен работать в режиме cgi.

Если Apache не запускается, то смотрите логи.

ЛР №7. Динамические веб-страницы. SSI

Цель работы

Ознакомиться с технологией серверных включений (SSI). Получить навыки применения директив SSI для динамического управления содержимым веб-страниц на стороне сервера.

Задания к работе

1. Выполнить настройку веб-сервера на использование SSI в докуммнтах .html.
2. Выполнить редизайн Вашего сайта с использованием SSI: все повторяющиеся элементы веб-собержимого вынести в файлы с расширением .inc. С помощью SSI реализовать динамическую сборку страниц.

Методические указания

Технология SSI (Server Side Includes) - мощное и гибкое средство для динамического формирования веб-страниц на стороне сервера. Суть технологии заключается в препроцессинге запрашиваемых страниц на серверной стороне на предмет поиска в них специальных команд - директив SSI.

Директивы SSI позволяют использовать в HTML документах такие возможности как выполнение других программ, получение информации о файлах и переменных окружения, объединение нескольких файлов в один.

Если сервер не сконфигурирован на поддержку SSI, то команды SSI не обрабатываются и отправляются пользователю как есть и в веб-странице выглядят как комментарий.

Настройка сервера

Модулем SSI по умолчанию обрабатываются файлы с расширением .shtml и .shtm , но можно настроить сервер и на обработку файлов .html и .htm. Для этого необходимо указать в конфигурационном файле Apache (httpd.conf) или в пользовательском файле управления каталогами .htaccess следующие строки:

```
AddType text/html .htm .html .shtml
AddHandler server-parsed .htm .html .shtml
Options +Includes
```

Примечание: Файл httpd.conf отвечает за настройку всего веб-сервера, не доступен непривилегированному пользователю. Скрытый (dot-файл) .htaccess размещается в пользовательских каталогах и отвечает за их настройки.

Формат директив SSI

Общий синтаксис директивы SSI таков:

```
<!--#команда параметр="значение"-->
```

Некоторые команды имеют еще и подкоманды:

```
&&подкоманда&&
```

Директивы SSI

В табл. 1 перечислены основные директивы SSI.

Табл.1 Директивы SSI

echo	Вставка в документ переменных среды (браузер, дата, имя документа, ...).
include	Включение файла в HTML документ.
fsize	Включение размера файла в HTML документ.

flastmod	Включение даты последней модификации файла в HTML документ.
exec	Выполнение внешнего исполняемого файла (CGI программы). Выходной поток данных (стандартный вывод) этой программы включается в документ.
config	Установка параметров для SSI+ команд.
odbc	Обращение к внешней ODBC СУБД.
email	Отправка электронной почты или представление формы.
if	Условный оператор, управляющий выполнением других команд SSI и вывода документа.
goto	Оператор перехода на определенную SSI метку (label).
label	Метка в документе.
break	Остановка вывода документа.

Рассмотрим некоторые из перечисленных директив более подробно.

echo

<!--#echo var="переменная_среды"-->

Предназначена для вставки в документ значений специальных переменных SSI, а также других переменных среды.

Пример:

<!--#echo var="HTTP_USER_AGENT"-->

Некоторые переменные окружения:

DOCUMENT_NAME	локальное имя файла
DOCUMENT_URI	URL файла
DATE_LOCAL	Текущая дата и время (локаль сервера)
DATE_GMT	Текущая дата и время по Гринвичу
LAST_MODIFIED	Дата и время последнего изменения текущего файла
REMOTE_ADDR	IP адрес удаленного клиента
QUERY_STRING	Строка, полученная от клиента
SERVER_SOFTWARE	Имя HTTP server software
SERVER_NAME	Имя компьютера, на котором работает WWW сервер
REMOTE_HOST	Имя компьютера удаленного клиента
HTTP_USER_AGENT	Имя браузера клиента (browser software).
HTTP_REFERER	URL адрес HTML документа из которого сделан запрос клиентом

include

```
<!--#include file="путь"-->
<!--#include virtual="путь"-->
```

Предназначена для вставки содержимого файла в текущий документ, также может использоваться для вывода результатов работы скрипта.

Аргументы:

- **file** - Путь к файлу, относительно текущей директории. (То есть не может быть абсолютным путем к файлу (начинаться с / или содержать ../).
- **virtual** - Виртуальный (фактический) путь к документу на сервере. Он может начинатся с /, но должен быть на том же сервере.

Пример:

```
<!--#include file="docext.html"-->
<!--#include virtual="/counters/spylog.txt"-->

<!--#include virtual="/cgi-bin/counter.pl" -->
```

fsize

<!--#fsize file="путь"-->
<!--#fsize virtual="путь"-->

Выводит размер файла, формат выдачи может быть изменен командой config

Аргументы:

- **file** - Путь к файлу, относительно текущей директории. (То есть не может быть абсолютным путем к файлу (начинаться с / или содержать ../).
- **virtual** - Виртуальный (фактический) путь к документу на сервере. Он может начинатся с /, но должен быть на том же сервере.

Пример:

<!--#fsize virtual="ssi.htm"-->

flastmod

```
<!--#flastmod file="путь"-->
<!--#flastmod virtual="путь"-->
```

Выводит дату/время последнего изменения файла, формат выдачи может быть изменен командой config

Аргументы:

- **file** - Путь к файлу, относительно текущей директории. (То есть не может быть абсолютным путем к файлу (начинаться с / или содержать ../).
- **virtual** - Виртуальный (фактический) путь к документу на сервере. Он может начинатся с /, но должен быть на том же сервере.

Пример:

```
<!--#flastmod virtual="index.html"-->
```

exec

```
<!--#exec cgi="/cgi-bin/counter.pl"-->
```

Вызывает внешние программы, формат выдачи может быть изменен командой config.

Аргументы:

- **cmd** - вызов исполняемых программ в UNIX shell.
- **cgi** - вызов CGI скрипта.

Пример:

<!--#exec cmd="/bin/finger vasjapjatkin@mail.ru"-->
<!--#exec cgi="/cgi-bin/puperscript.cgi"-->

config

Модифицирует формат вывода для других команд.

Аргументы:

errmsg - формат сообщения об ошибке, которое выдается когда обработчик SSI+ обнаруживает ошибки. Для SSI+ также можно использовать **onerr**.

Пример: *<!--#config errmsg="Ошибка обработки SSI запроса"-->*

sizefmt - формат информации об объеме файла. Допустимые значения sizefmt - *bytes* или *addrev*. Они задают округление значения объема файла до ближайшего килобайта.

Пример: *<!--#config sizefmt="addrev"-->*

timefmt - формат значений даты/времени.

Пример:

```
<!--#config timefmt="%D %r"-->
 <!--#flastmod file="ssi.htm"-->
```

Некоторые форматы представления даты и времени

Код	Значение	Пример
%a	День недели в сокращенном виде	Sun
%A	День недели	Sunday
%b	Месяц в сокращенном виде (также %h)	Jan
%B	Месяц	January
%d	Дата	01
%D	Дата в формате %m/%d/%y	06/23/99
%e	Дата	1
%H	Время в часах в 24-часовой системе	13
%I	Время в часах в 12-часовой системе	01
%m	Номер месяца	11
%M	Количество минут	08
%p	AM|PM	AM
%r	Время в формате %I:%M:%p	09:21:13 PM
%S	Количество секунд	09
%T	Время в 24-часовой системе в формате %H:%M:%S	12:22:40
%Y	Год	1996

Применение SSI

Пожалуй, наиболее значимым достоинством SSI является возможность выполнять шаблонную верстку веб-страниц, собирая их из нескольких файлов. Дело в том, что обычно страницы сайта включают некоторую часть одинаковой информации. Вынося такого рода фрагменты html-кода в отдельные файлы, можно затем повторно использовать их, собирая страницы с помощью SSI.

Типичный пример - это вынесение в отдельный файл логотипа страницы и меню навигации ("шапка"), в другой файл - нижнего колонтитула ("подвал" страницы), тогда третья часть такой страницы может содержать различную информацию. Такой подход позволяет ускорить разработку новых сайтов и упростить модернизацию имеющихся.

Пример шаблонной верстки

Задача: Выполнить верстку веб-страниц, включающих логотип сайта, меню навигации, основную часть и счетчик посетителей

Пусть общее оформление сайта должно выглядеть так, как показано на рис. 1. Код разметки одной из страниц, sample.html, приведен в листинге 1.

Рис. 1 Макет оформления

Листинг 1. Код страницы sample.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>My Cool Site</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta http-equiv="content-style-type" content="text/css">

<link href="s7a.css" rel="stylesheet" type="text/css">
</head>
<body>
<div class="logo"><img src="img/s7logo.png"></div>
<div class="menu">
<ul>
CONTENT
<li><a href="">Startpage</a>

<li><a href="">Sometext1</a>
<li><a href="">Sometext2</a>
<li><a href="">Sometext3</a>
<li><a href="">About</a>

</ul>
<div class="counter"><p><img src="img/cnt.gif" alt="TOP100(88x31.gif)"></div>
</div>
<div class="content"><h1>LOREM IPSUM</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin gravida.
Phasellus tincidunt massa vel urna. Proin adipiscing quam vitae odio. Nullam eget tellus
vitae tortor gravida scelerisque. In orci lorem, cursus imperdiet, ultricies non,
hendrerit et, orci. Nulla facilisi.
<h2>Sed dictum</h2>

<p>Ut tincidunt lorem ac lorem. Duis eros tellus, pharetra id, faucibus eu,
dapibus dictum, odio...
<p>Nulla sed nulla a consectetuer adipiscing elit. Proin gravida. Phasellus
tincidunt massa vel urna. Proin adipiscing quam vitae odio.</p>
<h2>Donec velit</h2>
Nullam eget tellus vitae tortor gravida scelerisque. In orci lorem, cursus imperdiet,
ultricies non, hendrerit et, orci. Nulla facilisi. Phasellus tincidunt massa vel urna.
Proin adipiscing quam vitae odio. Nullam velit nisl, laoreet id, condimentum ut,
ultricies id, mauris.</div>
</body>
</html>
```

Общими для всех страниц сайта будут те элементы, которые выделены на рис.1 красным пунктиром. Если выделенные фрагменты кода вынести в отдельные файлы, а в основном файле использовать директивы SSI, можно существенно облегчить написание сайта, добавляя аналогичную структуру в другие страницы

Посмотрите, как будет выглядеть исходный код рассматриваемой страницы (sample.html), после вынесения части кода во внешние файлы и использовании SSI (выделены директивы SSI-включений)

```
<!--#include virtual="s7logo.html"-->

<div class="content">
<h1>LOREM IPSUM</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin gravida.
Phasellus tincidunt massa vel urna. Proin adipiscing quam vitae odio.
Nullam eget tellus vitae tortor gravida scelerisque.
In orci lorem, cursus imperdiet, ultricies non, hendrerit et, orci. Nulla facilisi.

<h2>Sed dictum</h2>
<p>Ut tincidunt lorem ac lorem. Duis eros tellus, pharetra id, faucibus eu,
dapibus dictum, odio...
<p>Nulla sed nulla a consectetuer adipiscing elit. Proin gravida. Phasellus
tincidunt massa vel urna. Proin adipiscing quam vitae odio.</p>
<h2>Donec velit</h2>
Nullam eget tellus vitae tortor gravida scelerisque. In orci lorem, cursus imperdiet,
ultricies non, hendrerit et, orci. Nulla facilisi. Phasellus tincidunt massa vel urna.
Proin adipiscing quam vitae odio. Nullam velit nisl, laoreet id, condimentum ut, ultricies
id, mauris.
</div>

<!--#include virtual="s7menu.html"-->
<!--#include virtual="s7footer.html"-->
```

Код, содержащийся например в файле s7logo.html, включает лишь описание заголовка:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>My Cool Site</title>

<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta http-equiv="content-style-type" content="text/css">
<link href="s7a.css" rel="stylesheet" type="text/css">
</head>
<body>
<div class="logo"><img src="img/s7logo.png"></div>
```

Аналогичным образом разнесены по файлам и прочие общие элементы.

Для удобства позиционирования элементов на странице в этом примере использованы теги с соответствующим стилевым оформлением, фрагмент файла внешней таблицы стилей (s7a.css) приведен в листинге 2.

Листинг 2. Фрагмент файла s7a.css

```
body {
        margin: 0px;
        }
div {
        border : dashed 2px red;
        }
div.logo {
        position : absolute;
        top: 0;
        left: 0;
        right: 0;
        }
div.menu {
        position : absolute;
        top: 70px;
        left: 0px;
        width: 200px;
        }

div.content {
        position : absolute;
        top: 70px;
        left: 210px;
        right : 0px;
        border : dashed 2px blue;
        }
div.counter p {
        text-align : center;
        }
```

ЛР №8. Серверные приложения. Основы языка PHP

Цель работы

Изучить основы языка PHP. Разработать простое серверное приложение на языке PHP.

Задание к работе

Написать скрипт, учитывающий количество кликов по ссылкам на скачивание файлов, записывающий эти данные в файл, а затем выполняющий редирект на скачиваемый файл. Формат записи может быть, например таким:

```
имя_файла;дата/время;ip-адрес клиента;
```

Методические указания

Серверные приложения

Препроцессная обработка на стороне сервера подразумевает вызов программы-интерпретатора, которая обрабатывает запрашиваемый файл скрипта, исполняет его команды. Результат работы интерпретатора передается веб-серверу, который, в свою очередь, возвращает их клиенту (рис. 1)

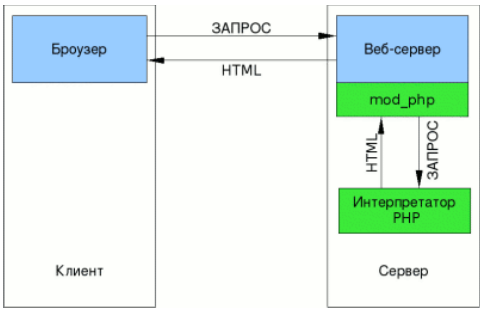


Рис. 1. Препроцессинг HTML на сервере

Основы PHP

PHP (рекурсивный акроним для "PHP: Hypertext Preprocessor") это широко распространённый язык сценариев, который создан специально для Веб и который можно внедрять в HTML.

PHP изначально был разработан с тем, чтобы код выполнялся на сервере. Это позволяет передавать клиенту результат работы скрипта, не показывая ему, каков был исходный код.

PHP предельно прост для новичка в программировании, но предлагает много продвинутых возможностей для программиста-профессионала. Возможности PHP весьма широки, но освоить основные из них несложно за пару часов (конечно же, имея понятие о программировании в целом). Листинг 1 иллюстрирует применение PHP для формирования веб-страницы с текстом «Hello, World!» (курсивом выделен исходный код PHP).

Листинг 1 Простая программа на PHP

```
<?php
        $site_title = "A first PHP program";
        $hello = "Hello, World!";
?>
<html>
<head>
        <title><?PHP echo $site_title; ?></title>

</head>
<body>
<?PHP
        //Вывести текст с меткой времени
        echo "<p style='font-size: 2em;'>I say $hello at ". date("H.i.s")."</p>";
?>
</body></html>
```

На рис. 2 показано, как выглядит приведенный сценарий при выполнении в браузере.

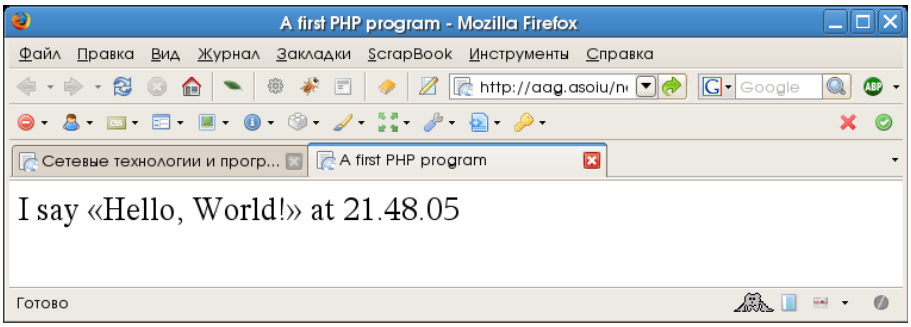


Рис. 2. Результаты выполнения сценария в браузере

Примеры скриптов

```
<?php

// получение информации о настройках веб-сервера и PHP
phpinfo();

//основы синтаксиса и программные единицы
//переменные и вывод

$str = "Hello, world";
echo "<h1>$str</h1>"; // это правильная конструкция. PHP способен разбирать строки и выделять переменные в них

// условный оператор

$min = -100;
$max = 100;
$i = rand($min, $max);
if ($i > 0) {echo "<p>Число положительное</p>";}
else if ($i < 0) {echo "<p>Число отрицательное</p>";}
else {echo "<p>Ноль</p>";}

// цикл for

echo "<h2>Таблица умножения</h2>";
echo "<table>";
for ($i=1; $i<=10; $i++) {
        echo "<tr>";
        for ($j=1; $j<=10; $j++) {
                echo "<td style = 'background-color: silver;width:25px; height:25px; text-align:center; vertical-align: middle;'>".($i*$j)."</td>";
                }
        echo "</tr>";
        }

echo "</table>";

// оператор выбора (скрипт должен быть вызван с параметром uname (e.g.: samples.php?uname=vasya))

echo "<h2>Использование оператора выбора </h2>";
$who = $_GET["uname"];
```

```
$reply = "<b>Liza say:</b> ";

switch ($who) {
        case "kolya" : $reply.="fiii..."; break;
        case "petya" : $reply.="khm..."; break;
        case "vasya" : $reply.="ohhh..yyyyes!"; break;
        default : $reply.= " Who's next?";
}

echo "<p><b>".$who." say:</b> What do U think about Me? <p>".$reply;

// массивы

$fruits = array("banana", "plum", "apple",  "peach");
sort($fruits); // сортировка - см. документацию PHP "Функции массивов"
$out = "";
foreach ($fruits as $f)
        $out .= "<li>$f</li>"; // конкатенация строк
echo "<ul>$out</ul>";

echo "<p>В массиве fruits ".count($fruits)." элем.</p>";
$fruits[5] = "pinapple";

print_r($fruits); // обратите внимание - элемент с индексом 4 в этом примере будет не определен

echo $fruits[4]; // ошибка! этот элемент не определен
$fruits[4] = "";
echo $fruits[4]; // элемент определен, но содержит пустую строку

// многомерные массивы
$vertex[1][0][0] = 1;
$vertex[0][1][0] = 1;
$vertex[0][0][1] = 1;

print_r($vertex);

// ассоциативные массивы
$coords[0]["X"] = 55;
$coords[0]["Y"] = 32;
$coords[1]["X"] = 27;
$coords[1]["Y"] = 0.56;

print_r($coords);

$page["head"] = "<head><title>PHP - it's easy</title></head>";
$page["body"] = "<body><p>A simple sample using associative arrays</p></body>";

print_r($page);

$p = $page["head"].$page["body"];

// файловые операции
$f = "read.me"; // файл в текущей директории
if (file_exists($f)) // проверка существования файла
        $text = file_get_contents($f); // чтение из файла

file_put_contents($f, $p); // запись в файл, директория должна быть доступна для записи (access rights - 777)

// еще о файлах
$fd = fopen($f, "a"); // открытие для дозаписи (здесь "a" - append)
$str = "some text";
fwrite($fd, $str); // запись в конец файла
fclose($fd);

// переменные окружения
echo "<h1>Переменные окружения</h1>";
foreach ($_SERVER as $var=>$val)  {
        echo "<p>$"."_SERVER['$var'] = $val</p>";
        };

// перенаправление запроса
$url = "index.html";
header("Location:$url"); // функция header заголовки http, поэтому должна вызываться раньше любого вывода

// Обработка параметров
echo "<p>Пусть на сервер передан запрос вида
http://myserv.dom/test.php?id=2344&uname=vasya&nick=vasiliok&age=19 (использован метод GET)
<p>Требуется вывести все переменные из строки запроса.";
echo "<ol><li>Использование функции печати массива: <br>";
print_r($_GET);

foreach ($_GET as $key => $val){
   echo "<li>parameter: <b>".$key."</b> value: <b>".$val."</b>";
}

echo "</ul>\n<li>Поэлементный вывод (выводим только значения)<ul>";
   echo "<li>ID: <b>".$_GET["id"]."</b>";
   echo "<li>Firstname: <b>".$_GET["uname"]."</b>";
   echo "<li>Nickname: <b>".$_GET["nick"]."</b>";
   echo "<li>Age: <b>".$_GET["age"]."</b>";

echo "</ul></ol>";

echo "<p>Проверка, что переданы нужные параметры:";
if (isset($_GET["id"])) {
        echo "<p><b>do something...</b>";
} else {
        echo "<p><b>nothing to do ...</b>";
};

echo "<h2>Проверка, что передан нужный параметр и требуемое значение</h2>";
```

```
if ((isset($_GET["id"]))&&($_GET["id"] == 2344)){
        echo "<p><b>do something...</b>";
} else {
        echo "<p><b>nothing to do ...</b>";
}

?>
```

В качестве примечания: Как видно из примеров, PHP очень тесно интегрируется с гипертекстом. Такой стиль кодирования называют "спагетти", поскольку сложно отделить содержание документа от его оформления, что напоминает слипшиеся макароны. На самом же деле, для профессиональных разработчиков PHP предлагает возможности разделения кода и данных как с помощью объектной парадигмы программирования, так и с помощью шаблонов, но рассмотрение этих вопросов выходит за рамки лабораторного практикума.

Ресурсы по PHP

Объем лабораторной работы не позволяет подробно рассматривать возможности языка (это справедливо не только для PHP), поэтому для экономии места укажем некоторые электронные ресурсы по PHP, а ниже разберем некоторые примеры скриптов.

- PHP, MySQL и другие веб-технологии
- phpclub.ru - портал для разработчиков на PHP

ЛР 9. Библиотечные функции PHP

Цель работы

Использование библиотечных функций обработки текста, HTML, массивов и пр.

Задания к работе

Написать скрипт, читающий файл статистики скачиваний (ЛР 8) и подсчитывающий количество скачиваний по каждому файлу. Полученную информацию внедрить в страницы вашего сайта возле ссылок на соответствующие файлы (программа, руководство пользователя и прочие файлы в форматах pdf, doc, zip и т.п.).

Указания к работе

Гибкость языка PHP во многом обеспечивается разнообразными библиотечными функциями. Основные библиотеки включены в типовую поставку пакета PHP и всегда доступны, т.е. не требуется подключать модули библиотек и пересобирать PHP. Подробное описание доступных библиотек приведено в руководстве программиста. Здесь приведем описание некоторых библиотечных функций, которые могут понадобиться для выполнения задания лабораторной работы. Прежде чем перейти к этому описанию приведем задачу к алгоритмическому представлению (на уровне описания шагов).

1. Открыть файл статистики скачиваний
2. Пока не достигнут конец файла
    1. Прочитать запись из файла
    2. Выделить из записи имя файла
    3. Для этого файла увеличить значение счетчика скачиваний
3. Закрыть файл
4. Вывести значение счетчика

По близкой к приведенному алгоритму схеме можно подсчитать количество уникальных скачиваний. Для этого нужно добавить обработку ip-адресов, исключая повторы.

Итак, функции, которые можно применить для реализации приведенного алгоритма.

Файловые операции

fopen

fopen -- Открывает файл или URL и возвращает дескриптор открытого файла.

Описание

```
resource fopen ( string filename, string mode [, bool use_include_path [, resource zcontext]] )
```

Параметр mode указывает тип доступа, который вы запрашиваете у потока. Он может быть одним из следующих:

Список возможных режимов для fopen() используя mode

```
mode    Описание
'r'     Открывает файл только для чтения; помещает указатель в начало файла.
'r+'    Окрывает файл для чтения и записи; помещяет указатель в начало файла.
'w'     Открывает файл только для записи; помещает указатель в начало файла и обрезает файл до нулевой длинны. Если файл не существует - пробует его создать
'w+'    Открывает файл для чтения и записи; помещает указатель в начало файла и обрезает файл до нулевой длинны. Если файл не существует - пробует его созда
'a'     Открывает файл только для записи; помещает указатель в конец файла. Если файл не существует - пытается его создать.
'a+'    Открывает файл для чтения и записи; помещает указатель в конец файла. Если файл не существует - пытается его создать.
'x'     Создаёт и открывает только для записи; помещает указатель в начало файла. Если файл уже существует, вызов fopen() закончится неудачей, вернёт FALSE
'x+'    Создаёт и открывает для чтения и записи; помещает указатель в начало файла. Если файл уже существует, вызов fopen() закончится неудачей, вернёт FALS
'b'     Открывает файл в двоичном режиме.
```

Необязательный третий параметр use_include_path может быть установлен в '1' или TRUE, если вы также хотите провести поиск файла в include_path.

Если открыть файл не удалось, функция вернёт FALSE и сгенерирует ошибку уровня E_WARNING. Вы можете использовать @ для того, чтобы подавить это предупреждение.

Листинг 1. Примеры использования функции fopen()

```
<?php
$handle = fopen("/home/rasmus/file.txt", "r");
$handle = fopen("/home/rasmus/file.gif", "wb");
$handle = fopen("http://www.example.com/", "r");
$handle = fopen("ftp://user:password@example.com/somefile.txt", "w");
?>
```

fgets

fgets -- Читает строку из файла

Описание

```
string fgets ( resource handle [, int length] )
```

Возвращает строку размером в length - 1 байт, прочитанную из дескриптора файла, на который указывает параметр handle. Чтение заканчивается, когда количество прочитанных байтов достигает length - 1, по достижении конца строки (который включается в возвращаемое значение) или по достижении конца файла (что бы ни встретилось первым). Если длина не указана, по умолчанию ее значение равно 1 килобайту или 1024 байтам.

Указатель на файл должен быть корректным и указывать на файл, успешно открытый функциями fopen() или fsockopen().

В случае возникновения ошибки функция возвращает FALSE.

Листинг 2. Построчное чтение файла

```
<?php
$handle = fopen("/tmp/inputfile.txt", "r");
while (!feof($handle)) {
    $buffer = fgets($handle, 4096);
    echo $buffer;
}
fclose($handle);
?>
```

feof

feof -- Проверяет, достигнут ли конец файла

Описание

```
bool feof ( resource handle )
```

Функция возвращает TRUE в случае, когда дескриптор указывает на достижение конца файла или же если произошла ошибка (включая таймаут сокета), иначе возвращает FALSE (см. листинг 2).

fclose

fclose -- Закрывает дескриптор файла

Описание

```
bool fclose ( resource handle )
```

Функция закрывает файл, на который указывает handle. Дескриптор должен указывать на файл, открытый ранее с помощью функции fopen() или fsockopen().

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки (см. листинг 2).

file_get_contents

file_get_contents -- Получить содержимое файла в виде одной строки

Описание

```
string file_get_contents ( string filename [, bool use_include_path [, resource context [, int offset [, int maxlen]]]] )
```

Содержимое файла возвращается в строке, начиная с указанного смещения offset и до maxlen байтов. В случае неудачи, file_get_contents() вернёт FALSE.

Использование функции file_get_contents() наиболее предпочтительно в случае необходимости получить содержимое файла целиком.

Функции обработки строк

```
explode

(PHP 3, PHP 4, PHP 5)
explode -- Разбивает строку на подстроки
Описание
array explode ( string separator, string string [, int limit] )

Возвращает массив строк, полученных разбиением строки string с использованием separator в качестве разделителя. Если передан аргумент limit, массив будет со

Если separator - пустая строка (""), explode() возвращает FALSE. Если separator не содержится в string, то explode() возвращает массив, содержащий один элем
```

По историческим причинам, функции implode() можно передавать аргументы в любом порядке, но для explode() это недопустимо. separator всегда должен содержать

    Замечание: Аргумент limit был добавлен в PHP 4.0.1

Пример 1. Примеры применения explode()
```
<?php
// Пример 1
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2

// Пример 2
$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user, $pass, $uid, $gid, $gecos, $home, $shell) = explode(":", $data);
echo $user; // foo
echo $pass; // *

?>
```

strstr

(PHP 3, PHP 4, PHP 5)
strstr -- Находит первое вхождение подстроки
Описание
string strstr (string haystack, string needle)

Возвращает подстроку строки haystack начиная с первого вхождения needle до конца строки.

Если подстрока needle не найдена, возвращает FALSE.

Если needle не является строкой, он приводится к целому и трактуется как код символа.

    Замечание: Эта функция учитывает регистр символов. Для поиска без учета регистра используйте stristr().

Пример 1. Пример использования strstr()
```
<?php
$email = 'user@example.com';
$domain = strstr($email, '@');
echo $domain; // выводит @example.com
?>
```

    Замечание: Если нужно лишь определить, встречается ли подстрока needle в haystack, используйте функцию strpos(), которая работает быстрее и потребляет м

С версии PHP 4.3.0 strstr() безопасна для обработки данных в двоичной форме.

См. также описание функций ereg(), preg_match(), stristr(), strpos(), strrchr() и substr().

substr

(PHP 3, PHP 4, PHP 5)
substr -- Возвращает подстроку
Описание
string substr (string string, int start [, int length])

substr() возвращает подстроку строки string длиной length, начинающегося с start символа по счету.

Если start неотрицателен, возвращаемая подстрока начинается в позиции start от начала строки, считая от нуля. Например, в строке 'abcdef', в позиции 0 наход

Пример 1. Пример использования substr()
```
<?php
$rest = substr("abcdef", 1);    // возвращает "bcdef"
$rest = substr("abcdef", 1, 3); // возвращает "bcd"
$rest = substr("abcdef", 0, 4); // возвращает "abcd"
$rest = substr("abcdef", 0, 8); // возвращает "abcdef"

// к отдельным символам можно обращаться с помощью фигурных скобок
$string = 'abcdef';
echo $string{0};                // выводит a
echo $string{3};                // выводит d
?>
```

Если start отрицательный, возвращаемая подстрока начинается с start символа с конца строки string.

Пример 2. Использование отрицательного start
```
<?php
$rest = substr("abcdef", -1);    // возвращает "f"
$rest = substr("abcdef", -2);    // возвращает "ef"
$rest = substr("abcdef", -3, 1); // возвращает "d"
?>
```

Если length положительный, возвращаемая строка будет не длиннее length символов. Если длина строки string меньше или равна start символов, возвращается FALS

Если length отрицательный, то будет отброшено указанное этим аргументом число символов с конца строки string. Если при этом позиция начала подстроки, опреде

Пример 3. Использование отрицательного length
```
<?php
$rest = substr("abcdef", 0, -1);  // возвращает "abcde"
$rest = substr("abcdef", 2, -1);  // возвращает "cde"
$rest = substr("abcdef", 4, -4);  // возвращает ""
$rest = substr("abcdef", -3, -1); // возвращает "de"
?>
```

См. также описание функций strrchr(), substr_replace(), ereg(), trim() и mb_substr().

Многобайтовые строки

PHP поддерживает работу с многобайтовыми кодировками (в частности, UTF-8). Для них имеется отдельная библиотека mbstring. В этой библиотеке имеются аналоги обычных строковых функций, адаптированные для многобайтовых строк.

Функции обработки регулярных выражений

Весьма полезными при обработке строк являются функции для работы с регулярными выражениями. PHP поддерживает POSIX- и PERL-совместимый синтаксис синтаксис regex. Для этого используются библиотеки "Regular Expression Functions (POSIX Extended)" и "Функции для работы с регулярными выражениями (Perl-совместимые)"

Рекомендации к выполнению задания

Поставленная задача подсчета скачиваний с помощью php-скрипта может быть решена различными способами. Так, возможно построчное чтение из файла, как это описано выше. Другой способ, более удобный, сводится к считыванию файла целиком и преобразованию содержимого в массив путем разбиения на подстроки. Затем полученный массив можно анализировать с помощью библиотечных функций для работы с массивами.

ЛР 10. Веб-формы

Задание:

ЛР 11. Взаимодействие с БД

Задание:

ЛР 12. Сессии. Ограничение доступа к содержимому веб-страниц

Задание:

ЛР 13. Использование .htaccess

Задание:

ЛР 14. Спецификация SiteMap

Задание:

ЛР 15. Формат RSS

Задание:

ЛР 16. Протокол WAP

Задание:

ЛР 17. Отладка сайта. Размещение сайта на веб-сервере