Основы Web - технологий

Часть 2. Языки разметок и стилей: HTML

- Общее представление об HTML - документе
- Классификация тегов
- Основные атрибуты
- Теги разметок и фреймов
- Мета – определения
- Вспомогательные файлы robot и favicon

Особенности работы с HTML - документами

********** ОСОБЕННОСТИ ОТОБРАЖЕНИЯ

- Браузеры не информируют об ошибке, поэтому неверная запись тега ими просто игнорируется. При ошибке – каждый браузер интерпретирует код так, как он его понимает
- Если не применять специальных мер, введенный текст, а также рисунки, отображаются в одну строку с переносом слов по ширине экрана и с удалением присутствующих в исходном тексте управляющих (невидимых) символов перевода строк, лишних пробелов и знаков табуляции.
- вид отображения задается не только тегами, но и типом используемого браузера, его настройками (безопасность, отображение графики, разрешение скриптов и т.д.), версией спецификации HTML, шириной окна, общей настройкой ОС. Плюс – таблицы стилей

********** ОСОБЕННОСТИ СИНТАКСИСА

- В новых стандартах все служебные слова записываются строчными буквами.
- Ссылка на файлы – как в DOS: относительная
- Независимо от расширения файла (htm, html, php, asp и др.-см.ниже), HTML – в основе верстки страницы и ее информационного заполнения

********** ФАЙЛОВАЯ СТРУКТУРА WEB - ПРИЛОЖЕНИЯ

- Исходный файл по умолчанию – index с любым допустимым расширением
- Специализированные служебные файлы: robot.txt (где не искать), favicon.ico (значок закладки), .htaccess (указание директив серверу http://httpd.apache.org/docs/current/mod/directives.html)
- Файлы, определяющие сайт: *.htm, *.html, *.php, *.css, *.js + *.pl, *.asp, *.aspx, а также мультимедийные файлы (звук, изображение, видео) и любые другие файлы данных, предполагающие программный обмен информацией с ними.
- Файлы – обработчики убираются в каталог cgi-bin или подобный. Под изображения, скрипты JS, файлы для скачивания и загрузки также выделяют специальные каталоги

HTML: общая структура документа


```
<!-- ПРОЛОГ: Задает основной стандарт документа: strict, transitional, frameset -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- HTML– корневой тег-оболочка. Атрибуты могут задавать ряд элементов
оформления: фон, шрифт и др. -->
<html xmlns="http://www.w3.org/1999/xhtml" >
  <!– HEAD - ЗАГОЛОВОК СТРАНИЦЫ:для описания служебной информации -->
  <head id="ctl00_ctl00_Head1">
    <!-- TITLE: название страницы для закладок браузера-->
    <title> Intellicast - Barnaul Weather Report in Russia  </title>
    <!– META-блок: сведения информационного и служебного характера для управления страницей -->
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <!– SCRIPT-блок: программы и функции для реализации бизнес - логики-->
    <script type="text/javascript"> /*текст JS*/ </script>
   <!– подключение внешних модулей:  стилей, скриптов и проч.-->
    <link href="http://img.intellicast.com/Styles/St.css" rel="stylesheet" type="text/css" />
    <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />
    <script type="text/javascript" src="http://ajax.g... /jquery.min.js"> </script>
    <!--[if IE 6]>
      <link href="http://images.intellicast.com/Styles/IE6_20091102.css"
      rel="stylesheet" type="text/css" />
    <![endif]-->
  </head>
  <!-- ТЕЛО СТРАНИЦЫ BODY – ее основное содержание. Может быть несколько, но активно – только одно-->
  <body id="ctl00_ctl00_MasterBody">
     <div id="header" style="width:auto;height:60px;…">
     …
  <body/>
<html>
```

HTML: Режимы отображения и типы документов

"the Standards mode" и "the Quirks mode"- стандартный и неопределенный режимы отображения браузеров

Рекомендовано применять:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang=«ru">
 <head> <title>An XHTML 1.0 Strict standard template</title>
  <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
 </head>
 <body> <p>… Your HTML content here …</p> …
 </body>
</html>
```

<!DOCTYPE [Элемент верхнего уровня] [Публичность] "[Регистрация]//[Организация]//[Тип] [Имя]//[Язык]" "[URL]">

- Элемент верхнего уровня — указывает элемент верхнего уровня в документе, для HTML это тег <html>.
- Публичность — объект является публичным (значение PUBLIC) или системным ресурсом (значение SYSTEM), например, таким как локальный файл. Для HTML/XHTML указывается значение PUBLIC.
- Регистрация — сообщает, что разработчик DTD зарегистрирован в международной организации по стандартизации (International Organization for Standardization, ISO). Принимает одно из двух значений: плюс (+) — разработчик зарегистрирован в ISO и - (минус) — разработчик не зарегистрирован. Для W3C значение ставится «-».
- Организация — уникальное название организации, разработавшей DTD. Официально HTML/XHTML публикует W3C, это название и пишется в <!DOCTYPE> (`//W3C//`)
- Тип — тип описываемого документа. Для HTML/XHTML значение указывается DTD.
- Имя — уникальное имя документа для описания DTD (`"XHTML 1.0 Strict//`)
- Язык — язык, на котором написан текст для описания объекта. Содержит две буквы, пишется в верхнем регистре. Для документа HTML/XHTML указывается английский язык (EN).
- URL — адрес документа с DTD (`"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"`).

HTML: основные типы документов

ОСНОВНЫЕ ТИПЫ DTD :

(HTML 4.01 , XHTML 1.0+Strict, Transitional, Frameset),XHTML 1.1 и HTML 5

1. **Строгий тип (Strict HTML 4.01 и XHTML 1.0). Не содержит элементов, помеченных как «устаревшие» или «не одобряемые» (deprecated)**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

2. **Переходный (Transitional HTML 4.01 и XHTML 1.0): содержит устаревшие теги в целях совместимости и упрощения перехода со старых версий HTML**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN«
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

3. **С фреймами (Frameset HTML 4.01 и XHTML 1.0): аналогичен переходному, но содержит также теги для создания наборов фреймов.**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

4. **XHTML 1.0 – DTD**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

5. **HTML 5.0 – DTD**

```
<!DOCTYPE html>
```

Информация взята с http://htmlbook.ru/html/!doctype

HTML: классификация тегов

Классификация (достаточно условная) и краткий обзор тегов:

Теги – контейнеры логической структуры документа: html, head, body/фрейм,

Теги – контейнеры спец. неотображаемых и информационных объектов: стилей, информации о странице, указатели действий, звук, программ и проч.: style, script, meta, …

Теги – контейнеры отображаемых элементов: таблица, текстовые блоки p, div. Суть – блочные элементы. Остальные элементы - последовательные

Теги – собственно элементы: текст, линия, изображение, элементы форм

Теги – форматы элементов, когда помещаемый в них текст приобретает определенные свойства: шрифт, цвет, фон и проч.

1. **Документ, элементы структуры, оформления и функционирования** `(html, head, meta, body, style, title, script)`
   - **Фреймы** `(frameset,frame,iframe,noframes,banner)`
2. **Основные элементы контента:**
   - **Блочные элементы** `(p – абзац, div - блок, hr, h1…h6, marquee, center)`
   - **Таблица** `(table, caption, thead, th, colgroup, col, tbody, tr, td, tfoot)`
   - **Списки** `(ul, ol, li, dl, dt, dd, menu)`
   - **Ссылки** `(a, base, link)`
   - **Формы** `(form, fieldset, legend, textarea, button, input, label, optgroup, option, select)`
3. **Мультимедийные элементы:**
   - **Объекты** `(object, embed, noembed, param)`
   - **Звук** `(bgsound, sound)`
   - **Изображения** `(img, map, area)`
4. **Форматирование текста:**
   - **Физическое** `(br-нов. строка, blink, font, b, strong, i, strike, tt, u, sub, sup big, small, pre, basefont)`
   - **логическое** `(acronym, abbr, address, bdo, blockquote, cite, code, comment, del, dfn, em, ins, kbd, nobr, q, samp, var) span,`

HTML: теги документа, фреймов и текстовые контейнеры

1	< !doctype >	Задает тип документа, используемый в нем стандарт
2	< html >	заключает в себе все содержимое веб-страницы
3	< head >	для хранения вспомогательных элементов, цель которых — помочь браузеру в работе с данными
4	< meta >	определяет метатеги - информация для поисковиков, браузеров
5	< body >	предназначен для хранения содержания веб-страницы
6	< style >	для определения стилей элементов веб-страницы
7	< title >	Определяет заголовок документа.

1	< frameset >	Определяет структуру фреймов на веб-странице
2	< frame >	определяет свойства отдельного фрейма
3	< iframe >	создает плавающий фрейм
4	< noframes>	Для отображения в браузере, когда он не поддерживает фреймы
5	< banner >	Фиксирует неподвижную, не зависящую от прокрутки область на экране (Не все браузеры понимают)

1	< p >	Абзац, текстовый параграф
2	< div >	Блок – контейнер
3	< h1> .. <h6>	Заголовки: 1 – самый крупный, 6 – самый мелкий
4	< marquee>	Бегущая строка текста или иных элементов
5	< center >	горизонтальное выравнивание ВСЕХ элементов (таблиц etc) посередине окна
6	< span >	для определения стилей части информации внутри других тегов документа

Примечание: цветом выделены наиболее употребительные теги

HTML: теги таблиц, списков и элементов оформления

1	< table >	Контейнер таблицы
2	< caption >	Задает заголовок, появляющийся сразу перед таблицей. 1-й тег внутри Table
3	< thead >	Контейнер верхнего колонитула (верхних строк) таблицы
4	< th >	Создает заголовочную ячейку таблицы
5	< colgroup >	Задает параметры группы колонок таблицы
6	< col >	Задает параметры колонок таблицы
7	< tbody >	Контейнер одной или нескольких строк (при использовании стилей)
8	< tr >	Контейнер строки таблицы. Вкладывается в контейнер колонтитулов/таблицы
9	< td >	Контейнер ячейки таблицы. Вкладывается в контейнер строк
10	< tfoot >	Контейнер нижнего колонитула (нижних строк) таблицы
	Списки	
8	< ul >	Контейнер маркированного списка (nordering text)
6	< ol >	Контейнер нумерованного списка
4	< li >	определяет отдельный элемент маркированного/нумерованного списка
2	< dl >	Список терминов и их определений: контейнер (Definition List)
3	< dt >	Список определений: название термина (Definition Title)
1	< dd >	Список определений: описание термина (Definition Define)
5	< menu >	устаревший - создание ненумерованного списка
	Элементы оформления	
1	< br >	перевод строки
2	< hr >	Рисует горизонтальную линию

HTML: теги ссылок и форм

	Ссылки	
1	< a >	устанавливает ссылку или якорь
2	< base >	в HEAD задает базовый адрес для относительных ссылок
3	< link >	Уст.связь с внешн.док-м/файлом со стилями,шрифтами,иконкой. Всегда внутри <HEAD>
	< area >	См. теги изображений
	Формы	
3	< form >	устанавливает форму на веб-странице. Основной способ передачи выводимых пользователем данных на сервер. Нужна для группировки передаваемых данных
2	< fieldset >	для группирования элементов формы
6	< legend >	создает заголовок группы элементов формы, которая определяется тегом FIELDSET
9	< textarea >	Элемент формы, для создания области, в которую можно вводить строки текста
1	< button >	кнопки различного типа
4	< input >	позволяет создавать разные элементы интерфейса: кнопки, поля ввода и прочее
5	< label >	задает связь между определенной меткой - текстом и элементом формы INPUT
7	< optgroup >	контейнер, внутри которого располагаются теги OPTION (пункты списков)
8	< option >	отдельные пункты списка, создаваемого с помощью контейнера SELECT
7	< select >	список формы с одним или множественным выбором, раскрывающийся список

HTML: теги мультимедиа и скриптов

Изображения	
< img >	отображение графических файлов
< map >	служит контейнером для элементов AREA
< area >	определяет активные области изображения в MAP, которые являются ссылками
Объекты (типа Flash Player, Video)	
< object >	Как и чем загружать и отображать инородные объекты, котор.Браузер не понимает
< embed >	Устаревший-use Object. Embed - внутрь для совместимости
< noembed>	текст сообщения, если браузер не понимает тег embed
< param >	Параметр объекта, например <param name="movie" value="8.swf">
Скрипты	
< applet >	Ссылка на файлы апплетов
< noscript >	Контейнер.Показывает содержимое, если скрипты отключены/не поддерживаются
< script >	Контейнер для описания скриптов. Может содержать ссылку на ПО/ее текст
Звук	
< bgsound >	Звуковой файл и его громкость
< sound >	нет пояснений
	текст заголовков 1-6 го уровней (Макс-мин)

HTML: теги физического форматирования

1	< blink >	Мигание (поддерживается не всеми браузерами). Не рекомендуется к применению
3	< font >	для изменения характеристик шрифта, таких как размер, цвет и гарнитуры
4	< b >	Тег B создает жирный текст. Закрывающий тег обязателен
5	< strong >	для акцентирования текста жирным
6	< i >	Устанавливает курсивное начертание шрифта
7	< strike >	отображает текст как перечеркнутый
8	< tt >	Отображает текст моноширинным текстом
9	< u >	Подчерк
10	< sub >	шрифт в виде нижнего индекса
11	< sup >	шрифт в виде верхнего индекса
12	< big >	увеличивает размер шрифта на единицу
13	< small >	уменьшает размер шрифта на единицу
14	< pre >	определяет блок предварительно форматированного текста (моноширинный, со всеми пробелами)
15	< basefont >	Шрифт п.у.

HTML: теги логического форматирования

1	**< acronym >**	Стиль общепринятых сокращений, терминов (акроним) типа DOS, ликбез и т.п.
2	**< abbr >**	Аббревиатура
3	**< address >**	Автор/адрес со ссылками
4	**< bdo >**	направление вывода текста (для иностр.алфавитов)
5	**<blockquote>**	для выделения длинных цитат
6	**< cite >**	помечает текст как цитату или сноску на другой материал
7	**< code >**	текст программы
8	**< comment>**	Для комментариев
9	**< del >**	текст, что был удален в новой версии документа
10	**< dfn >**	для выделения терминов при их первом появлении
11	**< em >**	для акцентирования текста курсивом
12	**< ins >**	для выделения текста, который был добавлен в новую версию
13	**< kbd >**	текста, который набирается на клавиатуре или для названия клавиш
14	**< nobr >**	Уведомляет браузер отображать текст без переносов
15	**< q >**	для выделения в тексте цитат (ставит кавычки)
16	**< samp >**	результат вывода компьютерной программы
17	**< var >**	для выделения переменных компьютерных программ (курсив)

Позволяют логически структурировать текст и при необходимости отформатировать ег должным образом

HTML5: Новые теги HTML5

<article> </article>	Внешний контент
<aside> </aside>	дополнительный контент
<audio> </audio>	определяют фоновый звук
<command> </command>	добавляют команду к кнопке
<datalist> </datalist>	определяют допустимые значения
<details> </details>	определяют детали документа
<dialog> </dialog>	определяют диалог
<embed />	внедряет интерактивный объект
<footer> </footer>	нижняя часть документа
<header> </header>	верхняя секция документа
<hgroup> </hgroup>	определяют группу заголовков
<mark> </mark>	определяют важную часть текста
<nav> </nav>	определяют группу ссылок
<section> </section>	определяют секцию документа
<time> </time>	определяют дату или время
<video> </video>	внедряют видео в web-страницу

http://ab-w.net/HTML5/html5.php - выделены новые и устаревшие теги

HTML: основные атрибуты

Спец. обозначения и единицы измерения:

- **Цвета –** словом или числом в шестнадцатеричном коде. Обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних — зеленую, а два последних — синюю.
- **размеры** (символы, пиксели – п.у., %)
- **специальные символы**, типа авторских знаков, пробелы – начинаются со знака «&». Отображаются только при использовании специальной системы обозначений (след.слайд)

Типовые атрибуты:

Цвет: *color="*цвет*", например, bordercolor, bgcolor

Выравнивание: align="left, right, center, justify, top, middle, bottom"

Размеры: size, width, height

Подсказка: title

Обращения к источникам данных, в том числе в ссылках :

- ссылки в пределах текущей страницы: *#метка*.
- ссылки на файлы в текущем и других каталогах: как в DOS. Ссылка на файл в том же каталоге – только имя, в нижележащем – путь, начиная с текущего, в вышележащих – запись вида «../» означает на каталог вверх. Если это не ссылка на другую страницу, то появится диалог: открыть/сохранить/отменить
- ссылки на URL, типа http:// - инициирует либо переход, либо открытие нового окна.
- ссылка на адрес электронной почты: mailto: *адрес* [?subject=*тема сообщения*].

Таблица обозначений цветов

Black	000000=черный
Silver	C0C0C0=Серебряный
Gray	808080=Серый
White	FFFFFF=Белый
maroon	800000=Темно-бардовый
red	FF0000=красный
green	008000=зеленый
Lime	00FF00=Известь, лайм
olive	808000=Оливковый
yellow	FFFF00=Желтый
navy	000080=Темно-синий
blue	0000FF=синий
purple	8000080=фиолетовый
fuchsia	FF00FF=фуксия
teal	008080=чирок
aqva	00FFFF=аква

HTML: обозначения специальных символов

&Copy или &#169	©
&reg или &#174	®
&#8482 или &#	™
&sect или &#167	§
&micro или &#181	µ
& или &#	π
&para или &#	¶
&laquo или &#171	« (\00ab)
» или &#187	» (\00bb)
<	<
>	>
"	"
&атр;	&
	Неразр.""
"	" (двойная кавычка, \0022)
'	(апостроф, \0027)
‘	‘ (откр.кавычка, \2018)
’	’ (закр.кавычка, \2019)
“	“ (откр.двойная кавычка, \201c)
”	” (закр.двойная кавычка, \201d)
„	„ (откр.двойная кавычка в русском, \201e)

HTML: тег разметки (ссылок и переходов)

Примеры:

```
<a href="images/xxx.jpg">Посмотрите на мою фотографию!</a> <!- показать изображение -->
<a href=«downloads/xxx.zip">Скачайте файл</a>          <!- открыть/скачать файл -->
<a href="tip.html">Как сделать такое же фото?</a>      <!- переход на новую страницу-->
<a href="../wormik/knob.html">Относительная ссылка</a> <!- переход на новую страницу-->
<a href="http://www.htmlb.ru/work/knob.html">Абс.ссылка</a> <!- переход на другой сайт -->
... <p><a name="top"> создаем место для перехода</a></p>    <!- поставить метку (якорь) -->
...
<p><a href="#top">Наверх (ссылка на место, куда переходим)</a></p> <!- переход на якорь-->
```

Тег <a>, в зависимости от наличия параметров name или href, ***устанавливает ссылку или якорь*** (закладка внутри страницы).

Параметры

href="ссылка" - Задает адрес перехода. Адрес ссылки в параметре href может быть внутренним, абсолютным и относительным. В качестве адреса может быть файл любого типа, и тогда этот файл откроется или появится диалог для запуска/сохранения. Ссылка на якорь имеет вид #имя якоря.

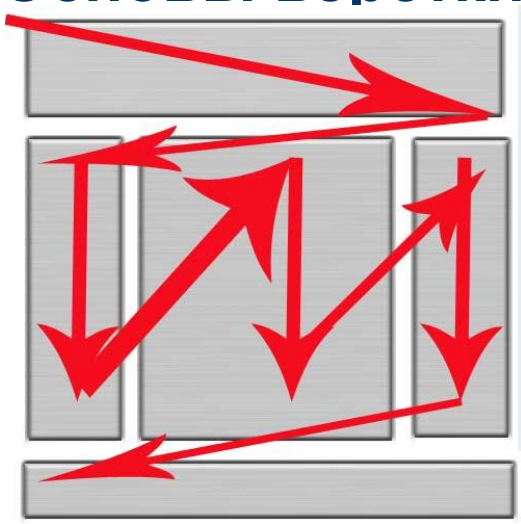
name - Устанавливает имя якоря внутри документа. Можно делать ссылку на закладку, находящуюся на другой веб-странице и даже другом сайте. Для этого в адресе ссылки надлежит указать ее адрес и в конце добавить символ решетки # и имя якоря с учетом регистра.

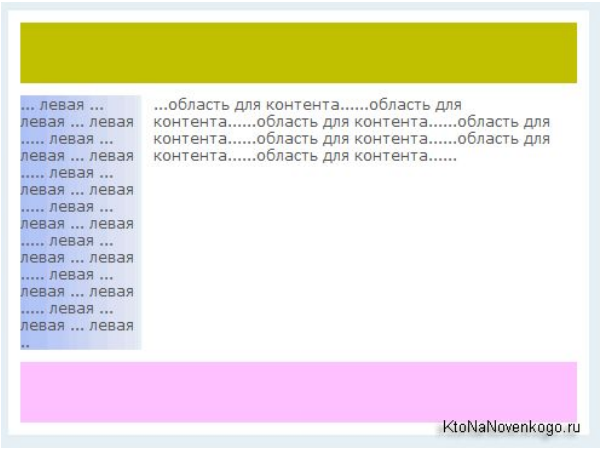
target - Имя окна или фрейма, куда браузер будет загружать документ. П.У. документ открывается в текущем окне или фрейме. В XHTML применение этого параметра запрещено. В качестве аргумента используется имя окна или фрейма, заданное параметром name. Если установлено несуществующее имя, то будет открыто новое окно. В качестве зарезервированных имен используются следующие:

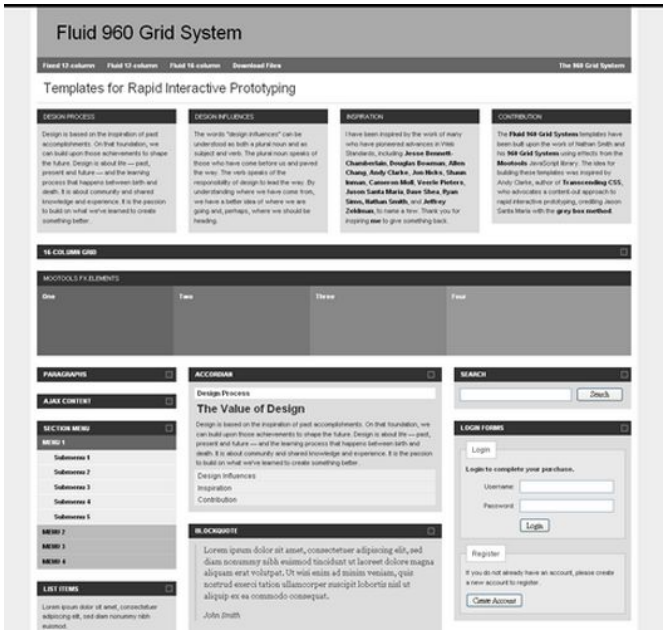
- _blank - Загружает страницу в новое окно браузера
- _self - Загружает страницу в текущее окно (ПУ)
- _parent - Загружает страницу во фрейм-родитель, если фреймов нет, то этот параметр работает как _self.
- _top - Отменяет все фреймы и загружает страницу в полном окне браузера. Если фреймов нет, аналогичен _self.

title - Добавляет всплывающую подсказку к тексту ссылки.

Основы верстки сайта







По степени заполнения экрана верстки бывают:

1. Фиксированные
2. Эластичные (резиновые)

По структуре информации на странице:

одно – двух – трех колоночные, сложные

По используемым тегам для компоновки страницы:

1. Табличные
2. Блочные
3. Фреймы

Рекомендации по верстке

1. Следует учитывать соотношение сторон экрана
2. Меню размещать с учетом соотношения сторон и возможности скрытия ненужных пунктов
3. Верхняя заставка не на главной странице может быть по высоте меньше, чем на главной - для увеличения размера информационного поля. В то же время стиль и оформление должны сохраняться на всех страницах
4. Не стоит дублировать пункты меню по разным сторонам страницы – это сбивает с толку
5. Без крайней необходимости не стоит размещать на странице информации столько, чтобы требовалась ее прокрутка на большинстве современных мониторов и мобильных устройствах. Если не выходит – добавляем содержание в начале и возврат к содержанию после просмотра фрагмента
6. Нужно всегда предоставлять пользователю информацию о его месте нахождения на сайте (подсвечивать пункты, либо выносить путь от вершины в отельную строку)

03.05.19 18:57

Основы Web - технологий

Часть 4. SSI. Скриптовый язык PHP и СУБД MySQL

- PHP. Общее представление о языке
- Основы синтаксиса PHP
- Прикладные задачи PHP

Примечание: далее наиболее важная для выполнения расчетного задания информация подеркнута

PHP: Общее представление

PHP– *Personal Home Page Tools.* скриптовый язык программирования общего назначения. Поддерживается подавляющим большинством хостинг-провайдеров. Является одним из лидеров среди языков программирования, применяющихся для создания динамических веб-сайтов. Распространяется под собственной лицензией, несовместимой с GNU GPL.

Скри́птовый язык, или язык сценариев — язык программирования, разработанный для записи и воспроизведения последовательности операций, которые пользователь может выполнять на компьютере, либо же для воспроизведения заранее написанных сценариев

Сценарий - программа, которая автоматизирует задачи, которые иначе пользователь делал бы вручную. Применительно к web - технологиям сценарий автоматизирует процесс генерации HTML – кода страниц сайта.

Помимо PHP, к скриптовым языкам из наиболее известных и популярных относятся:

- на стороне сервера: Perl, Python, Ruby, JS.node. Сегодня Google предлагает еще Go
- на стороне клиента: JavaScript, ActionScript, Visual Basic Script

На сегодня актуальна версия 5.3 -6.5. Синтаксис Си-подобный. Содержит ядро и подключаемые библиотеки, или модули.

Ядро - позволяет выполнять все основные функции – работу с БД, получение и отправку почты, работу с куками, обработку запросов из HTML - форм.

Библиотеки – расширяют и упрощают разработку кода. Часть из них изначально входят в стандартный пакет (графическая GD – библиотека), часть нужно подключать дополнительно.

Назначение некоторых других библиотек:

- Работа с E-mail: **Swift Mailer** , **PHPMailer**
- Работа с формами: **Securimage PHP Captcha** (создание капчей), **phpObjectForms** (проверка)
- Работа с базами данных: Propel, **ADOdb** (объектн.библиотека), **Doctrine**,(запросы) **PHPLINQ** (классы)
- Работа с документами: **TCPDF**, **PHPPowerPoint** , **PHPExcel** , **PhpRtf Lite**
- Работа с Javascript / AJAX: **PHPLiveX** , **Xajax**

Важная особенность функционирования – прорисовка страницы «с нуля» без сохранения внутренних переменных при обращении к файлу.

Язык PHP. Основы синтаксиса

Основная задача PHP- кода – вставка текста в html файл командами echo или print. Режим обращения к файлу - многопользовательский. Пока файл открыт – все в нем есть, по закрытию окна– все переменные, что в нем были определены, удаляются. Поэтому это и есть язык сценария или скриптовый язык – по ходу пишем html документ с учетом текущего состояния среды и реакции пользователя.

Общие особенности

Имена переменных – начинаются со знака «$», например, $x. За $ - любая последовательность

Чувствительность к регистрам:

- переменные различает
- служебные слова – не различает
- константы – зависит от способа их создания

Списки – через запятую

Комментарии:

- # или // - до конца строки
- /* … */ - произвольный фрагмент, не допускает вложенных аналогичных комментариев

Операторы – разделяются «;». В строке может быть много и один – в нескольких строках

Вставка кода php в текст html:

- XML – стиль (основной): `<?php … ?>` Самый популярный
- Сокращенный стиль: `<? … ?>` - требует активации Var «short_tags» в настройках php.
- SCRIPT – стиль: `<script language='php'> … </script>`
- ASP – стиль: `<% … %>` - если в настройках php задано asp_tags.

Текст php может вставляться в любое место html – кода *.php, *.php5 или *.html/htm – файла

Пример:

```
<body> Hellow <? Echo ', Word!' ?> </body>
```

Типы данных в PHP

Четыре скалярных типа:

1. **boolean** (двоичные данные) Раньше было 0=ложь, иначе истина. Сейчас появились true и false
2. **integer** (целые числа). ПУ – десятичное основание, префикс 0x – 16-ричное, О – восьмеричное, например `312, 0xFF, O177`
3. **double, float** (числа с плавающей точкой, или 'double') , например, `3.14, 314e-2`.
4. **string** (строки) – длина не фиксирована. Синтаксис в целом типовой, особенности будут описаны ниже.

Два смешанных типа (синтаксис будет дан далее):

1. **array** (массивы) – каждый элемент может иметь любой тип. Элемент может задаваться текстовым ключом (ассоциативный массив) или индексом, счет с 0 (индексные массивы)
2. **object** (объекты)

И два специальных типа:

1. **resource** (ресурсы) – задают специфические объекты, например, ссылки на открытые БД, на результат запроса, и т.д.
2. **NULL** ("пустые") – неопределенный тип переменной

Существуют также **несколько псевдотипов:** mixed (смешанные), number (числа), callback (обратного вызова)

Приведение типов

Язык не типизирован. Типы не объявляются. Тип переменной – в момент присваивания. Но можно конкретизировать и путем «приведения» типов. Например, `$z=(double)$x`, `$z=(double) $x*$y` . Есть и спец.функция `SetType($Var, "тип")`, где `"тип"` – это `int, integer` или `real,double, float` или `string,` например, `SetType($x, "int")`

Основные функции для работы с переменными:

IsSet (имя переменной:String) – проверка на наличие указанной переменной

Unset (имя переменной:String) - уничтожает содержимое переменной и возвращает занимаемые ею ресурсы системе.

Знаки операций

Конкатенация строк – это точка, например `'abc'.'def'`

Арифметические операции: `+, -, *, /` - как везде; `%` - деление по модулю.

Логические операции: `!` – это «НЕ», `||` – это «ИЛИ», `&&` – это «И», `and` и `or` – это те же «И» и «ИЛИ», но с меньшим приоритетом.

Поразрядные операции: `&, |, ~` («не»), `^` («исключающее ИЛИ»), `<<` (сдвиг вправо) и `>>` (сдвиг влево). При сдвигах первый операнд – в котором сдвигаем, второй операнд – на сколько позиций, например `x<<3`

Сравнение: `>=, <=,==, !=.`

Операции с классами: `new имя класса` – создание нового экземпляра, `->` - это обращение к элементам класса. Пример `$x=new Classname`. `$x->parameter1=null`.

Значение и тип операции – всегда равно значению левого операнда. Запись `$z=$y+($x=const)` эквивалентна записи `$z=$y+const`.

Сокращенные операции – аналогичны сокращенным операциям в C:

- Если результат и 1-й операнд – same Var, знак операции «уходит» влево.
  То есть `$y=$y+$x` эквивалентно `$y+=$x`,
- Суффиксные и префиксные инкременты и декременты - когда пишем `++$x, $x++, --$x` или `$x--`. Эти записи не эквивалентны. Если `$x=3`, то запись `echo $x++` выведет `3`, а `echo ++$x` – 4, хотя после этой команды `$x` будет равно 4 в обоих случаях.

Переменные переменных – это типа строковых подстановок. Если `$y='x'`, а `z=$x`, то можно записать, что `z=$$y`, то есть `$$y` это и есть `$x`, если `$y='x'`.

Ссылка – это копия переменной. Если имеем `$Var`, то `$copyVar=&$Var`. При таком определении `$copyVar` всегда будет иметь то же значение, что и `$Var`.
Используется при передаче в функции параметров по ссылке, а не по значению

Знак подавления ошибок `@` ставится перед выражением, в котором м.б. ошибка.
Например, `@ echo $y/$x` или `$x=@(5/0)`.

Команды ОС берутся в одинарные левые кавычки. Вызывают выполнение команды ОС из командной строки и возвращают результат ее действия. Пример: `$out=`dir c:`

.

Особенности задания строковых типов

Способы задания

1. 'текст' – допускает вставку двойных кавычек
2. “текст” – допускает вставку одинарных кавычек
3. **heredoc-синтаксис:** **<<<Ключ** <Текст …. > **Ключ**. Здесь **ключ** – это некоторый идентификатор: произвольная комбинация символов, лишь бы ее в тексте не было. Закрывающий идентификатор должен начинаться в первом столбце строки. Кроме того, идентификатор должен соответствовать тем же правилам именования, что и все остальные метки в PHP: содержать только буквенно-цифровые символы и знак подчеркивания, и должен начинаться не с цифры и не со знака подчеркивания, например:

```
<<<abc_22 Теперь можно писать любой текст с любыми знаками, одиночными и двойными кавычками…
abc_22 //признак конца текста
```

4. Специальные типы – задаются обратным слэшэм (знак «экранирования»). \n=LF, \r=CR, \t=TAB, \\=\, \$=$, \”=” и др. *Распознаются при применении двойных кавычек*!

Особенности конкатенации с переменными.

- Переменные могут вставляться в текст с автоматическим преобразованием в текст:

```
$z='text1'.$x+$y.' text2'
```

- Если текст заключен в двойные кавычки, то переменные внутри него автоматически преобразуются в текст, например,

```
$x=2; $y=3; $z= $x*$y; echo "Результат: $x*$y=$z";
```

выведет на html – страницу строку “**Результат: 2*3=6**”, тогда как строка

```
$x=2; $y=3; $z= $x*$y; echo 'Результат: $x*$y=$z';
```

выведет строку “**Результат: $x*$y=$z**”;

Константы

Задаются через функцию define:

```
define (S=имя константы, V=значение, [L=true=нечувств.к рег.])
```

Обычно пишут заглавными буквами

Свойства:

- не имеют приставки в виде знака доллара ($)
- могут иметь только скалярные значения
- могут быть определены и доступны в любом месте без учета области видимости
- не могут быть переопределены или аннулированы после первоначального объявления

Проверка наличия – функция `defined(<имя константы>)`

Пример:

```
define("pi",3.14,true);
if  (defined("pi")==true) echo "Константа pi объявлена!";
```

Стандартные (специальные) константы (значение зависит от места вызова):

1. `__LINE__` - Текущая строка в файле.
2. `__FILE__` - Полный путь и имя текущего файла.
3. `__CLASS__` - Имя класса. (Добавлена в PHP 4.3.0.)
4. `__DIR__` - каталог файла
5. `__FUNCTION__` - имя функции как было объявлено (с учетом регистра), PHP 4.3.0
6. `__METHOD__` – Имя метода класса. (Добавлена в PHP 5.0.0)
7. `__NAMESPACE__` - Имя текущего пространства имен с учетом регистра, определяется во время компиляции, PHP 5.3.0
8. `__TRAIT__` - Имя трейта (средство повторного использования кода в классах) с учетом регистра, PHP 5.4.0

Специальные константы нечувствительны к регистру

Массивы

В PHP существует два типа массивов – индексный и ассоциативный.

В первом типе элемент выбирается по индексу, во втором – по ключу (текстовая строка-идентификатор).

Индексы – могут быть численно-индексированные (счет с 0)

Ключи – это обычные литералы, текстовые константы или переменные

К элементам ассоциативных массивов можно обращаться как по номеру, так и по ассоциации.

Индекс или ключ ставят в квадратных скобках. **Пример:** `$x= $a[0], $y=$a['one']`

Если массив многомерный, то обращаются так: $a[\$i_1]\ [\$i_2]\ \ldots\ [\$i_N]$

Создание массивов:

1. `$var=array(список элементов через запятую)`
   - Элемент списка значений ключевого массива задается как: `ключ=>значение`:
   - Для индексных массивов элемент - просто значение, если 1-й элемент – нулевой индекс
   - Можно для первого элемента, если он не нулевой, указать `<начальный индекс>=<значение>`, и далее – только значения элементов через запятую.
2. Можно и «в лоб»: `<элемент с указанием индекса или ключа> = <значение>`
3. Можно просто писать `Var[]=Value` или `Var[«key»]=Valuekey`. При этом появляется очередной элемент массива.

Примеры: 1. `$color = array ('red', 'green', 'blue');`

1a. `$color = array ('one'=>'red', 'two'=>'green', 'three'=>'blue');`

1c. `$color = array (1='red', 'green', 'blue');`

2. Для индексных: `$color [0] = 'red'; $color[2]='blue';`

2. Для ассоциативных: `$color ['one'] = 'red'; $color ['two'] = 'green';` …

3. `$color [] = 'red'; $color [] = 'green'; $color [] = 'blue';`

Свойства и функции:

К тексту можно обращаться как к массиву.

Разложить массив на Var можно функцией list(Var1, Var2, … VarN)= массив из N элементов.

Если в массив преобразуется объект, то элементами массива станут свойства (переменные-члены) этого объекта. Ключами будут имена переменных-членов.

Сравнение массивов

Массивы можно сравнивать при помощи функции array_diff() и операторов массивов

Пример	Название	Результат
`$a + $b`	Объединение	Объединение массива $a и массива $b.
`$a == $b`	Равно	**TRUE**в случае, если $a и $b содержат одни и те же элементы.
`$a === $b`	Тождественно равно	**TRUE** в случае, если $a и $b содержат одни и те же элементы в том же самом порядке.
`$a != $b`	Не равно	**TRUE** если массив $a не равен массиву $b.
`$a <> $b`	Не равно	**TRUE** если массив $a не равен массиву $b.
`$a !== $b`	Тождественно не равно	**TRUE** если массив $a не равен тождественно массиву $b.

Пример сравнения массивов:

```
<?php
$a = array("apple", "banana");
$b = array("1" => "banana", "0" => "apple");//важно: ключ="1",а индекс=0

var_dump($a == $b); // bool(true)
var_dump($a === $b); // bool(false)
?>
```

Функции

```
function Имя_Функции(список переменных)
```

……

Return имя_переменной

Если передаем не по значению, а по ссылке – перед Var слитно ставим знак &, например,

```
&$arg[i]
```

Для задания области видимости – директива `global` (см. ниже на слайде)

Видимость переменных

Суперглобальные переменные

Их ID фиксированы. Это массивы, служащие для передачи параметров в сценарий.

Из них наиболее распространенные:

- **_GET** – массив переданных в php – файл параметров GET – методом:
  `file.php?p1=1&p2=«abc»&...`
  получаем значения как, например, `x=GET['p1']`
  Минус Get - метода – длина 2 кБайт
- **_POST** - массив переданных в php – файл параметров POST– методом (из форм).
  При обращении к файлу передаются значения всех тегов формы, имеющих имена, например `<input ... name='p1'>`

Глобальные – задаются директивой `global $Var.` Видны из любой функции

Все переменные сохраняются только на время работы файла.

Классы и объекты в PHP (http://php.net/manual/ru/language.oop5.traits.php)

Объявление класса

```
class Имя_класса {Var <список свойств>; <список функций> // описание членов
    класса - свойств и методов для их обработки }
```

Свойства описываются как `$Var [= Default Value]`.

Метод – описывается как функция, которая определяется как обычно.

Функция с именем класса – конструктор класса.

При описании можно использовать `this`.

Пример: `class Книга {Var $название, $тираж=1000, $цена; function Книга (…) }`

Объявление объекта

Объект-переменная = new <Имя_класса>;

Пример:

```
<?php // Создаем новый класс Coor:
    class Coor { // данные (свойства):
    var $name;
    var $addr;
    // методы:
    function Getname() {
       echo "<h3>John</h3>"; echo $this->name; // Использование this
    }
    }
    // Создаем объект класса Coor:
    $object = new Coor;
    // Получаем доступ к членам класса:
    $object->name = "Alex";   echo $object->name; // Выводит 'Alex'
    // Доступ к методу класса (фактически, к функции внутри класса):
    $object->Getname(); // Выводит 'John' (если name не задано)
?>
```

Вместо деструктора надо вызывать функцию PHP unset(). Она уничтожает содержимое переменной и возвращает занимаемые ею ресурсы системе

Классы и объекты в PHP (http://php.net/manual/ru/language.oop5.traits.php)

Обращение к элементам классов: http://www.php.su/learnphp/phpoo/?classes

Обращение к элементам классов осуществляется с помощью оператора :: "двойное двоеточие". Используя "двойное двоеточие", можно обращаться к методам классов. При обращении к методам классов программист должен использовать имена этих классов.

```
<?php
     class A {function example() {echo "Это первоначальная функция A::example().<br>"; }
     }
     class B extends A {
          function example() {echo "Это переопределенная функция B::example().<br>";
     A::example(); }
     }
     // Не нужно создавать объект класса A.
     // Выводит следующее:
     // Это первоначальная функция A::example().
     A::example();
     // Создаем объект класса B.
     $b = new B;
     // Выводит следующее:
     //   Это переопределенная функция B::example().
     //   Это первоначальная функция A::example().
     $b->example();
     ?>
```

Обращение к элементам классов в PHP5 (::) http://www.php.su/learnphp/phpoo/?php5_pm

Используя этот оператор ::, можно обращаться к константам, статическим или перегруженным свойствам или методам класса. При обращении к этим элементам извне класса, программист должен использовать имя этого класса.

Обозначение "двойное двоеточие" (::) не менялось ни разу в течение всего времени разработки PHP.

Использование :: вне объявления класса

```
<?php
class MyClass {
   const CONST_VALUE = 'Значение константы';
}
echo MyClass::CONST_VALUE;
?>
```

Классы и объекты в PHP (http://php.net/manual/ru/language.oop5.traits.php)

Использование :: в объявлении класса

Для обращения к свойствам и методам в объявлении класса используются ключевые слова self и parent.
Пример использования :: в объявлении класса:

```
<?php
class OtherClass extends MyClass {
   public static $my_static = 'статическая переменная';

   public static function doubleColon() {
      echo parent::CONST_VALUE . "\n";
      echo self::$my_static . "\n";
   }
}

OtherClass::doubleColon();
?>
```

Когда дочерний класс перегружает методы, объявленные в классе-родителе, PHP не будет осуществлять автоматический вызов методов, принадлежащих классу-родителю. Этот функционал возлагается на метод, перегружаемый в дочернем классе. Данное правило распространяется на конструкторы и деструкторы, перегруженные и другие методы.
Обращение к методу в родительском классе

```
<?php
class MyClass {

   protected function myFunc() {
     echo "MyClass::myFunc()\n";
   }
}

class OtherClass extends MyClass {

   /* Override parent's definition */
   public function myFunc() {

     /* But still call the parent function */
     parent::myFunc();
     echo "OtherClass::myFunc()\n";
   }
}

$class = new OtherClass();
$class->myFunc();
?>
```

Все специфические возможности, которыми обладает оператор :: в PHP5, не являются доступными в более ранних версиях PHP.

Структурные операторы. Операторы условия

```
1.  If (условие 1)
     Операторы 1;
    [ElsIf (условие 2)
        Операторы 2;]
     …
    [ElseIf (условие N)
     Операторы N;]
    [Else
     Операторы;]
```

2. Аналог Если(..) в Excel, Iif() в FoxPro:

```
Результат = (условие)?ExpressionTrue: ExpressionFalse
```

3. Break & Exit. Задают завершение:

- Break – условия переходов, циклов,
- Exit – всего PHP - сценария.

4. Switch:

```
Switch ($Var)  {
   Case value1: … Break;
   Case valueN: … Break;
   default: … Break;
}
```

Если не ставить Break, после первого выполненного оператора будут выполняться и все остальные.

Операторы циклов

1. **While (условие) выражение;**
   Пример:
   ```
   $x=0; while ($x++<10) echo $x;// Выводит 12345678910
   ```
   Обратите внимание на последовательность выполнения операций условия $x++<10. Сначала проверяется условие, а только потом увеличивается значение переменной.
   Если мы поставим операцию инкремента перед переменной (++$x<10), то сначала будет выполнено увеличение переменной, а только затем - сравнение.
2. **For (выражение1; условие; выражение2) операторы цикла;**
   выражение1 – задается перед началом цикла,
   условие – определяет условие проверки: пока истинно, цикл есть.
   Выражение2 – после каждого цикла, для изменения условия цикла.
   Пример: `For ($i=1; $i<=N; $i++) Echo "i=$i";`
3. **Do операторы while (условие);**
4. **Foreach (массив as $ключ=>$значение) команды;** Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара ключ=>значение оказывается в переменных $ключ и $значение.
   Другая форма записи, когда не интересует значение ключа очередного элемента:
   ```
   foreach (массив as $значение) команды;
   ```

Вложенные операторы:

1. **break(номер_цикла);** // Для вложенных циклов (указывается номер прерываемого цикла). Самый внешний цикл имеет макс.номер, номер внутреннего (ПУ)равен 1.
2. **Continue.** Конструкция работает так же, как и break, только "в паре" с циклическими конструкциями. Она немедленно завершает текущую итерацию цикла и переходит к новой (конечно, если выполняется условие цикла для цикла с предусловием). Точно так же, как и для break, для continue можно указать уровень вложенности цикла, который будет продолжен по возврату управления. В основном continue позволяет вам сэкономить количество фигурных скобок в коде и увеличить его удобочитаемость. Это чаще всего бывает нужно в циклах-фильтрах, когда требуется перебрать некоторое количество объектов и выбрать из них только те, которые удовлетворяют определенным условиям.

Подключение к БД из PHP

```
<?php
//------------------------ Определяем константы: ----------------------------*
define(DBuser, 'root');                          //логин  пользователя: root
define(DBpsw, '');                               //пароль пользователя: ''
define(DBserver, 'localhost');                   //Сервер БД: localhost(или http://...)
define(DBname, 'abc');                           //Название БД: в данном случае – «abc»
define(HOST, 'http://termo.vsib.astu/');         //URL сайта  (на этом слайде не нужно)
function connect() {//Подключение к SQL-серверу
    $DBLink=@mysql_connect(DBserver,DBuser,DBpsw) or die("Нет соединения с
    MySQL-server:".mysql_error());
  Return open($DBLink); // возвращается ссылка (ресурс) на подключение к серверу
} //mysql_pconnect – создает постоянное соединение
function open($DB) {//Открытие БД
 If ($DB) {
  $dbaseid=@mysql_select_db(DBname) or die("Нет соединения с БД:".mysql_error());
  if ($dbaseid) {
    mysql_query("SET CHARSET cp1251") or die (mysql_error());
    //@mysql_query("SET NAMES utf8");die–функция завершения работы+вывод аргумента
    //принудительно ставим нужную кодировку Win-1251/utf8
  } else return $dbaseid; //возвращаем дескриптор соединения
 } return $DB;
}
function disconnect() { //Закрытие соединения
$RES = @mysql_close($DBLink); return $RES; }?>
?>
```

Обмен данными с БД из PHP

```
<?php
//Вариант 1. Выполнение запроса. Его текст – в строке $query
function query($query) { $QRes = @mysql_query($query); //выполнить запрос
  if ($QRes == false) return false; //запрос не прошел
  //копим массив строк. Последняя строка - пустая и удаляется!
  while ($Res[] = mysql_fetch_array($QRes)); //Преобразуем ресурс в массив строк
  array_pop(&$Res); //Уменьшаем на 1 размерность  массива (пустой элемент NULL)
  return $Res; //Возвращаем массив строк
}
//Вариант 2. Выполнение запроса с разборкой результата в ассоциативный массив
function SQLk($query) { $QRes = @mysql_query($query);   //возвращаем ассоциативный
    массив данных
 if ($QRes == false) {If (!pconnect()) return false; //проверка соединения
     else $QRes = @mysql_query($query);};
 if ($QRes == false) return false;
 while ($x= mysql_fetch_assoc($QRes)) $Res[]=$x; return $Res; //копим массив
    строк запроса
}
//Вариант 3. Выполнение запроса с разборкой результата в индексный массив
    function SQLi($query) { $QRes = @mysql_query($query);    //возвращаем индексный
    массив данных
 if ($QRes == false) {If (!pconnect()) return false; //проверка соединения
     else $QRes = @mysql_query($query);};
 if ($QRes == false) return false;
 while ($x= mysql_fetch_row($QRes)) $Res[]=$x; return $Res; //копим массив строк
    запроса
}
?>
```

Обмен данными с БД из PHP

```
//Пример 1. Формирование фрагмента XML – данных для Adobe-Flash приложения
$res=query("SELECT id, name_r, Loc, type FROM chnl ORDER BY name_r ASC");
if($res)// еще есть строки с информацией об используемых в системе датчиках
{   $str .= "<sensors>";
    foreach($res as $row) //в row-поля id,Loc(место),type(тип),name_r(название)
    if($row['id']!="")
    {extract($row);//создаст из полей переменные id,Loc,type,name_r,…
     $SensorName=iconv("WINDOWS-1251", "UTF-8", $name_r);
     $str.="<sensor id=\"".$id."\" locID=\"".$Loc."\"type=\"".$type.
    "\"name=\"".$SensorName"\">".$name_r."</sensor>";
    } // здесь \"  - для включения в строку кавычки как символа
    $str .= "</sensors>";
} // результат вернет строку вида "<sensors> <sensor id="25" locID="ПК"
    type="temp" name="термодатчик 98ПК«\> ... <\sensors>"
//Пример 2. Формирование списка для раскрывающегося списка
$x=($Rus)? "Выберете организацию":"Select Organization";//рус/англ.текст.строка
echo "<option value='-1'>$x ...</OPTION>"; //вставлен текст переменной в HTML
$x=($Rus)? "OrgNmShtRus":"OrgNmShtEng";//краткое название рус/англ. текст.строка
$x="SELECT pk_org,OrgNmShtRus,OrgNmShtEng FROM Org ORDER BY $x";//строка запроса
$myselect = mysql_query($x); //выполнение запроса
while ($mysel=mysql_fetch_array($myselect)) //пока возвращаются не пустые строки
{$x='<OPTION value="'.$mysel[0].'"'; // атрибут value = pk_org
  //Если переменная формы имеет ту же аббревиатуру, пункт помечается выбранным:
  If ((int)$_POST['OrgID']==$mysel[0]) $x.=" Selected"; $x.='>';
 //Если выбран русский язык – вперед в пункте стоит русское,затем англ. название:
   $x.=($Rus)?"$mysel[1]"."  ($mysel[2])":"$mysel[2]"."  ($mysel[1])";
  echo $x.'</OPTION>'; //Вывод собранной строки с текстом HTML в текст HTML-файла
}
```

Обработка форм в PHP. Загрузка файлов

Код HTML - страницы

```
<form action='handler.php' method='post' enctype="multipart/form-data"/>
 //отображение на форме элементов: поля пароля, поля ввода текста и имени файла
 <input type="hidden" name="var" value="val"/> <input type="text" name="field"/>
 <input type="file" name="uploadfile"/> <input type="submit" value="Загрузить"/>
</form>//атрибут «name» задает ключ элемента в ассоциативном массиве $_POST
```

Настройка php.ini и прочие настройки

```
file_uploads=On //разрешение загрузки файлов на сервер по протоколу HTTP;
upoad_tmp_dir=/tmp //задание каталога для временного хранения загруженных файлов;
upload_max_filesize=2M //установка максимального объема загружаемых файлов.
Разрешение записи в каталоги загрузки
```

Код файла обработчика handler.php:

```
If (!Empty($_POST['field'])) $MyVar=$_POST['field']; //чтение значений поля ввода
$uploaddir = './files/'; // Каталог, в который мы будем принимать файл
$uploadfile = $uploaddir.basename($_FILES['uploadfile']['name']);//полное имя
// Копируем файл из каталога для временного хранения файлов:
if (copy($_FILES['uploadfile']['tmp_name'], $uploadfile)) {echo "Все ОК"; }
    else { echo "Ошибка загрузки"; exit; }
// Выводим информацию о загруженном файле:
echo "<p><b>Исходное имя: ".$_FILES['uploadfile']['name']."</b></p>";
echo "<p><b>Mime-тип файла:" .$_FILES['uploadfile']['type']."</b></p>";
echo "<p><b>Размер файла в байтах: ".$_FILES['uploadfile']['size']."</b></p>";
echo "<p><b>Временное имя: ".$_FILES['uploadfile']['tmp_name']."</b></p>";
```

Отправка почтовых сообщений – не нужно!

Исходные данные: `mail_to` – кому слать, `mail_msg` – текст сообщения, `mail_subject` - тема

```
if(empty($_POST['mail_to'])) exit("Введите адрес получателя");
// проверяем правильности заполнения с помощью регулярного выражения:
if (!preg_match("/^[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}$/i", $_POST['mail_to']))
   exit("Введите адрес в виде somebody@server.com");
$_POST['mail_to'] = htmlspecialchars(stripslashes($_POST['mail_to']));//Замена & на &amp и \\ на \
$_POST['mail_subject'] = htmlspecialchars (stripslashes($_POST['mail_subject']));
$_POST['mail_msg'] = htmlspecialchars (stripslashes ($_POST['mail_msg'])); $picture = "";
// Если поле выбора вложения не пустое - закачиваем его на сервер:
if (!empty($_FILES['mail_file']['tmp_name']))  { // Закачиваем файл:
  $path = $_FILES['mail_file']['name']; if (copy($_FILES['mail_file']['tmp_name'], $path))
    $picture = $path;
}   $thm = $_POST['mail_subject'];  $msg = $_POST['mail_msg'];
$mail_to = $_POST['mail_to'];  // Отправляем почтовое сообщение
if(empty($picture)) mail($mail_to, $thm, $msg);
else send_mail($mail_to, $thm, $msg, $picture);
// Вспомогательная функция для отправки почтового сообщения с вложением
 function send_mail($to, $thm, $html, $path) {$fp = fopen($path,"r");
  if (!$fp){print "Файл $path не читается"; exit();}$file= fread($fp,filesize($path));
  fclose($fp);$boundary="--".md5(uniqid(time()));//создан разделитель
  $headers.= "MIME-Version: 1.0\n Content-Type: multipart/mixed; boundary=\"$boundary\"\n";
  $multipart.= "--$boundary\n";$kod ='koi8-r' Content-Type: text/html; charset=$kod\n";
  $multipart.= "Content-Transfer-Encoding: Quot-Printed\n\n html\n\n";
  $message_part = "--$boundary\n Content-Type: application/octet-stream\n";
  $message_part.= "Content-Transfer-Encoding: base64\n Content-Disposition: attachment;"
  $message_part.= " filename = \"".$path."\"\n\n chunk_split(base64_encode($file))."\n";
  $multipart.= $message_part."--$boundary--\n";
  if(!mail($to,$thm,$multipart,$headers)){ echo "Письмо не отправлено"; exit();}
 } // htmlspecialchars – заменяет спецсимволы макроопределениями,
Stripslashes – удаляет экранирующие символы (обратный слэш)
```

PHP и Cookies. Другие полезные функции PHP – не нужно

Установка кук:

SetCookie(). Для этой функции можно указать 6 параметров. Обычно используют три первых

- name - задает имя, закрепленное за Cookie (string);
- value - определяет значение переменной (string);
- expire - время "жизни" переменной (целое число в секундах). Если данный параметр не указать, то Cookie будут "жить" до конца сессии, то есть до закрытия браузера. Если время указано, то, когда оно наступит, Cookie самоуничтожится.
- path - путь на сервере, где куки доступны (string); (If set to '/', the cookie will be available within the entire *domain*)
- domain - домен (string). В качестве значения устанавливается имя хоста, с которого Cookie был установлен;
- secure - передача Cookie через защищенное HTTPS-соединение (boolean).

Пример установки Cookies:

```
SetCookie("Test","Value"); // Устанавливаем Cookie до конца сессии:
SetCookie("My_Cookie","Value",time()+3600); // Устанавливаем Cookie на один 1час
```

Применение кук:

```
if (SetCookie("Test","Value")) echo "Cookies успешно установлены!"; //проверка
echo @$_COOKIE['test'];//получение данных из кук
SetCookie("Test",""); //удаляем Cookie
```

Сессии данных

```
session_start();      //создание сеанса/сессии
$_SESSION ['fn']=0;   //задание значения переменной сессии
If (IsSet($_SESSION[`fn'])) $x=$_SESSION [`fn']//использование переменной сессии
```

Вывод информации в файл:

```
$f = fopen(`filename.txt', "w"); fwrite($f,$txt); … fwrite($f,$txt); fclose($f);
```

Порядок создания диалоговой формы для л.р.7

1. В соответствии со структурой сайта, пользуясь только средствами HTML, причем с использованием преимущественно табличной верстки, реализовать разработанные шаблоны для всех основных страниц web – приложения:
   - страницы, содержащей информацию из базы данных (главной страницы);
   - страницы, содержащей информацию о задаваемых критериях отбора информации из базы данных (главной страницы);
   - страницы, содержащие форму, с помощью которой осуществляется взаимодействие с базой данных: запрос информации и ее модификацию (редактирование, добавление или удаление записи
   - Других типовых дополнительных страниц для работы со справочными данными – по необходимости;

   Примечание: общее число страниц определяет сам разработчик . Удобнее и проще всего для реализации основных функций обойтись одной страницей

2. С использованием HTML -кода занести на страницы подготовленный для них статический материал, не связанный с содержимым базы данных, в требуемой последовательности с учетом взаимного расположения объектов. При наличии нескольких страниц
3. Разместить на странице в усеченном виде таблицу (заменив вывод из многих строк данными из одной- двух строк) информацию, которая в дальнейшем будет взята из базы данных.
4. Выяснить, какая часть из отображаемой информации потребует в дальнейшем для ее правильного полного отображения написания сценария на стороне сервера в виде PHP - кода
5. Выделить на страницах структурные блоки, информация на которых может изменяться в процессе интерактивных действий пользователя
6. Освоить разные способы переходов между страницами и сайтами (включая переходы с использованием ссылок и изображений).
7. Проверить, что содержимое всех необходимых для расчетного задания страниц правильно отображаются на всех основных используемых на сегодня браузерах (Explorer; Mozilla FireFox, Opera, Chrome и, возможно, Safari)
8. Написать PHP-скрипт, генерирующий строки таблицы, отображающей хранимые в базе данные

Содержание отчета:

1. Скриншоты сверстанных страниц с комментариями
2. Эскизы дополнительных экранных форм, необходимых для реализации полного функционала приложения
3. Структуру меню, необходимого для реализации полного функционала приложения
4. Программный код сверстанных страниц с построчными комментариями