

ОГЛАВЛЕНИЕ

<i>ВВЕДЕНИЕ</i>	5
<i>БАЗОВЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ</i>	6
Структура HTML-страницы.....	9
Комментарии в HTML	10
Атрибуты тегов	11
Специальные символы	12
<i>ГИПЕРССЫЛКИ</i>	13
Относительная адресация	14
Внутренняя адресация	15
<i>ИЗОБРАЖЕНИЯ</i>	16
<i>КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ CSS</i>	18
Каскадирование и наследование	19
Селекторы	20
Псевдоклассы и псевдоэлементы	22
Теги и <DIV>	24
Свойства шрифта и текста	25
Управление цветом и фоном.....	26
<i>ТАБЛИЦЫ</i>	28
<i>ФОРМЫ</i>	30
Элемент <INPUT>.....	31
Элемент <SELECT>.....	33
Многострочное текстовое поле <TEXTAREA>	34
Пример формы	35
<i>PHP</i>	35
Синтаксис PHP	36
Переменные в PHP	36
Математические операции	37
Вывод данных на экран	37
Ветвление.....	37
Цикл.....	39

Массивы	41
Получение данных из формы	43
<i>MySQL</i>	43
Принципы хранения информации.....	43
Создание БД.....	46
SQL-запросы.....	49
<i>РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ HTML, CSS, PHP, MySQL</i>	51
Создание каркаса сайта и подключение таблицы CSS-стилей	51
Разделение сайта на блоки	54
Создание БД и получение из нее информации	56
Создание раздела «Статьи».....	58
Разработка зоны администрирования (Адмика).....	64
Добавление новой статьи	66
Редактирование статьи	68
Удаление статьи	71
Настройка меню	73
<i>ЛИТЕРАТУРА</i>	74
<i>СВЕДЕНИЯ ОБ АВТОРАХ</i>	74

ВВЕДЕНИЕ.

Данное методическое пособие предназначено для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.02 «Информационные системы и технологии» изучающих курс «Технология создания web-сервисов и web-приложений». Основные задачи курса:

Знать:

- возможности современных инструментальных программных средств для решения задач создания web-сервисов и web-приложений;
- основные современные библиотеки для автоматизации разработки web-ресурсов;
- особенности использования фреймворков для автоматизированной разработки web-ресурсов;
- технические методы разработки структуры сайта с использованием инструментальных средств;
- подходы к организации хранения данных web-ресурсов;
- приемы использования баз данных при организации работы web-ресурсов;
- существующие технологии организации работы клиент-серверных приложений.

Уметь:

- производить установку и настройку программного инструментария для решения задач создания web-ресурсов;
- осуществлять выбор необходимых программных средств для решения прикладных задач по созданию web-ресурсов;
- разрабатывать структуру хранилищ данных (баз данных) для работы web-ресурсов;
- осуществлять установку и настройку клиент-серверной модели для функционирования web-ресурсов;
- применять полученные знания для разработки концепции и структуры данных web-ресурсов в соответствии с предъявляемыми требованиями.

Владеть:

- методами настройки программного обеспечения для решения требуемых задач в рамках разработки web-сервисов и web-приложений;

- навыками работы с программными средствами в рамках разработки web-сервисов и web-приложений
- методиками создания и оптимизации аппаратно-программных компонентов и баз данных в рамках разработки web-сервисов и web-приложений
- приемами использования различных программных решений в рамках создания web-ресурсов.

Данный курс предполагает изучение базовых принципов написания структуры web-страницы на языке HTML и использования CSS для унификации отображения страниц и создания клиентской части приложений и сервисов. Также предполагается рассмотреть язык PHP для создания серверной составляющей приложений и сервисов, возможности интерактивной работы с системой и удаленным администрированием. Помимо этого в курсе рассматриваются вопросы использования баз данных MySQL для организации хранения данных приложений и сервисов в стандартизированном формате, что в значительной степени позволяет упростить дальнейшее использование и поддержку веб-приложений. Для возможности интерактивной работы с приложением на стороне клиента и создания динамических страниц предполагается рассмотреть в курсе язык JavaScript и его библиотеку jQuery, позволяющую формировать создавать скрипты на стороне клиента для управления веб-приложениями и веб-сервисами.

БАЗОВЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Для создания полноценного веб-приложения необходимо использовать достаточно большой набор различных программных решений, включающий в себя языки разметки (HTML), каскадные таблицы стилей (CSS), серверные скриптовые языки (PHP, Python), базы данных (MySQL), клиентские скриптовые языки (JavaScript), а также готовые библиотеки, что в значительной степени упрощает и автоматизирует процесс разработки (рис 1).

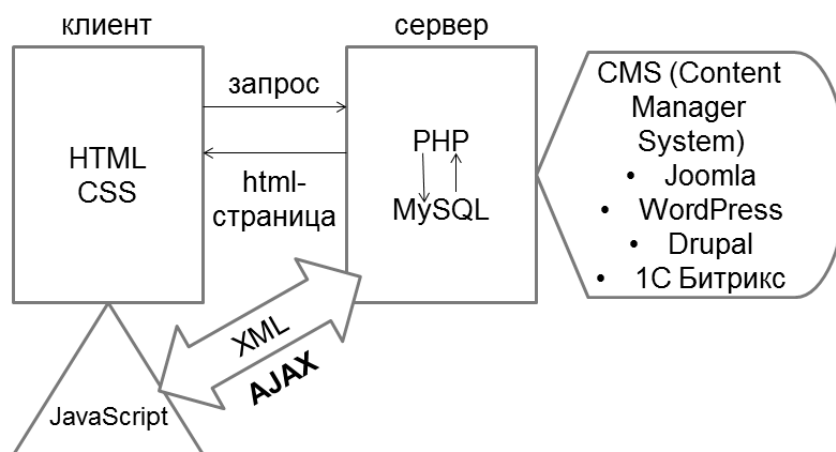


Рисунок 1. Организация построения современных веб-сервисов и веб-приложений

Основными понятиями при работе с веб-приложениями и веб-сервисами являются понятия веб-сервера, понятия сайта и страницы. Веб сервер - удаленный компьютер с предустановленным программным обеспечением, позволяющим обрабатывать http-запросы от клиента и направлять ему необходимые данные для отображения. В качестве такого программного обеспечения на сегодняшний день используются Apache (UNIX) и IIS (Windows). При этом как правило современные хостеры предоставляют на выбор пользователя оба варианта для работы. Следует отметить, что каждое программное обеспечение накладывает специфические особенности на разработку страниц проекта, что необходимо учитывать на начальных этапах проектирования.



Рисунок 2. Веб-сервер

Также неотъемлемой составляющей веб-приложений является понятие страницы. Это документ, содержащий необходимую пользователю информацию, оформленный и размеченный таким образом, чтобы программное обеспечение на стороне клиента (браузер) могло ее корректно отобразить.

Под понятием «сайт» принято понимать совокупность страниц, оформ-

ленных в едином стиле, находящихся в едином пространстве доменного имени и имеющих внутреннюю навигацию. Таким образом в качестве примера можно привести сайт yandex.ru, содержащий набор страниц: карты, маркет, новости, переводчик, музыка и т.д.

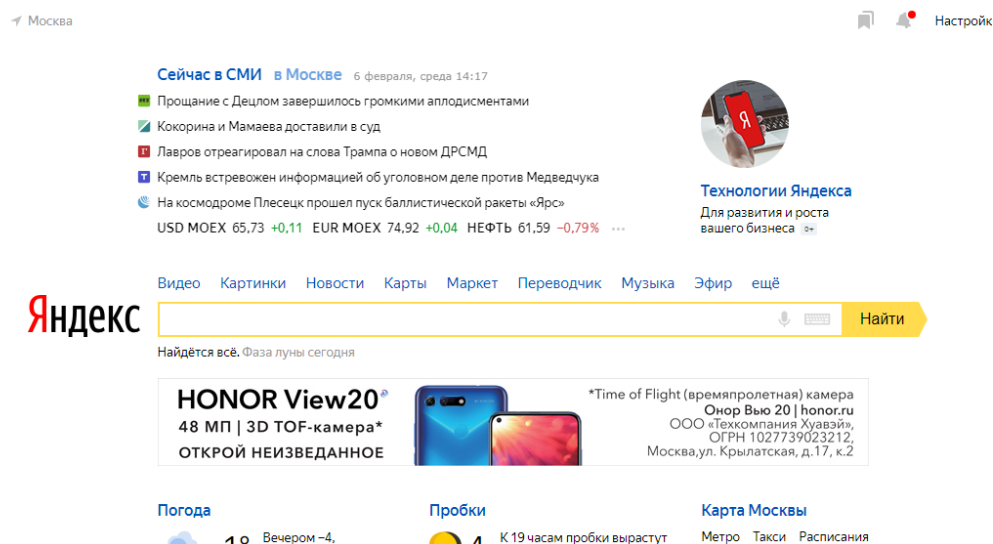


Рисунок 3. Сайт Яндекс

Для просмотра страницы на стороне клиента используется специализированное программное обеспечение – браузер. На сегодняшний день на рынке представлено достаточно большое количество браузеров: Google Chrome, Яндекс браузер, Mozilla Firefox, Opera и другие. Каждый из них отличается особенностями отображения страниц и прочтением html-кода. Таким образом при разработке веб-приложений необходимо руководствоваться требованиями заказчика для обеспечения необходимого уровня кроссбраузерности. Однако, на сегодняшний день наиболее популярными являются браузеры Google Chrome и Яндекс браузер, поскольку максимально корректно отображают страницы за счет использования специальных интеллектуальных алгоритмов.

Для создания веб-сервисов и веб-приложений необходимо на начальных этапах проектирования создавать каркас документа, основанный на разметке расположения элементов. Для этих целей используется язык гипертекстовой разметки HTML, предложенный Тимом Бернсом Ли 1986 году и принятый для разметки страниц в 1991 году. Основным элементом для разметки страниц является тег – специальный код заключенный в знаки $< >$. Все теги жестко регламентированы стандартом и их спецификации можно изучить на официальном сайте консорциума w3c.org.

Теги разделяются на две группы: парные и непарные. Парные теги воз-

действуют на часть страницы и имеют начало действия и конец (например, жирный текст). Непарные теги выполняют законченное действие (например, переход на новую строку).

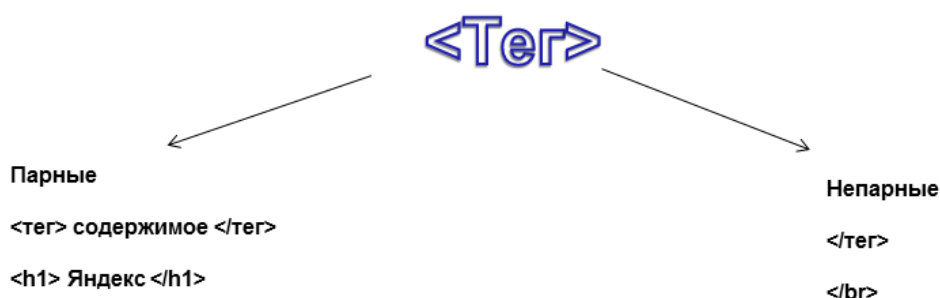


Рисунок 4. Парные и непарные теги

Структура HTML-страницы

HTML-страницы создаются при помощи разметки текста и изображений, а также других внешних ресурсов тегами. При создании HTML-страниц следует использовать любые текстовые редакторы для написания кода. При этом более удобными являются специализированные редакторы, позволяющие «подсвечивать» код, тем самым упрощая его чтение. Также удобны редакторы с автоматическим вводом тегов, что во много раз упрощает и ускоряет процесс написания кода. В качестве таких редакторов можно использовать Notepad++, Adobe Dreamweaver, Sublime Text 2 и другие.

Отдельно следует отметить класс редакторов кода типа WYSIWYG. Данный тип редакторов позволяет создавать страницы в визуальном режиме, т.е. работать в том числе с уже отображенной в окне страницей и редактировать ее в режиме просмотра. Данный подход не рекомендуется использовать при создании web-страниц, так как это во-первых, приводит к некорректному применению тегов (так как система сама выбирает теги исходя из действий разработчика страницы), а во-вторых, отображение страницы в данных программах не всегда соответствует реальному отображению в различных браузерах. Таким образом рекомендуется использовать подобный класс редакторов для удобства поиска необходимых элементов на странице путем их выделения в окне визуализации и синхронной подсветкой в окне кода. К таким редакторам относятся Adobe Dreamweaver, Microsoft FrontPage и другие.

Для создания страницы следует в начале создать базовый шаблон, включающий в себя типовую разметку тегами, а далее создавать необходимые элементы на странице.

Также рекомендуется при написании какого-либо кода проверять его в браузере. Таким образом максимально эффективным является процесс при котором одновременно открыта страница в браузере и в редакторе кода. Переключение между этими окнами позволит в максимальной степени эффективно создавать web-страницы.

Для создания базового шаблона страницы следует использовать следующую конструкцию (рис. 5):

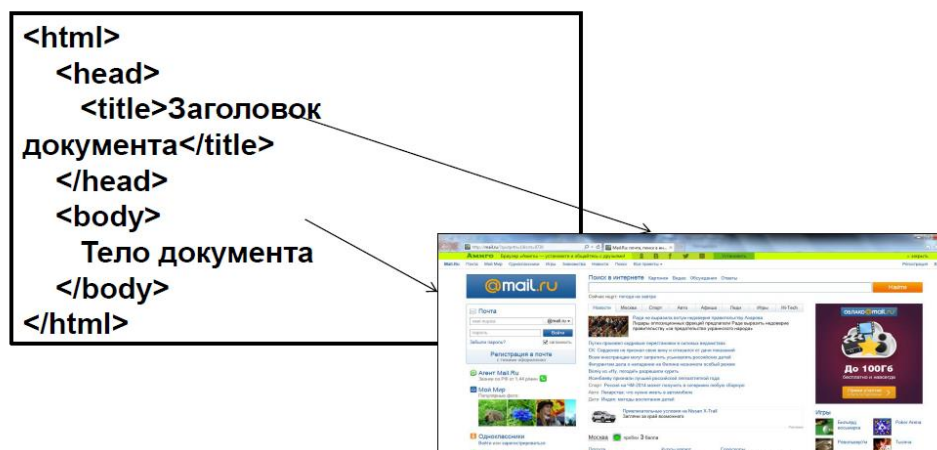


Рисунок 5. Шаблон HTML-страницы

Раздел <BODY> отвечает за содержимое страницы (т.е. ее видимую часть), а раздел <HEAD> за конфигурационную составляющую страницы. В разделе <HEAD> располагается информация о заголовке страницы (раздел <TITLE>), CSS-стили, JS-скрипты, мета-теги и другие элементы.

alink	Цвет активной ссылки.
background	Фоновое изображение для страницы.
bgcolor	Задаёт фоновый цвет web-страницы.
bgproperties	Задаёт, будет ли прокручиваться фон содержимым страницы
bottommargin	Внешний отступ начиная с нижнего края окна страница до контента.
leftmargin	Внешний отступ по горизонтали от края окна браузера до контента.
link	Цвет ссылок на web-странице.
scroll	Задаёт, отображать полосы прокрутки или нет.
text	Цвет текста на странице.
topmargin	Внешний отступ от верхней границы окна браузера до содержимого страницы.
vlink	Цвет ссылок, которые пользователь посетил ранее.

Таблица 1. Атрибуты тега <BODY>

Комментарии в HTML

Комментарий в коде – вспомогательная информация, не обрабатываемая

браузером. Как правило, комментарии используются для «служебной» информации, когда разработчик хочет дать дополнительное описание какому-либо участку кода для возможности дальнейшей правки спустя какое-то время. Однако на практике комментарии удобно использовать для отладки кода. Данный процесс предполагает внесение в блок комментария части кода при отладке. Таким образом «закомментированный» участок кода не обрабатывается и можно выяснить – являлся ли он причиной неправильной работы web-страницы. В общем виде технология отладки при помощи комментариев сводится к процессу, когда в комментарий вносится большой участок кода, а далее этот участок уменьшается и при просмотре страницы обнаруживается момент некорректной работы участка кода.

Правило записи комментария в HTML следующее:

```
<!--Комментарий-->
```

В коде web-страницы может использоваться сколь угодно много блоков комментариев.

Атрибуты тегов

Для каждого тега в HTML предусмотрен набор атрибутов, определенных стандартом языка. Атрибуты позволяют производить корректировку параметров тега. Например, тег сам по себе ни каким образом не воздействует на текст, расположенный внутри него. Но, при использовании атрибута color тексту можно назначить цвет, отличный от черного (по умолчанию определенного браузером). Для каждого тега можно применять несколько атрибутов. В этом случае они записываются через пробел. Общий принцип записи атрибутов тега сле

```
<тег атрибут1="значение" атрибут2="значение">содержимое </тег>
```

В случае, если используется непарный тег, следует атрибуты размещать непосредственно в нем, а если парный – то в открывающем теге.

Примеры

```
<font color='red' size='2'> Добрый день </font>
```

```
<hr size='2' align='center' noshade>
```

Специальные символы

Специальные символы в HTML-символы, которые являются зарезервированными (например, $\langle \rangle$), либо символы отсутствующие на клавиатуре (например, ©). Для их добавления используется следующая конструкция (рис. 6)

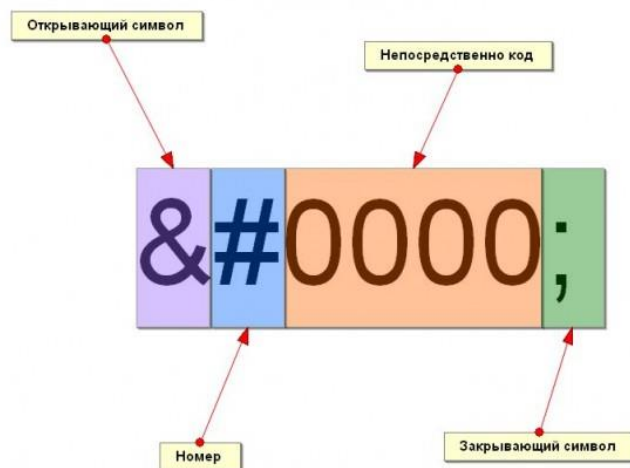


Рисунок 6. Структура кода для вставки спецсимвола

При этом для самого кода используется четырехзначный блок. У каждого символа, использующегося в кодировке, есть код. Для просмотра этого кода достаточно открыть кодовую таблицу (таблица 1).

Символ	Назначение	Мнемоника	Код
	неразрывный пробел	 	
¢	символ цента	¢	¢
£	символ фунта	£	£
¤	знак валюты	¤	¤
¥	символ иены	¥	¥
§	знак параграфа	§	§
©	знак охраны авторского права	©	©
«	направленная влево двойная угловая кавычка	«	«
»	направленная вправо двойная угловая кавычка	»	»
®	знак правовой охраны товарного знака	®	®

Таблица 2. Пример кодовой таблицы

В случае частого использования кода какого-либо символа можно использовать его мнемонику – специальное название. Это позволяет использовать спецсимволы без поиска нужного кода в кодовой таблице. Например, символ © можно вставить по коду © или используя мнемонику ©. И первое и

второе даст один и тот же результат.

ГИПЕРССЫЛКИ

В основе языка HTML лежит понятие гипертекста – системы из некоторого набора страниц с возможностью перехода между ними. Для реализации этого подхода в HTML существуют гиперссылки.

Гиперссылка - часть страницы (текст, изображение, кнопка и т.д.), кликнув по которой происходит переход на другой ресурс (страницу, сайт, файл, почтовый адрес и т.д.).

Все гиперссылки можно разделить на две группы:

1. Внешние, позволяющие осуществлять навигацию за пределы страницы (переход на другую страницу, другой сайт, на скачивание файла, на почтовый клиент и т.д.);
2. Внутренние, позволяющие осуществлять навигацию внутри страницы (организация оглавления).

Для создания гиперссылки используется парный тег <A>. Содержимое между открывающим и закрывающим тегом становится гиперссылкой.

Для тега <A> характерны следующие параметры (таблица 3):

Атрибут	Действие
href	Задаёт адрес документа, на который следует перейти.
name	Устанавливает имя якоря внутри документа.
target	Имя окна или фрейма, куда браузер будет загружать документ.
title	Добавляет всплывающую подсказку к тексту ссылки

Таблица 3. Атрибуты тега <A>

Примеры использования тега <A> для создания гиперссылок

Пример 1 (ссылка на ресурс yandex.ru):

```
<a href='http://www.yandex.ru' target='_blank'>Перейти на сайт Ян-  
декс </a>
```

Пример 2 (ссылка на другую страницу сайта):

```
<a href='kont.htm'>Контакты</a>
```

Для навигации внутри сайта существует два типа адресации:

1. Абсолютная адресация, которая предполагает указание адреса перехода начиная с корневого каталога носителя информации. В случае если сайт расположен на компьютере – буквы диска, если в Интернете – доменного имени.

Пример:

```
c:\мои документы\сайт\kontakt\kontakt.htm
```

2. Относительная адресация, которая предполагает указание адреса перехода относительно текущего местоположения страницы.

Пример:

```
kontakt\kontakt.htm
```

Использование абсолютной адресации является крайне неправильным подходом, так как перенос сайта с одного носителя на другой или размещение его на хостинге приведет к тому, что все ссылки перестанут работать и появится необходимость их изменять с учетом текущего местоположения. Для корректной работы сайта и обеспечения его переносимости следует использовать относительную адресацию.

Относительная адресация

При создании относительного адреса следует учитывать текущее местоположение:

3. Если страница находится в том же каталоге – следует указать в ссылке ее название файла с расширением.

4. Если страница выше по уровню в дереве каталогов – следует использовать ../ для перехода на 1 уровень выше.
5. Если страница ниже по уровню в дереве каталогов – следует указывать имя внутреннего каталога/ название файла с расширением.

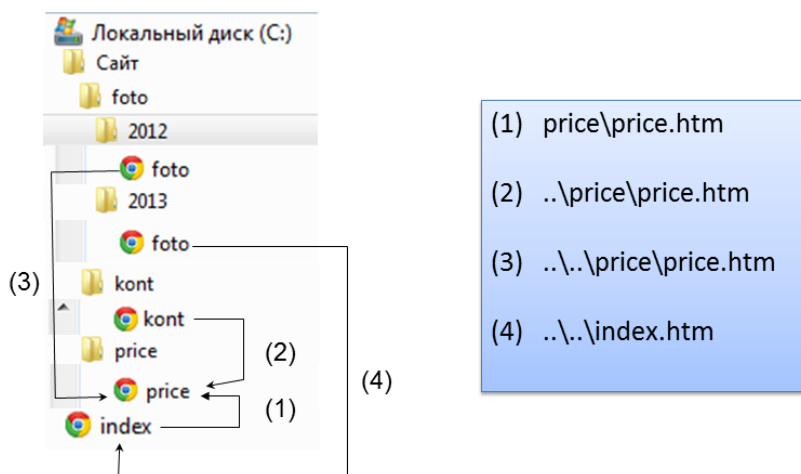


Рисунок 7. Примеры создания ссылок при относительной адресации

Для внешней адресации существует возможность перехода не только на другую страницу, но и другой сайт, почтовый клиент с указанием в поле «Кому» почтового адреса, переход на скачивание файла. Для этого перед ресурсом устанавливается соответствующий префикс.

Примеры:

`href= 'http://www.yandex.ru'`

`href= 'mailto:e.kashkin@gmail.com'`

`href= 'file:l.doc'`

Типы префиксов:

- **http:** - переход на Интернет-ресурс;
- **mailto:** - переход на почтовый клиент;
- **file:** - переход на скачивание файла.

Внутренняя адресация

Данный тип адресации позволяет осуществлять навигацию внутри страницы. Данный подход является удобным, в случае наличия на одной странице большого количества информации и необходимости прокручивать ее для поиска нужного раздела. Примером таких страниц являются ленты новостей соци-

альных сетей, электронные учебники и т.д. Для решения задачи быстрого перехода к нужному разделу следует в начале страницы формировать оглавление и используя «якорные» ссылки осуществлять навигацию внутри страницы (рис. 8).

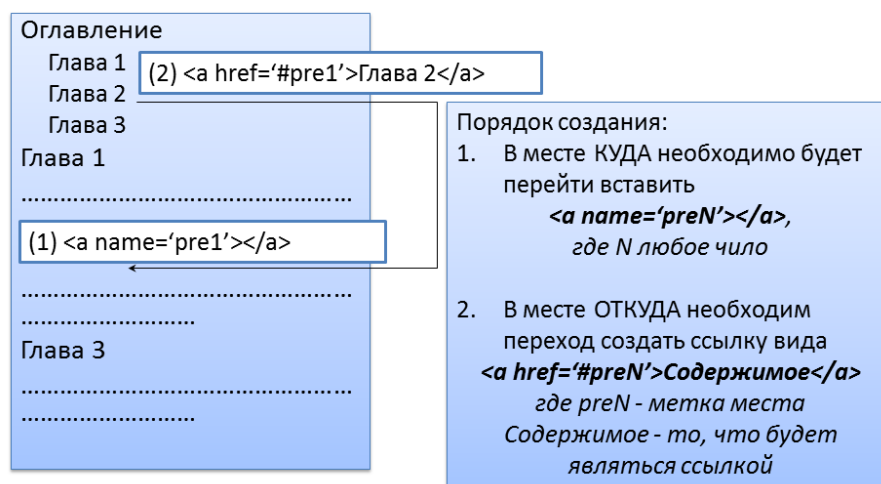


Рисунок 8. Принцип создания внутренней адресации на странице

ИЗОБРАЖЕНИЯ

Ни один современный web-ресурс не обходится без наличия на нем различных изображений. Это может быть и непосредственно фотографии, графики диаграммы, и элементы дизайна, и анимированные элементы для создания дружелюбного интерфейса.

Современные web-ресурсы поддерживают 3 основных типа изображений – GIF, JPEG и PNG, каждый из которых обладает рядом характеристик, определяющих сферу его применения при создании web-сервисов и web-приложений (Таблица 4).

Формат	Характеристики	Назначение
GIF	Количество цветов - 256 Прозрачность - ДА Анимация - ДА	Анимированные кнопки, аватарки
JPEG	Количество цветов - 16/32 млн. Прозрачность - НЕТ Анимация - НЕТ	Фотографии, изображения
PNG	Количество цветов - 16/32 млн. Прозрачность - ДА Анимация - НЕТ	Элементы дизайна сайта

Таблица 4. Форматы изображений WEB

Для добавления изображения на страницу используется непарный тег . При этом по умолчанию тег является блочным, то есть заполняет все пространство по ширине. Для тега характерны следующие атрибуты (Таблица 5).

Атрибут	Назначение
Width, height	ширина, высота изображения
Alt	описание картинка, выводится при наведении мыши
Border	черная рамка вокруг изображения
Hspace, vspace	отступы от изображения по горизонтали и вертикали
Align	расположение изображение и обтекание его текстом

Таблица 5. Атрибуты тега

Пример добавления изображения на страницу:

```
<img src='1.jpg' alt='Цветы'>
```

1.jpg – файл изображения

alt='Цветы' – всплывающая подсказка при наведении курсора мыши

Атрибут **align** отвечает за обтекание содержимого страницы вокруг изображения. При этом по умолчанию тег является блочным. Однако применение данного атрибута переводит его в строчный элемент. Возможные значения атрибута **align**:

- left
- right
- top
- middle
- bottom

При этом следует отметить, что используя значения атрибута align разместить изображение по центру страницы не представляется возможным. Для этого используется конструкция, при которой изображение помещается в блочный элемент, а тот в свою очередь выравнивается по центру.

```
<p align='center'></img src='1.jpg'></p>
```

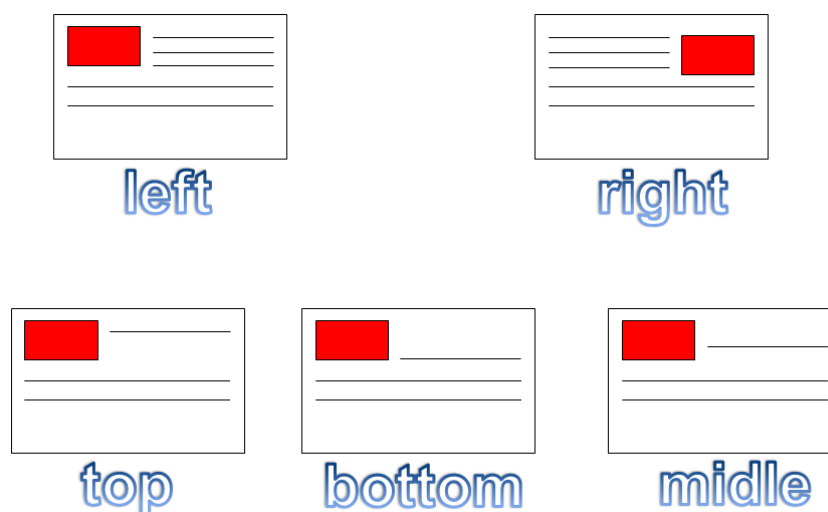


Рисунок 9. Обтекание изображения в зависимости от значения атрибута align

КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ CSS

Современные web-приложения включают в себя не только информационный контент, но и средства, улучшающие визуальное восприятие. Именно для этой задачи в 1996 году были предложены каскадные таблицы стилей (Cascading Style Sheets – CSS). Это формальный язык описания внешнего вида документа, написанного с использованием языка разметки. По сути CSS является «надстройкой» для языка HTML и позволяет расширить возможности оформления web-страницы за счет воздействия на разметку документа. Для CSS характерны две основные задачи:

1. Расширяют функциональные возможности языка HTML с точки зрения оформления страниц;
2. Унифицируют процесс оформления HTML страниц.

Помимо возможностей «расширения» возможностей HTML с точки зрения оформления CSS еще позволяет локализовать настройки и управлять ими централизованно, что в значительной степени облегчает управление web-приложениями.

Для задания CSS-стилей используются CSS-правила (Рис.)

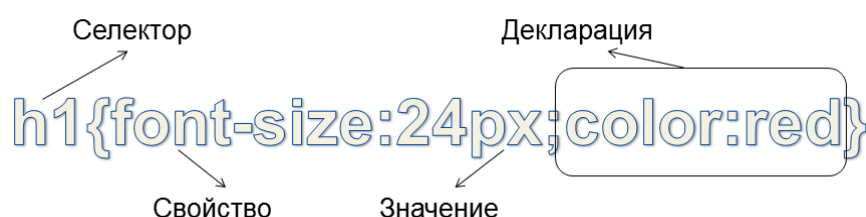


Рисунок 10. Структура CSS-правила

CSS-правило состоит из селектора – объекта к которому применяются правила и набора деклараций. Каждая декларация – это css-свойство и его значение, разделенные двоеточием. Для одного селектора может использоваться несколько деклараций, разделенных точкой с запятой.

Для того, чтобы задать CSS-стили для HTML-документа существует 3 базовых способа (рис. 11):

1. Связывание с внешним файлом;
2. Встраивание стилей в документ;
3. Определение стилей на уровне тега (вложение).

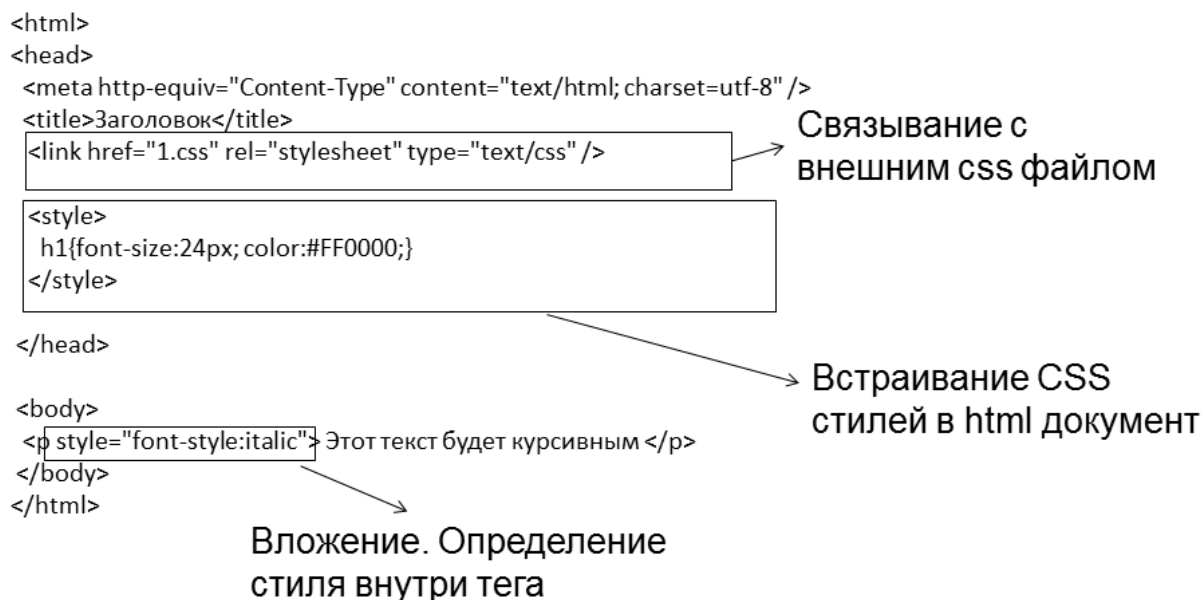


Рисунок 11. Способы определения стилей

Каскадирование и наследование

В случае если для одного селектора определено несколько различных свойств, то для определения итогового свойства следует руководствоваться правилом «Чем ближе к селектору тем больше приоритет» (рис.). При условии, что существует 3 уровня установки стилей CSS: внешний файл, встроенные в html-документ стили и определение стиля на уровне тега, то при одинаковом наборе стилей для конкретного селектора максимальный приоритет будет у атрибута style (определение стиля на уровне тега).

Наследование также является неотъемлемой составляющей каскадных таблиц стилей. Основная задача – определение различного набора стилей на

разных уровнях и вычисление итогового значения для конкретного тега. При этом в дереве стилей корневым узлом является внешний файл, а определение стилей на уровне тега – листом дерева (рис. 12).

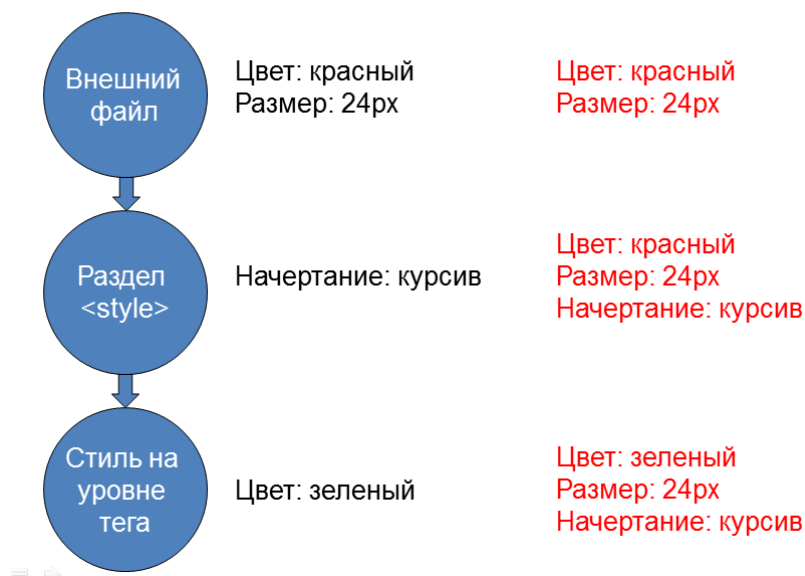


Рисунок 12. Определение стилей при наследовании

Селекторы

CSS селектор – это составная часть CSS правила отвечающая за определение конкретных HTML тегов, к которым будут применены стили оформления, прописанные в этом правиле. Существует несколько различных видов селекторов для возможности «гибкого» назначения правил для тегов HTML.

Селектор «Название тега»

Применяется автоматически к конкретному тегу на всей web-странице. Название селектора – это название тега без знаков <>.

`p{border:1px solid red;}`

устанавливает рамку красного цвета для всех абзацев на странице

Селектор «Класс тега»

Применяется автоматически ко всем тегам, у которых в качестве атрибута class указан соответствующий класс на web-странице. Имя класса может быть любое (латинские буквы и цифры, первый символ – латинская буква).

```
p{border:1px solid red;}
```

```
p.size {font-size:24px;}
```

```
<p> Текст в красной рамке </p>
```

```
<p class='size'> Текст в красной рамке размера 24 пикселей </p>
```

Селектор «Независимый класс»

Применяется автоматически к любому тегу у которого в качестве атрибута class указан соответствующий класс на web-странице.

```
p{border:1px solid red;}
```

```
.cont {font-size:24px;}
```

```
<p> Текст в красной рамке </p>
```

```
<p class='cont'> Текст в красной рамке размера 24 пикселей </p>
```

```
<h1 class='cont'> Заголовок размера 24 пикселя </h1>
```

Селектор «Идентификатор»

Применяется автоматически к любому тегу у которого в качестве атрибута class указан соответствующий класс на web-странице.

```
p{border:1px solid red;}
```

```
#cont {font-size:24px;}
```

```
<p> Текст в красной рамке </p>
```

```
<p id='cont'> Текст в красной рамке размера 24 пикселей </p>
```

```
<h1 id='cont'> Заголовок размера 24 пикселя </h1>
```

Селектор «Контекстный селектор»

Выполняются лишь в том случае, если на странице встречается ИМЕННО ТАКАЯ комбинация вложенности тегов, какая указана при определении селектора. Имя класса может быть любое (латинские буквы и цифры).

```
p{border:1px solid red;}  
p b{font-size:24px;}
```

<p> Текст в красной рамке </p>

<p> Текст в красной рамке размера 24 пикселей </p>

<h1 class='cont'> Заголовок размера 24 пикселя </h1>

Группировка свойств

Если для одного селектора применяется несколько отдельных свойств, то их можно прописать один раз через точку с запятой (рис. 13).

```
p{font-size:12px;}  
p{color:red;}  
p{border:1px solid black;}  
↓  
p{font-size:12px; color:red;  
border:1px solid black;}
```

Рисунок 13. Группировка свойств

Аналогично, если одно и то же свойство используется для нескольких селекторов, можно прописать его указав селекторы через запятую, а далее в фигурных скобках указать свойства (рис. 14).

```
p{font-size:12px;}  
h1{font-size:12px;}  
h3{font-size:12px;}  
↓  
p, h1, h3{font-size:12px;}
```

Рисунок 14. Группировка селекторов

Псевдоклассы и псевдоэлементы

Псевдоклассы определяют динамическое состояние элементов, которое

изменяется со временем или с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с их помощью можно получить разные динамические эффекты на странице.

Существует 4 базовых псевдокласса, определяющих состояние гиперссылок:

1. `:hover` – при наведении курсора мыши;
2. `:link` – непосещенная ссылка;
3. `:active` – активная ссылка (в момент удержания левой кнопки мыши);
4. `:visited` – посещенная ссылка.

Пример:

`a:hover{cursor:pointer}`

При наведении мыши на ссылку курсор меняет вид

Псевдоэлементы позволяют задать стиль логических элементов, не определенных в дереве элементов документа (первая строка абзаца, выделенный текст и т.д.)

Псевдоэлемент	Назначение
::-moz-selection	Применяется к выделенному пользователем фрагменту документа. Поддерживается только браузером Firefox
::-ms-browse	Позволяет задать стиль кнопки «Обзор» при загрузке файлов через <code><input type="file"></code> в Internet Explorer
::-ms-check	Задаёт стиль переключателей (<code>input type="radio"</code>) и флажков (<code>input type="checkbox"</code>)
::-ms-clear	Задаёт стиль кнопки для очистки текстового поля. Исходно эта кнопка не видна, она появляется в правой части поля только при вводе текста
::-ms-expand	Задаёт стиль кнопки раскрытия списка, созданного с помощью тега <code><select></code> в браузере Internet Explorer
::-ms-fill	Задаёт стиль индикатора элемента <code><progress></code> в браузере Internet Explorer. Само значение индикатора и его положение

	меняется динамически посредством скриптов
::-ms-reveal	Задаёт стиль кнопки для просмотра пароля в поле <code>input type="password"</code> . Кнопка исходно не видна и появляется при вводе пароля в правой части поля
::-ms-value	Позволяет изменять стиль элементов формы, сделанных с помощью тега <code><input></code> или <code><select></code> , в браузере Internet Explorer
::after	Псевдоэлемент CSS3, который используется для вывода желаемого контента после элемента, к которому он добавляется
::before	Псевдоэлемент CSS3, применяется для отображения желаемого контента до элемента, к которому он добавляется
::first-letter	Псевдоэлемент CSS3, определяет стиль первого символа в тексте элемента, к которому добавляется
::first-line	Псевдоэлемент CSS3, задает стиль первой строки форматированного текста
::selection	Применяет стиль к выделенному пользователем фрагменту текста
:after	Используется для вывода желаемого контента после элемента, к которому он добавляется
:before	Применяется для отображения желаемого контента до элемента, к которому он добавляется
:first-letter	Определяет стиль первого символа в тексте элемента, к которому добавляется
:first-line	Задаёт стиль первой строки форматированного текста

Таблица 6. Псевдоэлементы

Для применения псевдоэлементов к селектору используют то же правило, что и при применении псевдоклассов.

Теги `` и `<DIV>`

Эти два тега предназначены для создания блоков в html документе. При этом визуально они ни как не влияют на содержимое.

Тег `` является строчным тегом и позволяет создать строчный контейнер без применения каких-либо дополнительных свойств.

Тег `<DIV>` является строчным тегом и позволяет создать строчный контейнер без применения каких-либо дополнительных свойств.

Свойства шрифта и текста

Для управления шрифтом на web-странице существует следующий набор свойств (таблица 7).

Свойство	Значение	Описание	Пример
font-family	имя шрифта	Задаёт список шрифтов	P {font-family: Arial, serif}
font-style	normal italic oblique	Нормальный шрифт Курсив Наклонный шрифт	P {font-style: italic}
font-variant	normal small-caps	Капитель (особые прописные буквы)	P {font-variant: small-caps}
font-weight	normal lighter bold bolder 100–900	Нормальная жирность Светлое начертание Полужирный Жирный 100 — светлый шрифт, 900 — самый жирный	P {font-weight: bold}
font-size	normal pt px %	нормальный размер пункты пиксели проценты	font-size: normal font-size: 12pt font-size: 12px font-size: 120%

Таблица 7. Свойства шрифта

Для управления представлением текста на web-странице существует следующий набор свойств (таблица 8).

Свойство	Значение	Описание	Пример
line-height	normal множитель значение %	Интерлиньяж (межстрочный интервал)	line-height: normal line-height: 1.5 line-height: 12px line-height: 120%
text-decoration	none underline overline	Убрать все оформление Подчеркивание Линия над текстом	text-decoration: none

	line-through blink	Перечеркивание Мигание текста	
text-transform	none	Убрать все эффекты	text-transform: capitalize
	capitalize	Начинать С Прописных	
	uppercase	ВСЕ ПРОПИСНЫЕ	
	lowercase	все строчные	
text-align	left	Выравнивание текста	text-align: justify
	right		
	center		
	justify		
text-indent	значение	Отступ первой строки	text-indent: 15px; text-indent: 10%
	%		

Таблица 8. Свойства текста

Управление цветом и фоном

Для задания цвета в CSS существуют следующие варианты:

1. Символьное представление (green);
2. Шестнадцатичное представление в модели RGB (#00FF00);
3. Десятичное представление в модели RGB (0, 255, 0);
4. Процентное представление в модели RGB (0%, 100%, 0%).

Модель RGB (Red Green Blue) является аддитивной моделью (основа – свет) и формирует цвета на основе смешения трех каналов – красного, зеленого и синего. Значение канала изменяется от 0 до 255 в десятичном формате и от 00 до FF в шестнадцатичном формате соответственно. При этом нулевое значение определяют отсутствие соответствующего цвета в итоговом. При этом при значениях отличных от нуля действует правило – чем меньше значение тем темнее цвет. Таким образом белый цвет имеет значение #FFFFFF, черный - #000000, красный - #FF0000, серый - #CCCCCC(все каналы имеют одинаковое значение).

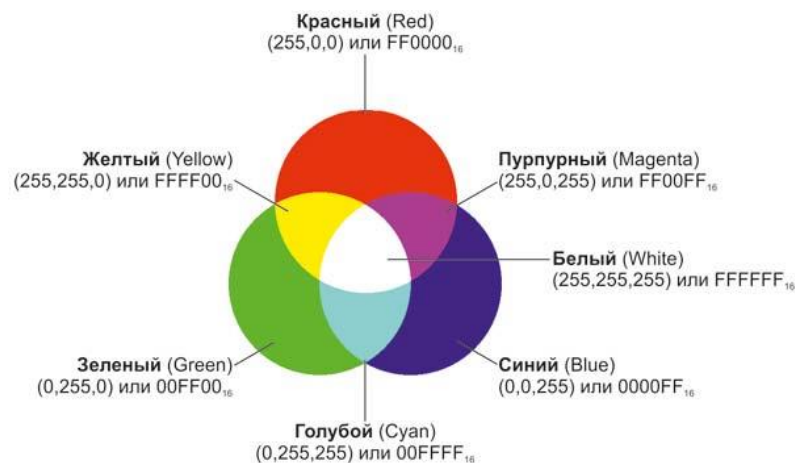


Рисунок 15. Формирование цвета в модели RGB

Для управления цветом и фоном на web-странице существует следующий набор свойств (таблица 9).

Свойство	Значение	Описание	Пример
color	Цвет	Устанавливает цвет текста	P { color: #330000 }
background-color	Цвет transparent	Цвет фона	BODY { background-color: #6699FF }
background-image	URL none	Фоновый рисунок	BODY { background-image: url (bg.gif) }
background-repeat	repeat repeat-x repeat-y no-repeat	Повторяемость фонового рисунка	BODY { background-image: url (bg.gif); background-repeat: repeat-y }
background-attachment	scroll fixed	Прокручиваемость фона вместе с документом	BODY { background-image: url (bg.gif); background-attachment: fixed }
background-position	Проценты Пиксели top center bottom left right	Начальное положение фонового рисунка	BODY { background-position: left top }

Таблица 9. Свойства цвета и фона

ТАБЛИЦЫ

Таблицы в HTML позволяют представлять информацию в структурированном виде. При этом не только для отображения табличных данных, но и как один из способов для создания каркаса страницы.

Правила создания таблиц следующие: таблица располагается внутри блока `<TABLE>`; сначала в таблице создается строка `<TR>`, а уже в ней ячейки `<TD>` (рис. 16).



Рисунок 16. Способ задания таблиц в HTML

При этом количество столбцов определяется количеством ячеек в первой строке и не должно меняться во всех остальных строках.

Для тега `<TABLE>` существует следующий набор свойств (таблица 10).

Атрибут	Описание
width	Ширина таблицы или ячейки, задается в процентах, или в пикселах.
height	Высота таблицы или ячейки, задается в процентах, или в пикселах.
align	Определяет выравнивание таблицы: ("center" "right" "left" "justify")
background	Задаёт фоновый рисунок таблицы: (background="images/foto.gif")
bgcolor	Цвет фона таблицы или ячейки: (bgcolor="#003399")
border	Толщина рамки в пикселах, если без рамки то задать ноль:(border="0")
bordercolor	цвет рамки: (bordercolor="#0066CC")

Таблица 10. Атрибуты тега `<TABLE>`

Для тега `<TR>` и `<TD>` существует следующий набор свойств (таблица 11).

Атрибут	Описание
align	Устанавливает горизонтальное выравнивание в ячейке: ("center" "right" "left")
valign	Устанавливает вертикальное выравнивание, по умолчанию "center" - по центру.
bgcolor	Цвет фона или ячейки: (bgcolor="#003399")
background	Задаёт фоновый рисунок строки или ячейки: (background="images/foto.gif")

Таблица 11. Атрибуты тега <TR> и <TD>

Только для тега <TD> существует следующий набор свойств (таблица 12).

Атрибут	Описание
width	Ширина ячейки, задается в процентах, или в пикселах.
height	Высота ячейки, задается в процентах, или в пикселах.
rowspan	Растягивание ячейки по строкам: <td rowspan="3"> - ячейка растягивается на три строки.
colspan	Растягивание ячейки по вертикали: <td colspan="3">- ячейка растягивается на три колонки.
background	Задаёт фоновый рисунок ячейки: (background="images/foto.gif")
nowrap	Устанавливает размещение текста в одну строку.

Таблица 12. Атрибуты тега <TD>

Для того, чтобы создать таблицу 3x3. Необходимо задать три строки <TR> в каждой из которых будет 3 ячейки <TD>.

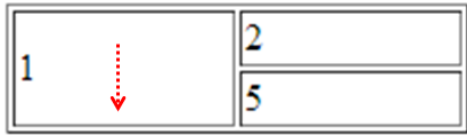
Пример:

```
<table width="200" border="1">
  <tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
    <td>6</td>
  </tr>
  <tr>
    <td>7</td>
    <td>8</td>
    <td>9</td>
  </tr>
</table>
```

1	2	3
4	5	6
7	8	9

В процессе создания таблицы нередко может возникать необходимость объединения ячеек. Для этого у тега <TD> есть два атрибута: colspan – объединение по горизонтали и rowspan – объединение по вертикали.

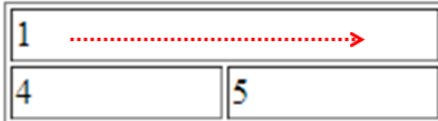
```
<table width="200" border="1">
  <tr>
    <td rowspan="2">1</td>
    <td>2</td>
  </tr>
  <tr>
    <td>5</td>
  </tr>
</table>
```



The diagram shows a table with two rows and two columns. The first column contains the number '1' in the top row, and the second column contains '2' in the top row and '5' in the bottom row. A red dotted arrow points downwards from the '1' cell, indicating that it spans two rows (rowspan="2").

Рисунок 17. Объединение ячеек по вертикали

```
table width="200" border="1">
  <tr>
    <td colspan="2">1</td>
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
  </tr>
</table>
```



The diagram shows a table with two rows and two columns. The top row is a single cell containing the number '1', with a red dotted arrow pointing to the right, indicating it spans two columns (colspan="2"). The bottom row contains the numbers '4' and '5' in the two columns respectively.

Рисунок 18. Объединение ячеек по горизонтали

Также для таблиц характерны следующие CSS свойства:

- Border-left – свойства левой границы;
- Border-right – свойства правой границы;
- Border-top – свойства верхней границы;
- Border-bottom – свойства нижней границы;
- Color – цвет текста в ячейках;
- Background-color – цвет заливки ячеек;
- Background-image – фоновое изображение;
- Text-align – горизонтальное выравнивание текста.

ФОРМЫ

Веб-форма – элемент страницы, позволяющий пользователю передавать данные на сервер для последующей обработки. Формы являются связующим звеном между клиентской и серверной частями web-приложения. Язык HTML

позволяет создавать различные элементы форм для решения задач сбора информации и передачи ее в стандартизированном формате на сервер.

На одной странице может быть сколько угодно форм. При этом НЕЛЬЗЯ помещать одну форму в другую. Форма размещается между открывающим и закрывающим тегами <FORM>.

Для тега <FORM> характерны следующие атрибуты (таблица 13):

Атрибут	Описание
action	Имя скрипта, куда передаются данные
method	Способ передачи данных GET или POST
name	Уникальное имя формы (используется в JavaScript)
accept-charset	Кодировка передаваемых данных

Таблица 13. Атрибуты тега <FORM>

При создании элементов формы важно помнить, что у КАЖДОГО элемента формы должен быть атрибут NAME, определяющий имя поля из которого поступают данные на сервер. При отсутствии данного атрибута сервер не сможет обработать значения. Значение атрибута NAME придумывается пользователем. Важно помнить, что название должно состоять из латинских букв без пробелов.

Для создания элементов формы существуют 3 тега:

1. <INPUT>
2. <SELECT>
3. <TEXTAREA>

Элемент <INPUT>

Данный элемент позволяет реализовывать ввод данных различными способами. Его внешний вид и функциональные возможности зависят от атрибута type (Таблица).

Тип	Описание	Вид
button	Кнопка.	<input type="button" value="Кнопка"/>
checkbox	Флажки. Позволяют выбрать более одного варианта из предложенных.	<input type="checkbox"/> Вода <input type="checkbox"/> Чай <input type="checkbox"/> Кофе


file	Поле для ввода имени файла, который пересылается на сервер.	
hidden	Скрытое поле. Оно никак не отображается на веб-странице.	
image	Поле с изображением. При нажатии на рисунок данные формы отправляются на сервер.	
password	Обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками. Предназначено для того, чтобы никто не подглядел вводимый пароль.	<input type="password"/>
radio	Переключатели. Используются, когда следует выбрать один вариант из нескольких предложенных.	<input type="radio"/> Вода <input type="radio"/> Чай <input type="radio"/> Кофе
reset	Кнопка для возвращения данных формы в первоначальное значение.	<input type="reset"/>
submit	Кнопка для отправки данных формы на сервер.	<input type="submit"/>
text	Текстовое поле. Предназначено для ввода символов с помощью клавиатуры.	<input type="text"/>

Таблица 14. Виды элемента <INPUT>

Помимо этого HTML5 расширяет возможности элемента <INPUT> за счет добавления новых вариантов значений атрибута TYPE (Таблица 15).

Значение type	Описание
search	Текстовое поле, предназначенное для ввода поискового запроса. Отличается от обычного текстового поля своим лексическим назначением. Некоторые браузеры отображают на нем дополнительную кнопку очистки поля.
email	Текстовое поле для ввода адресов электронной почты. По умолчанию элемент принимает только один адрес, но указав булев атрибут multiple="multiple" можно разрешить пользователю ввод нескольких адресов через запятую.

url	Текстовое поле для ввода абсолютного IRI.
tel	Поле для ввода телефонных номеров. В отличие от полей ввода почтовых адресов и URL, телефонный номер по умолчанию не проходит проверку при отправке данных, поскольку существует множество различных форматов телефонных номеров.
number	Поле числового ввода. Визуально похоже на текстовое, но с добавлением кнопок-стрелок, позволяющих увеличивать и уменьшать значение.
range	Специальный элемент для выбора значения из заданного диапазона. Представляет собой ползунок, минимальное и максимальное значения которого задаются в атрибутах min и max соответственно, а шаг — в атрибуте step.
time	Элемент для ввода времени. Похож на поле для ввода чисел, но с разделением на часы и минуты.
date	Элемент для выбора даты, представляющий собой выпадающий григорианский календарь.
week	Элемент для выбора недели. Визуально аналогичен элементу с type="date", отличается лишь формат значения.
month	Элемент для выбора месяца. Визуально аналогичен элементу с type="date", отличается лишь формат значения.
color	Специальный элемент для выбора цвета в формате RGB.

Таблица 15. Значения атрибута TYPE в HTML5

Элемент <SELECT>

В том случае, если необходимо ограничить пользователя в вводе значений, следует использовать готовый набор значений. Например, это актуально, если заранее известны названия городов, в которые осуществляется доставка. Таким образом исключается возможная ошибка при вводе значения и исключаются варианты, которые не смогут быть обработаны. Для этой цели используется элемент <SELECT> (рис.)

Рисунок 19. Варианты использования элемента <SELECT>

Атрибут `SIZE` отвечает за количество отображаемых вариантов. Если значение равно 1 – то представление в виде выпадающего списка, а если больше 1 – то поле для выбора значения с полосой прокрутки (если количество возможных значений больше значения атрибута `SIZE`).

Для добавления значений в элемент `<SELECT>` используется парный тег `<OPTION>`. При этом его атрибут `VALUE` отвечает за передаваемое на сервер значение при выборе конкретного варианта.

Пример:

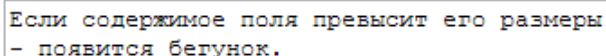
```
<form>
  <select name="s1">
    <option value="tea">Чай</option>
    <option value="coffee">Кофе</option>
    <option value="milk">Молоко</option>
    <option value="ham">Ветчина</option>
    <option value="cheese">Сыр</option>
  </select>
</form>
```

Многострочное текстовое поле <TEXTAREA>

Элемент `<TEXTAREA>` позволяет создавать многострочные текстовые поля. При этом атрибуты `COLS` и `ROWS` определяют ширину (в символах) и высоту (в строках) данного поля. Элемент является парным. При этом содержимое расположенное между открывающим и закрывающим тегом отображается в данном окне при просмотре браузером.

Пример:

`<textarea name='text' cols=20 rows=5>` Если содержимое поля превысит его – появится бегунок`</textarea>`

A screenshot of a web browser showing a text area. The text area contains the text "Если содержимое поля превысит его размеры - появится бегунок." and has a vertical scrollbar on the right side, indicating that the content is longer than the visible area.

Пример формы

```
<p> Форма заказа обучающего видеоматериала: </p>
<form action="obrabotka_form.php" method="post" name="send"
<p> Ваше имя: * </p>
<p> <input type="text" name="name" size="40"> </p>
<p> Ваш заказ: </p>
<p> <select name="spisok">
<option value="HTML"> Видео-урок по HTML </option>
<option value="CSS"> Видео-урок по CSS </option>
<option value="PHP"> Видео-урок по PHP </option>
</select> </p>
<p> Выберите носитель: </p>
<p> <input type="radio" name="disk" value="CD" checked> CD </p>
<p> <input type="radio" name="disk" value="DVD"> DVD </p>
<p> <input type="radio" name="disk" value="USB Flash"> USB Flash </p>
<p> Ваш E-mail: * </p>
<p> <input type="text" name="mail" size="40"> </p>
<p> Ваш адрес: * </p>
<p> <textarea name="adress" cols="30" rows="5"> </textarea> </p>
<p> <input type="reset" value="Сбросить"> </p>
<p> <input type="submit" value="Отправить"> </p>
</form>
```

Форма заказа обучающего видеоматериала:

Ваше имя: *

Ваш заказ:

Видео-урок по HTML ▾

Выберите носитель:

- ☒ CD
☐ DVD
☐ USB Flash

Ваш E-mail: *

Ваш адрес: *

* - поля обязательные для заполнения!

Рисунок 20. Пример HTML-формы

PHP

Скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Для выполнения скриптов PHP необходимо наличие специального программного обеспечения – web-сервера. На сегодняшний день лидером является Apache. Это web-сервер на базе операционной системы UNIX и соответственно его конфигурация осуществляется посредством редактирования конфигурационных файлов. Для удобства установки и настройки сервера Apache, а также настройки дополнительных модулей, таких как PHP, MySQL и phpMyAdmin

существует готовое программное решение – Денвер. Его можно скачать бесплатно по ссылке <http://www.denwer.ru/>. Установка Денвера не представляет значительных сложностей, поскольку проходит в режиме диалога в консольном формате. После установки Денвера будет открыто окно браузера с информацией об успешном начале работы. Для последующего запуска и остановки Денвера необходимо зайти в папку, куда он был установлен и в папке denwer воспользоваться программами Run и Stop.

Принцип выполнения php-скриптов:

1. Создать папку сайта по адресу Z:\home\localhost\www (не обязательно)
2. Сохранить созданный php – скрипт в данную папку. Файл должен иметь расширение .php
3. В браузере ввести адрес `http:localhost/имя_папки/скрипт.php`

Синтаксис PHP

Следует помнить что на странице может быть сколь угодно много php-скриптов. По сути, php-скрипт – это «умный тег», вывод которого определяется некоторым алгоритмом.

```
<?php  
  
    //Здесь располагается php-скрипт  
  
?>
```

PHP – скриптовый язык, поддерживающий базовые принципы процедурного программирования (последовательную, условную и циклическую конструкции).

Переменные в PHP

Переменные в PHP имеют следующий вид \$имя_переменной. При этом в качестве имени используются латинские буквы, цифры и знак нижнего подчеркивания. Цифра не может идти первой в названии. Определять тип переменной не надо. PHP сам определяет тип переменной и использует ее с учетом содержимого.

Математические операции

PHP поддерживает следующие операции над числовыми данными

- + сложение
- - вычитание
- * умножение
- / деление
- ++ инкремент
- -- декремент

Вывод данных на экран

Для вывода информации на экран используется конструкция **echo**

```
echo "Текст";  
  
echo $переменная;  
  
echo "Текст $переменная <тег>";
```

Следует обратить внимание, что вывод значения и переменной стандартный – текст в кавычках, переменные без кавычек. Но в echo есть возможность выводить и текст и переменные внутри кавычек. Также вывод тегов осуществляется внутри кавычек, так как для php – это обычный текст, который он должен передать браузеру.

Ветвление

Базовая конструкция «Ветвление» подразумевает выбор одного из двух вариантов – истинного и ложного (рис.).

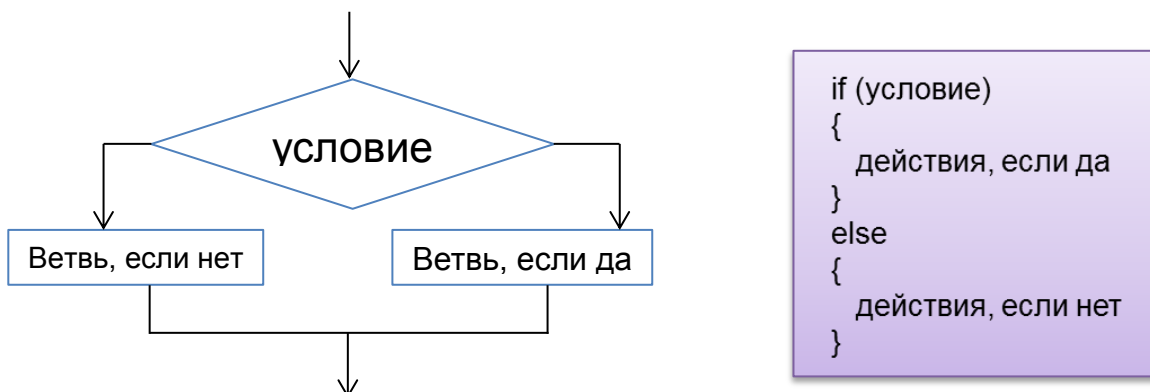


Рисунок 21. Блок-схема и синтаксис ветвления

При этом блок `else` не является обязательным, т.е. его можно не прописывать, если нет в этом необходимости.

Для построения условия необходимо применять условные операторы.

- больше
- `<` меньше
- `==` равно
- `!=` не равно
- `>=` больше либо равно
- `<=` меньше либо равно

При этом возникает возможность создавать простые условия, например `$a>3`. Однако, часто возникает необходимость составлять более сложные условия, включающие в себя несколько простых условий. Например, проверить правильность введенного логина и пароля. При этом два этих отдельных условия должны выполняться как одно. В этом случае необходимо использовать логические операторы – `&&`(И), `||`(ИЛИ).

```
<?php
$log='Вася';
$pass='123';
if (($log=='Вася') && ($pass=='123'))
{
    echo "Привет, Вася";
}
else
{
    echo "Мы не знакомы!";
}
?>
```

Оператор SWITCH

В случае необходимости множественного выбора среди заранее известного набора значений более удобным является оператор `SWITCH`, предполагающий набор блоков `CASE` для выполнения действия, если входная переменная получает конкретное значение.

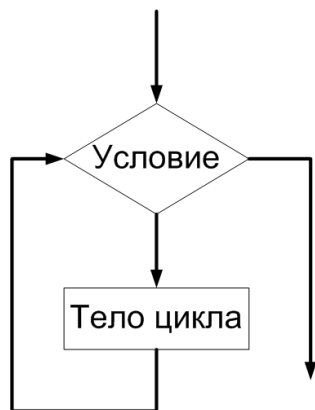
```

switch ($day){
  case 1: //выполнение при $day ==1;
    //выполнение блока кода 1;
    break; //останов
  case 2: //выполнение при $day ==2;
    //выполнение блока кода 2;
    break; //останов
  case 3: //выполнение при $day ==3;
    //выполнение блока кода 3;
    break; //останов
  default: //ничего из предыдущего не подходит;
    //выполнение блока кода 1;
    break; //останов
}
?>

```

Цикл

Данная конструкция позволяет выполнять однотипную операцию некоторое количество раз (пример – вывод строк таблицы).



Обязательные элементы цикла:

1. Начало цикла (до входа в цикл)
2. Шаг (в теле цикла)
3. Конец цикла (условие)

Рисунок 22. Блок-схема и синтаксис ветвления

Порядок работы с циклом:

1. Инициализация "счетчика";
2. Проверка условия;
3. Выполнение инструкций;
4. Изменение "счетчика".

Существуют 2 базовых оператора цикла WHILE и FOR.

Цикл WHILE

```
while(условие)
{
    Действия внутри цикла
    (тело цикла)
}
```

Действия внутри тела цикла выполняются до тех пор, пока условие истинно.

Пример:

```
<?php
    $i=1; //Начальное значение
    while($i<11) //Условие (Конечное значение)
    {
        echo "$i <br>";
        $i++; //Шаг
    }
?>
```

Результат – вывод чисел от 1 до 10

Цикл FOR

Позволяет указывать начальное значение, условие и шаг непосредственно внутри цикла.

```
for(начальное значение ; условие ; шаг)
{
    Действия внутри цикла
    (тело цикла)
}
```

При этом блоки начальное значение, условие и шаг можно не заполнять, но в обязательном порядке они должны быть разграничены точкой с запятой.

Пример:

```
<?php
for($i=1;$i<11;$i++)
{
    echo "$i <br>";
}
?>
```

Результат – вывод чисел от 1 до 10

Массивы

Массив - структура данных, хранящая набор значений, идентифицируемых по индексу или набору индексов, принимающих целые значения из некоторого заданного непрерывного диапазона (рис. 23).



Рисунок 23. Массив

Для создания массивов в PHP используется два способа:

1. Заполнение каждого элемента массива по отдельности.
Например, $\$mas[0]=4$, где $\$mas$ – имя массива, 0 – индекс элемента, 4 – значение, записанное в ячейку с индексом 0.
2. Заполнение всего массива целиком.
Например, $\$mas=array(4,3,-9)$, где $\$mas$ – имя массива, 4 – значение, записанное в ячейку с индексом 0, 3 в ячейку с индексом 1, -9 в ячейку с индексом 2.

Помимо классического подхода к организации хранения данных в массиве в php существует понятие ассоциативного массива – массива, у которого в качестве индексов используются строковые значения (рис. 24).

АССОЦИАТИВНЫЙ МАССИВ

	white	black	red	green	blue
\$name =	белый	черный	красный	зеленый	синий

Рисунок 24. Ассоциативный массив

Чтобы организовать подобный массив используется необходимо индекс указывать в одиночных кавычках. Например, `$name['white'] = "Белый"`, где *white* – индекс, а “Белый” – значение. В данном случае вывод `echo $mas['white']` выведет на экран слово **Белый**.

Для работы с ассоциативными массивами использование циклов `while` и `for` не подходит, так как индекс – не числовое значение. Для ассоциативных массивов используется цикл **foreach**.

```
foreach ($массив as $key => $value)  
{  
    Действия внутри цикла  
    (тело цикла)  
}
```

\$key – переменная, в которую помещается индекс элемента

\$value – переменная, в которую помещается значение элемента

Таким образом, отпадает необходимость определять начальное и конечное значение для цикла. Очередные значения записываются в соответствующие переменные и выполняется тело цикла. Так продолжается до последнего элемента в массиве.

Пример:

```
$names["Ритчи"] = "Деннис";  
$names["Томпсон"] = "Кен";  
$names["Гейтс"] = "Билл";  
$names["Джобс"] = "Стив";
```

```
foreach ($names as $key => $value) {  
    echo "$value $key";  
}
```

Вывод на экран:

Деннис Ритчи
Кен Томпсон
Билл Гейтс
Стив Джобс

Сортировка массивов.

Для сортировки массива используются следующие функции PHP:

1. **sort()** – сортировка массива по значениям по возрастанию;
2. **rsort()** - сортировка массива по значениям по убыванию;
3. **asort()** - сортировка ассоциативного массива по значениям по возрастанию;
4. **arsort()** - сортировка ассоциативного массива по значениям по убыванию;
5. **krsort()** - сортировка ассоциативного массива по индексам по возрастанию;
6. **krsort()** - сортировка ассоциативного массива по индексам по убыванию.

Получение данных из формы

Для получения данных из HTML-формы используется следующая конструкция:

```
$переменная= $_POST  
$_GET      ['имя_поля_формы'];  
$_REQUEST
```

Таким образом, в переменную попадают данные из поля формы.

Существуют 3 способа получить данные:

1. **\$_GET** – если в форме method=GET;
2. **\$_POST** – если в форме method=POST;
3. **\$_REQUEST** – не критичен к значению атрибута method.

MYSQL

Принципы хранения информации

При работе с любой изменяющейся системой (а сайт в первую очередь относится к таковой) необходимо где-то хранить информацию, которая определяет содержимое этой системы. В нашем случае – это наполнение сайта информацией, которую можно в режиме формы добавлять или удалять с сайта. Это актуально для форумов, гостевых книг, голосований, блогов, CMS сайта и много другого. Хранение информации производится в файлах на диске, где данные хранятся в обычном текстовом виде. Например, файлы блокнота для хранения

записей гостевой книги содержат в себе текст этой записи: «Привет, я Вася!». При таком подходе возникает ряд неудобств: во-первых, размер файла ограничен; во-вторых, при создании нескольких файлов требуется соблюсти уникальность их имен (т.к. не могут существовать два файла с одинаковыми именами в одной папке); в-третьих, быстродействие работы сайта на файлах очень низкое, за счет сложностей чтения-записи и сортировки данных. Ну и наконец – главный недостаток: НЕВОЗМОЖНОСТЬ производить поиск и выборку нужной информации из всего объема данных. Таким образом, использование хранилища данных в виде файлов – невыгодный сложный процесс.

Альтернативным подходом работы с файлами являются базы данных, состоящие из таблиц. Таблицы в свою очередь хранят в структурированном виде (в виде двумерного массива) данные, которые возможно использовать для работы. Таким образом, исключаются все недостатки, присутствующие при работе с файлами.

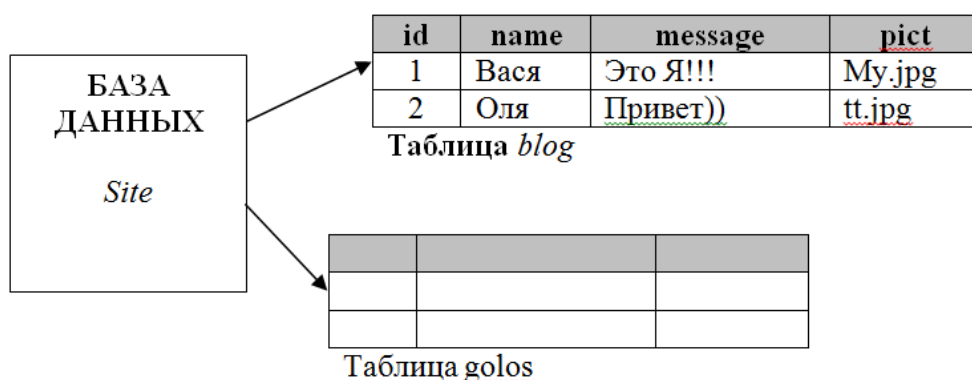


Рисунок 25. Организация хранения данных с использованием БД

В данном примере (рис.25) существует база данных сайта с названием Site, состоящая из 2х таблиц: blog и golos, которые в свою очередь хранят информацию о странице сайта Блог и голосовании на сайте. В таблице blog содержатся реальные записи, которые в последствии переносятся на сайт в оформленном виде. Каждый пользователь может добавлять информацию о себе: Имя, Сообщение, Фотографию (Name, Message, Pict), которая будет формировать новую запись в таблице (новую строку) с уникальным (следующим по номеру) идентификатором ID. Идентификатор присутствует во всех таблицах с целью сохранения целостности данных. Например, если два человека внесут в базу абсолютно одинаковые данные, то строки все равно будут отличаться по идентификатору. В этом случае не произойдет путаницы при поиске данных в таблице. На основании сформированных таблиц заполняются страницы сайта.

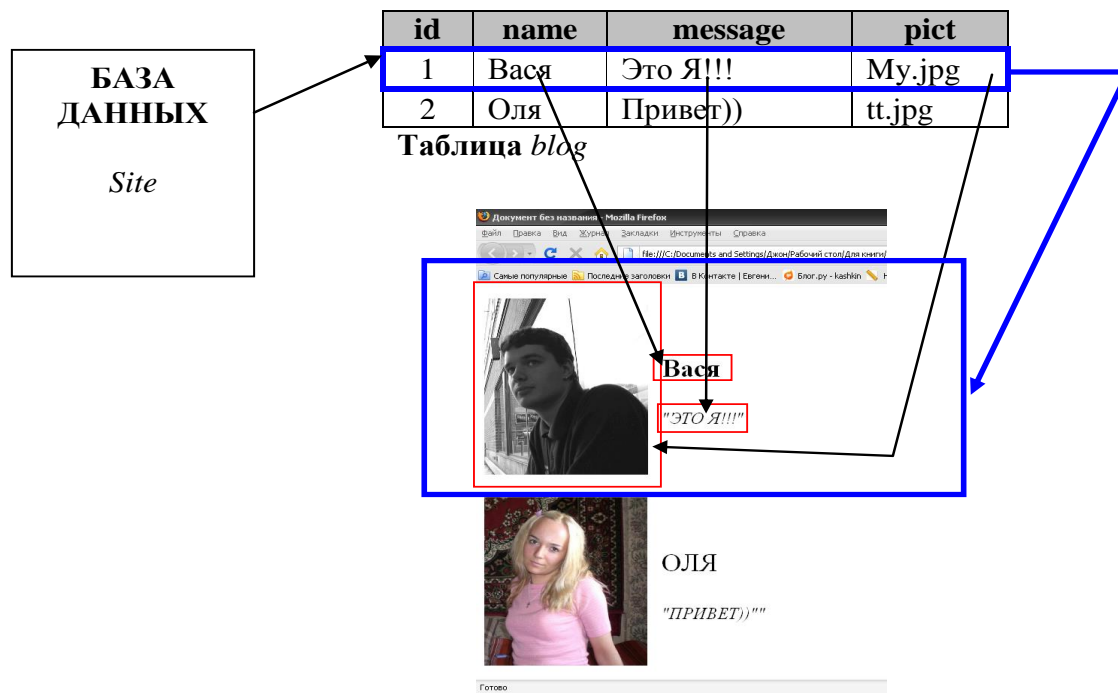


Рисунок 26. Использование данных из БД для формирования web-страницы

На рисунке 26 проиллюстрировано, каким образом данные из таблицы формируют страницу сайта. Из записи с ID = 1 формируется первая запись вида (рис. 27):

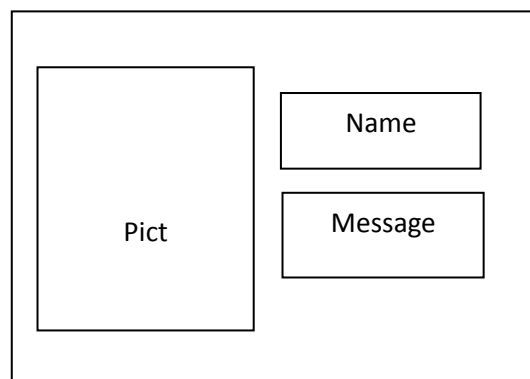


Рисунок 27. Шаблон вывода данных на web-страницу

При этом соответствующие значения полей таблицы *blog* подставляются в соответствующие места в шаблон, написанный на HTML в виде строки таблицы (`<tr>...</tr>`). Если в таблице БД содержится несколько записей, то они выводятся в цикле от ID = 1 до последнего (пока ID = true, т.е. пока ID существует). Алгоритм приведенного примера в общем виде выглядит так (рис. 28):

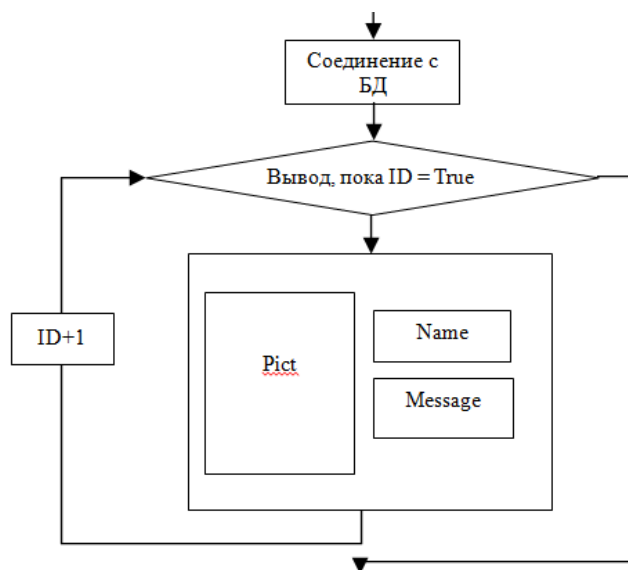


Рисунок 28. Алгоритм вывода данных на web-страницу

Создание БД

Базы данных MySQL хранятся на MySQL-сервере – специальной службе, устанавливаемой на компьютер (в случае локального использования), или покупаемую на хостинге. MySQL-сервер для работы с БД MySQL также необходим, как для работы с PHP необходим Apache! При покупке хостинга указывается не только версия установленного на ней PHP, но и поддерживаемая версия MySQL-сервера с максимально возможным количеством используемых баз. Чем выше стоимость хостинга, тем больше различных БД можно разместить на сервере. Исходя из вышесказанного следует помнить, что при покупке хостинга следует обращать особое внимание на максимальное количество размещаемых БД, а при локальной работе – установлен ли MySQL-сервер. Если вы использовали для установки пакет Денвер, то беспокоиться не о чем – он автоматически устанавливает как PHP, так и MySQL. В случае, если вы устанавливали Apache, то не забудьте дополнительно установить MySQL-сервер.

Для входа в MySQL-сервер необходимо в строке URL браузера ввести **localhost/tools/phpmyadmin** после чего загрузится непосредственно окно MySQL (рис. 29).

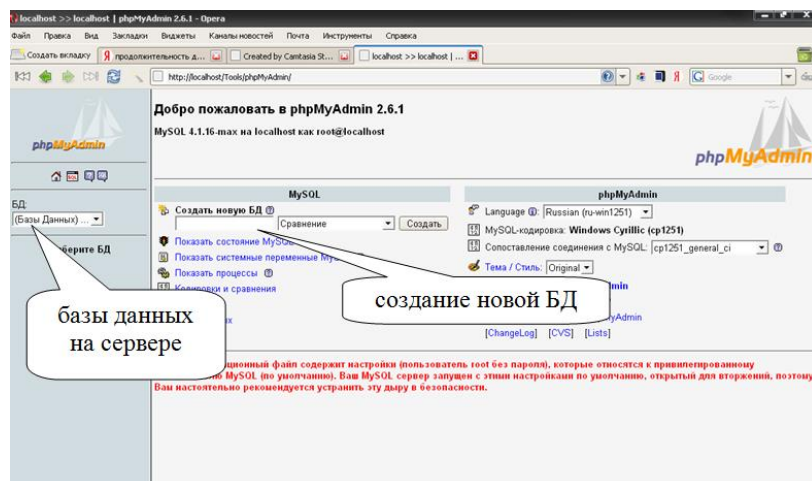


Рисунок 29. Интерфейс phpMyAdmin

В левой части окна можно увидеть в раскрывающемся списке все расположенные на сервере БД. Выбрав одну из них появляется возможность работы с ее структурой и наполнением. Чтобы создать новую БД необходимо в поле ввести имя новой базы и нажать кнопку **Создать**. После этой операции появится окно (рис. 30), в котором высветится результат выполнения операции.

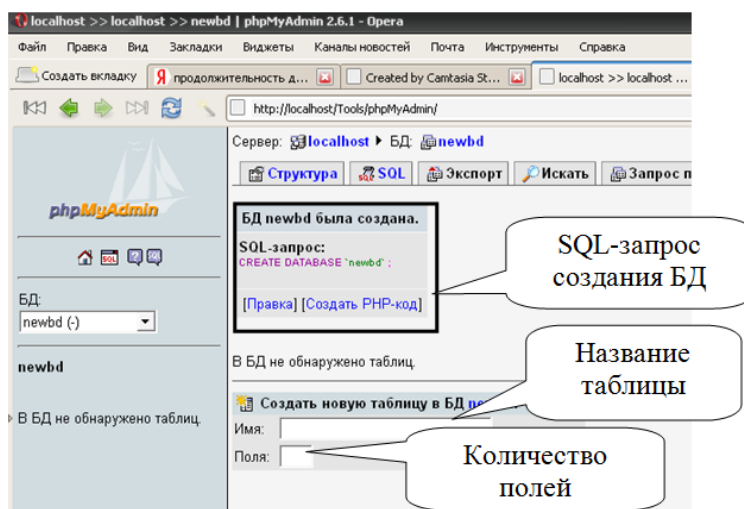


Рисунок 30. Создание таблицы БД

Следующий шаг – определение количества полей 1й таблицы только что созданной БД. Для примера используем таблицу blog, в которой 3 рабочих поля и поле ID – уникальное поле с изменяющимся инкрементно (увеличивается на 1 при каждой новой записи). Итого – 4 поля (рис. 31).

1	2	3	4
id	name	message	pict
1	Вася	Это Я!!!	My.jpg
2	Оля	Привет))	tt.jpg

Таблица *blog*

Рисунок 31. Пример таблицы *blog*

Чтобы создать такую таблицу (см. рис. 30) необходимо в поле **Имя** ввести имя новой таблицы *blog*, а в поле **Поля** количество полей таблицы 4. Далее следует нажать кнопку Пошел. В результате появится окно (рис. 32).

Сервер: localhost БД: newbd таблица: blog

Поле	Тип	Длины/Значения	Сравнение	Атрибуты	Ноль	По умолчанию	Дополнительно
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		
	VARCHAR				not null		

Комментарий к таблице: Тип таблицы: По умолчанию Сравнение:

Add 1 field(s) Пошел Сохранить

* Для типов поля "enum" и "set", введите значения по этому формату: 'a','b','c'...
Если вам понадобится ввести обратную косую черту ("\") или одиночную кавычку (") среди этих значений, поставьте перед ними обратную косую черту (например, '\xyz' или 'a\b').

** Для значений по умолчанию просто введите единственное значение без экранирования и кавычек, используя этот формат: a

*** Пожалуйста, введите значения для трансформации, используя этот формат: 'a','b','c'...
Если вам нужно поставить бэкслэш("\") или кавычку("), экранируйте их (например '\xyz' or 'a\b').
Для списка доступных опций трансформации и трансформаций их MIME-типов кликните на [описание трансформаций](#)

Рисунок 32. Интерфейс определения типов полей таблицы

В столбец **Поле** вводятся имена полей таблицы (id, name, message, pict). В столбце **Тип** устанавливается тип хранимых данных этого поля.

Существует 4 базовых типа данных:

1. INT – целочисленный тип;
2. VARCHAR – строка символов;
3. TEXT – многострочный текст;
4. DATE – дата в стандартном формате.

Поле **Длины/Значения** определяет сколько знаков будет в данном поле. Например, если поле числовое и максимальное число 999999, то количество знаков – 6, а в VARCHAR – значения считаются посимвольно. Для типов DATE и TEXT длины не устанавливаются.

Для ключевого поля (id) устанавливают A_I – авто инкремент, чтобы не заполнять это поле вручную и PRIMARY – чтобы определить его как ключевое.

Для соединения базы данных и PHP-скрипта используется следующий алгоритм:

1. Соединение с SQL-сервером
\$db=mysql_connect(“хост”,”логин”,”пароль”);
по умолчанию: хост - localhost, логин – root, пароль отсутствует
2. Выбор БД на сервере
mysql_select_db(“БД”, \$db);
\$db – указатель на сервер, полученный в предыдущем пункте
3. Установка кодировки
mysql_set_charset(“utf8”);
4. Запрос к таблице БД
\$rez=mysql_query(“SQL-запрос”);
\$rez – абстрактная таблица, содержащая результаты запроса
5. Преобразование абстрактной таблицы в ассоциативный массив
\$mas=mysql_fetch_array(\$rez);
\$mas – ассоциативный массив, где индексы – названия полей таблицы, а значения – значения текущей строки. Данная функция выбирает построчно записи и записывает их в ассоциативный массив. При этом каждый новый вызов функции осуществляет переход к следующей строке.

SQL-запросы

Основной целью любой базы данных является работа с данными: их поиск, представление, удаление, изменение и добавление. Таким образом следует учитывать, что при работе с БД необходимо грамотно формировать к ней запросы для получения требуемого результата

Существует 4 базовых типа запросов SQL для работы web-приложений:

1. **SELECT** – выбор данных;
2. **INSERT** – добавление данных;
3. **UPDATE** – изменение данных;
4. **DELETE** – удаление данных.

Запрос выбора данных

**SELECT * FROM таблица WHERE поле=’значение’ AND/OR
поле=’значение’ ORDER BY поле DESC LIMIT число**

1. **SELECT * FROM таблица** – обязательная часть запроса. * - обозначает выбрать все поля, либо необходимо через запятую указать поля, попадающие в запрос. Таблица – название таблицы из которой выбираются данные.

2. **WHERE поле='значение'** – необязательная часть, используется для фильтрации выбираемых данных. Если фильтров несколько, то они указываются через **AND** – логическое «И», **OR** – логическое «ИЛИ».
3. **ORDER BY поле** – сортировка выбранных данных по полю по возрастанию. Добавление конструкции **DESC** определяет сортировку по убыванию.
4. **LIMIT число** – ограничивает количество записей попадающих в результат выборки.

Запрос добавления данных

INSETR INTO таблица(поле1, поле 2, ..., поле N)
VALUES('значение1', 'значение2',..., 'значениеN')

Таблица - таблица, в которую добавляются данные. В скобках указываются поля, в которые будут добавлены данные. **VALUES** определяет, какие данные будут добавлены. Так, значение1 будет добавлено в поле1, значение2 в поле2 и т.д. Количество полей и значений должно совпадать по количеству.

Запрос обновления данных

UPDATE таблица **SET** поле1='значение1, поле2='значение2,...,
полеN='значениеN **WHERE** поле='значение'

Таблица - таблица, в которой обновляются данные. После конструкции **SET** располагается группа **поле1='значение1**, в которой прописываются изменения. **ОБЯЗАТЕЛЬНО** необходимо указывать условие обновления **WHERE**, иначе данные изменятся **ВО ВСЕХ СТРОКАХ** таблицы!

Запрос удаления данных

DELETE FROM таблица **WHERE** поле='значение'

Таблица - таблица, в которой удаляются данные. **ОБЯЗАТЕЛЬНО** необходимо указывать условие обновления **WHERE**, иначе данные удалятся **ВО ВСЕЙ ТАБЛИЦЕ!**

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ HTML, CSS, PHP, MYSQL

Данный раздел предназначен для обобщения знаний по разработке web-сайтов на основе технологий разметки страницы сайта на языке HTML, применении каскадных таблиц стилей CSS и создании скриптов на языке PHP для упрощения администрирования сайта. Также при разработке применяются базы данных MySQL для хранения содержимого страниц. Раздел представляет из себя пошаговую разработку сайта с момента создания каркаса до создания администраторской зоны. Теоретическую информацию по работе с технологиями, использующимися в данном курсе можно найти в предыдущих разделах данного пособия, посвященных изучению конкретных языков.

Предлагается разработать сайт информационного направления для представления информации о товарах и услугах. Сайт будет иметь вид, представленный на рис. 33.

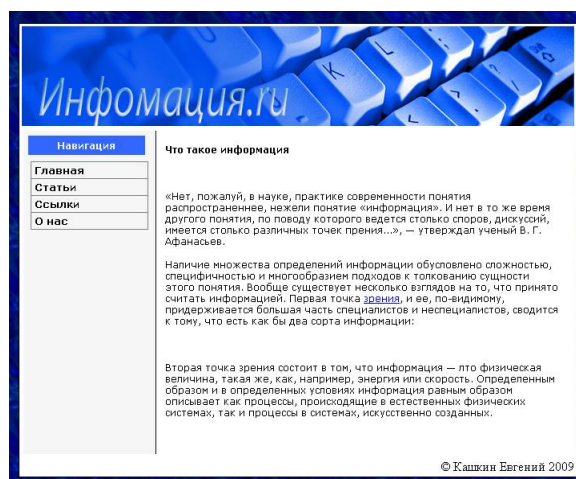


Рисунок 33. Общий вид разрабатываемого проекта

Создание каркаса сайта и подключение таблицы CSS-стилей

1. В начале создадим папку для нашего сайта в папке www Денвера (C:\WebServers\home\localhost\www). Например, *INTSITE*;
2. Создайте первый файл сайта – **index.php**. Укажите для атрибута **title** заголовков сайта «Главная страница»;
3. Настройте кодировку страницы **UTF-8 с BOM**.
4. Далее создайте файл CSS стилей. И сохраните его в папке сайта под именем **style.css**

5. Подключите его к странице **index.php**. Для этого переходим на страницу `index.php` и в разделе **head** добавляем тег `<link>` в котором указываем ее месторасположение.
6. Создайте таблицу на основной странице с количеством столбцов 1 и строк 3. Ширина таблицы 690 пикселей.
7. Выроните ее по центру (`align='center'`) и установите в качестве фонового белый цвет (`#FFFFFF`);
8. На следующем этапе необходимо создать обрамление таблицы в черном цвете. Для этого воспользуемся стилями CSS. В файле **style.css** создайте класс:

.main_border { border: 1px solid #000000;}

Где: **1px** – толщина рамки, **solid** – сплошная, **#000000** – цвет черный.

9. Подключите стиль к таблице. Для этого у тега `<table>` в атрибуте `class` укажит **main_border**;
10. Установите стиль для абзацев сайта. В таблице стилей укажите настройки тега `<p>`:

```
p {  
  
    font-family: Verdana;  
    font-size: 12px;  
    margin: 15px;  
  
}
```

11. Далее создайте шапку сайта с шириной 690 пикселей. И сохраните ее в формате `jpeg` в папку сайта `\img`.
12. Установите изображение в верхнюю строку. Для этого используйте тег ``
13. Заполните в нижней строке таблицы информацию о разработчике;
14. В средней строке таблицы создайте таблицу в 1 строку и 2 столбца;
15. Установите серый цвет левой ячейки и выделите у левой ячейки правую границу через `css` стили:

```
.left { background-color: #F6F6F6;
border-right:1px;
}
```

16.Выберите ячейку через тег <td> и установите стиль к ячейке;

17.Установите в CSS для BODY в качестве фона графический файл:

```
body {background-image:url(img/имя_файла_фона);}
```

18.Заполните главную страницу текстом, предварительно установив выравнивание в ячейке по верхнему краю (valign='top');

19.Создайте меню на основе приведенных кодов:

HTML код

```
<p align="center" class="title">Навигация</p>
<div id="coolmenu">
<a href="index.php">Главная</a>
<a href="articles.php">Статьи</a>
<a href="lessons.php">Уроки</a>
<a href="contacts.php">О нас</a>
</div>
```

CSS код

```
p.title {
background-color: #A72E37;
color: #ffffff;
font-weight:bold;
margin:5px;
padding:5px;
}
```

```
#coolmenu{
border: 1px solid gray; /*Стиль рамки */
```

```

border-bottom-width: 0;
width: 93%; /*Ширина меню */
background-color: #f6f6f6; /*Фоновый цвет ячейки*/
margin: 8px; /*Ширина отступов*/
}

* html #coolmenu{
width: 150px; /*Ширина меню для Internet Explorer*/
}

#coolmenu a{
font: bold 13px Verdana; /*Шрифт текста*/
padding: 2px; /*Внутренний отступ текста ссылки от краев ячейки*/
padding-left: 4px; /*Внутренний отступ текста ссылки от левого края ячейки*/
display: block;
width: 100%; /*Ширина ячейки*/
color: #000000; /*Цвет текста*/
text-decoration: none; /*Подчеркивание у ссылок - нет*/
border-bottom: 1px solid gray;
}

html>body #coolmenu a{
width: auto;
}

#coolmenu a:hover{
background-color: #cccccc; /*Фоновый цвет ячейки при наведение курсора*/
color: #000000; /*Цвет текста при наведении курсора*/
}

```

Разделение сайта на блоки

Для того, чтобы администрирование сайта было удобным, следует разбить имеющийся сайт на блоки, каждый из которых будет храниться в отдельном файле. Через конструкцию `include` файлы будут подключаться к страницам сайта. Таким образом изменение всего в одном файле приведет к изменению на всех страницах сайта. Предлагается организовать следующую структуру папок сайта (рис. 34):

IMG	папка с изображениями
index.php	главная страница
stat.php	страница Статьи
link.php	страница Ссылки

Рисунок 34. Структура папок сайта

Имеющийся каркас сайта предлагается разбить на 3 основных блока – заголовков (**header**), меню и нижнюю часть (**footer**) (Рис. 35). При таком подходе все элементы страницы, общие на всем пространстве сайта хранятся в отдельных файлах.

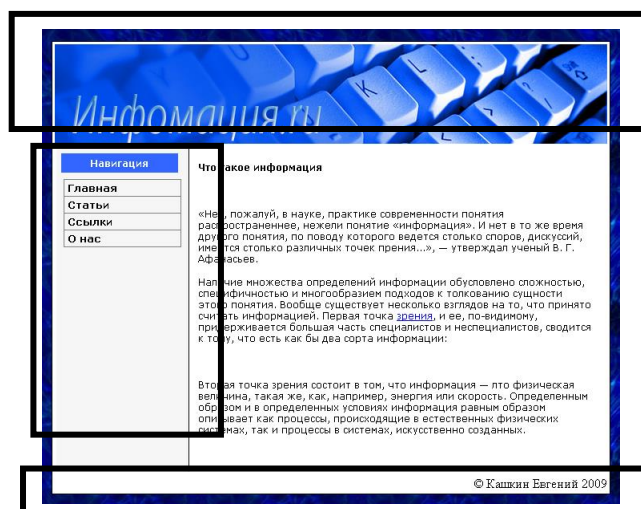


Рисунок 35. Блоки сайта

1. В папке сайта создайте новую папку для блоков (blocks);
2. На странице index.php вырежете первую строку таблицы и содержимое поместите в пустой файл blocks/header.php (Рис. 36);

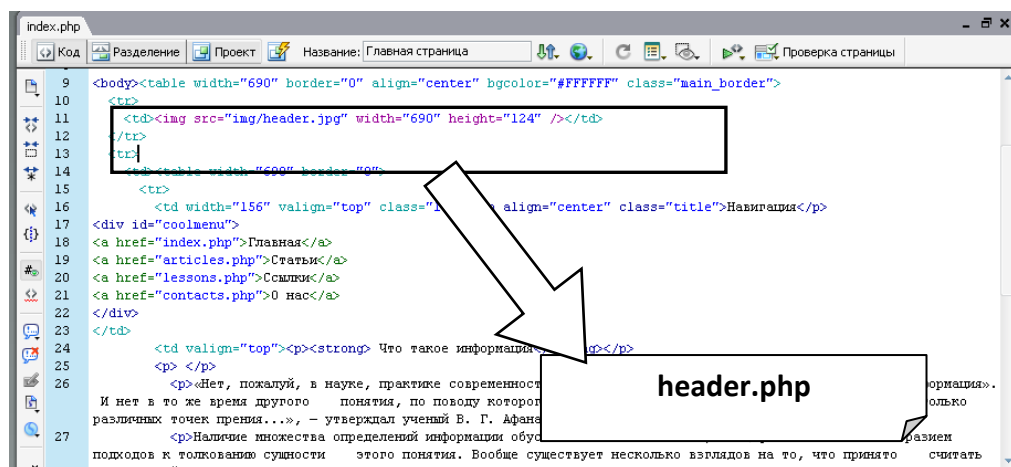


Рисунок 36. Выделение HEADER из страницы в отдельный файл

3. Сохраните файл `header.php`;
4. В файле `index.php` на месте вырезанного участка кода вставьте следующий код:

`<?php include ("blocks/header.php")?>`

Где `header.php` – файл с кодом, который надо подставить в это место основного кода, находящийся в папке `blocks`

5. Аналогичным образом поступите со всеми остальными блоками;
6. Для удобства дальнейшего корректирования кода удобно добавлять комментарии. Они не обрабатываются браузером, а программисту подсказывают, за что отвечает участок кода. В HTML комментарии находятся между `<!--Тест комментария -->`, а в PHP `/* Тест комментария */`. Добавьте комментарии к каждому `include` в коде главной страницы.
7. Создайте остальные страницы сайта через Файл > Сохранить Как главной страницы. Заполните полученные страницы текстом.

Создание БД и получение из нее информации

Для автоматизации работы с информационной частью сайта, а также для возможности создать систему управления сайтом необходимо для хранения информации использовать БД. Чтобы обратиться к СУБД `phpmyadmin` необходимо в адресной строке браузера набрать **`http://localhost/phpMyAdmin/`**.

1. Создайте БД с именем **`intsite`** (в качестве имени используйте имя сайта) на главной странице в поле **Создать новую БД**;
2. Первой таблицей нашей базы будет таблица **`settings`**, хранящая настройки страницы и их содержимое (**`id`**, заголовок страницы (**`title`**), название страницы (**`pages`**), мета-теги для описания страницы и ключевых слов поиска (**`meta_opis`**, **`meta_key`**), непосредственно текст страницы (**`text`**)) – 6 полей;
3. Для поля **`id`** следует установить **`auto_increment`** и сделать его ключевым, тип - **`int`**; поле `text` имеет тип **`text`**; остальные поля тип – **`varchar`**, длина – 255;
4. Заполните базу информацией о страницах; ВНИМАНИЕ! В поле `text` текст должен заноситься ВМЕСТЕ с тегами форматирования!
5. Далее в `PhpMyAdmin` создайте пользователя в разделе Привилегии на главной странице;
6. После того как БД создана и заполнена самое время начать выводить из нее информацию на сайт. Для этого перейдите на страницу `index.php` и

ПЕРЕД ВСЕМ КОДОМ добавьте код подключения к SQL-серверу и выбора БД.

```
<?php  
  
$db=mysql_connect("localhost","z","1234");  
  
mysql_select_db("intsite",$db);  
  
mysql_set_charset("utf8");  
  
?>
```

Где z – имя пользователя, 1234 – пароль, intsite - имя БД.

7. Удобно вынести эту часть кода в отдельный файл, а затем подключить его через конструкцию include. Таким образом экономится время при подключении к БД, т.к. достаточно написать лишь строчку include(путь к файлу с кодом подключения), взамен всего вышеуказанного кода. Да и при изменении любой информации в нем (имя пользователя, пароль) легко поменять ее в одном месте, нежели на всех страницах сайта. Не забудьте, что когда будете писать конструкцию include в коде странице её необходимо будет заключить в <?php ?>, т.к. иначе ее не поймет HTML;
8. Далее информацию из БД стандартными средствами надо выбрать в массив. Для работы нам понадобятся поля **title, meta_opis, meta_key, text**. Для того, чтобы на конкретной странице отображалась информация, характерная именно для нее, выбирать будем не все записи, а лишь ту, которая соответствует имени данной страницы:

```
SELECT title, meta_opis, meta_key, text FROM settings WHERE  
page='index'
```

Полностью команда выглядит так:

```
$rez=mysql_query("SELECT title, meta_opis, meta_key, text FROM settings  
WHERE page='index' ");
```

Где \$rez – переменная, в которую помещается результат выполнения запроса – некоторая таблица

Таким образом будет выбрана лишь информация для страницы index. Понятно, что таким же образом информация выбирается и для других страниц, меняется лишь название страницы после слова pages (**pages=имя страницы**);

9. После получения информации ее следует преобразовать в ассоциативный массив:

```
$mas=mysql_fetch_array($rez);
```

Где \$mas – имя массива, \$rez – результат выборки из БД

10. Теперь необходимо вывести информацию из массива в соответствующие места в HTML коде. Для этого следует писать конструкцию вида:

```
<?php echo $mas['поле'] ?>
```

Создание раздела «Статьи»

После того, как завершено создание базовой части сайта самое время перейти к созданию таблиц БД и заполнению страницы «Статьи» нашего сайта. Предполагается вариант, когда на странице отображается краткая информация о статьях: название (которое будет в дальнейшем переадресовывать пользователя на соответствующую статью), краткого содержания статьи и информации об авторе и дате создания.

На начальном этапе следует создать в **PhpMyAdmin** таблицу для хранения соответствующей информации.

1. Создадим в БД нашего сайта таблицу **stat**, содержащую поля **title** (название), **date** (дата создания), **opis** (описание), **text** (непосредственно текст статьи) и **author** (автор статьи). Плюс к этому понадобится поле **id** – ключевое поле таблицы. Поле id помимо всего прочего будет еще выполнять и роль связующего поля, по которому мы будем переходить на страницу с полной информацией о нужной статье. Итого – 6 полей (рис. 37)

Поле	Тип	Длины/Значения*
id	INT	4
title	VARCHAR	255
date	DATE	
opis	TEXT	255
text	TEXT	
author	VARCHAR	255

Рисунок 37. Структура таблицы «Статьи»

- Следующий этап – создание структуры папок для имеющихся статей сайта. Для этого создаем в папке сайта структуру вида:

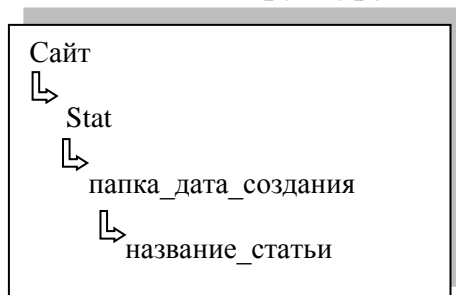


Рисунок 38. Структура каталогов раздела «Статьи»

В папке с названием статьи хранятся все изображения нашей статьи. Таким образом очень просто производить дальнейшее администрирование сайта, т.к. все упорядочено.

- Далее нужно создать шаблон страницы **Статьи**, которую мы будем заполнять информацией. Для этого создайте копию страницы **stat.php** и назовите её **stat_temp.php**. Далее заполните ее информацией и добавьте в нужные места изображения из соответствующей папки. При необходимости создайте соответствующие стили с отступами для заголовков соответствующего уровня в таблице стилей **style.css**;
- Заполните информацией из шаблона таблицу **stat** в БД. Не забудьте внести информацию об авторе, названии и т.д.;
- На странице **stat.php** создайте таблицу в 2 строки и 1 столбец для представления краткой информации о статьях. Выводите ее по центру;
- В таблице стилей **style.css** создайте стиль для созданной таблицы. Граница – серая, толщиной 1px, ширина 95%:

```
.stat {
    border:1px solid #CCCCCC;
    width:95%;
}
```

7. Примените стиль к таблице;
8. Для верхней ячейки создайте дополнительный стиль фоновой цвета (светло-серого) и примените его к ячейке:

```
.stat_head
{ background-color:#CCCCCC;}
```

9. Организуйте вывод из БД информации в таблицу страницы stat.php. Для этого в самом начале кода напишите команду подключения к БД (код был ранее написан и хранится в файле db.php) и далее создайте запрос к таблице:

```
<?php
include("db.php");
$rez=mysql_query("SELECT id,title,date,opis,author FROM stat");
?>
```

10. Теперь необходимо организовать вывод информации из БД через цикл. При этом каждый раз должна формироваться новая таблица, содержащая нужную информацию о статье. Для этого поместите таблицу (2 строки x 1 столбец) внутрь цикла while, условием которого будет вывод из БД, пока это возможно:

```
while ($mas=mysql_fetch_array($rez))
{
    Код таблицы
}
```

Здесь условием цикла является выбор очередной строки из абстрактной таблицы \$rez в ассоциативный массив \$mas пока это возможно делать, т.е. пока не конец записей

11. На следующем шаге следует произвести вывод в таблицу соответствующих значений из массива через конструкцию **printf()**. Предварительно

замените в коде таблицы все двойные кавычки на одинарные, иначе PHP выведет информацию об ошибке, т.к. для него двойные кавычки — окончание действия параметра оператора:

```
<?php
while ($mas=mysql_fetch_array($rez))
{
    printf("
        <table align='center' class='stat'>
        <tr class='stat_head'>
            <td><p>%s</p><p>Автор:%s </p><p>Дата
                добавления:%s</p></td>
        </tr>
        <tr>
            <td><p>%s</p></td>
        </tr>
        </table>",
        $mas['title'],$mas['author'],$mas['date'],$mas['opis']);
    }
?>
```

12. После это следует создать из заголовка статьи ссылку, которая будет указывать на страницу **stat_view.php**, на которой статья будет полностью отображаться. Для этого поместите заголовок статьи в тег **<a>**, в качестве ссылки указав **stat_view.php**. Для того, чтобы на странице **stat_view.php** отображалась не информация обо всех статьях, а лишь о нужно, необходимо в качестве параметра передать этой странице вместе ссылкой **id**, по которому нужно будет найти страницу в БД и вывести. Параметры вместе с ссылкой передаются через знак **?**. Таким образом в общем виде ссылка будет выглядеть так:

```
<?php
while ($mas=mysql_fetch_array($rez))
{
    printf("
        <table align='center' class='stat'>
        <tr class='stat_head'>
```

```

        <td>
        <p><a href='stat_view.php?id=%s'>%s</a>%s</p>
        <p>Автор:%s </p><p>Дата
        добавления:%s</p></td>
    </tr>
    <tr>
        <td><p>%s</p></td>
    </tr>
</table>",

```

```

$mas['id'],$mas['title'],$mas['author'],$mas['date'],$mas['opis']);
    }
?>

```

Здесь внутри таблицы была добавлена ссылка, а в качестве параметра указано значение текущего id записи таблицы (\$mas['id']).

13. Для более приятного визуального отображения названия статьи и информации о ней создадим стили для названия (stat_name) и для информации (stat_dob):

```

.stat_name
{ font-weight:bold; /*Жирный шрифт*/
  margin-top:7px;
  margin-bottom:7px;
}

.stat_dob
{ font-size:11px;
  margin-top:7px;
  margin-bottom:7px;
  color:#777777;
}

```

И подключим их к соответствующим элементам таблицы, добавив к параметрам тега <p> class='имя_класса';

14. После создания страницы с краткой информацией о статьях следует создать страницу, на которой будет отображаться полная информация об интересующей пользователя статье. Для этого мы будем использовать файл **stat_view.php**, который необходимо создать в корневой папке сайта через копирование файла **index.php**;

15. Для того, чтобы выводить на этой странице текст нужной нам статьи, необходимо выбрать ее из базы. Для этого мы будем использовать индекс (**id**) этой статьи, который мы передали вместе с ссылкой из файла с краткой информацией (**a href='stat_view.php?id=%s'**). Чтобы возможно было использовать эту переменную в самом верху кода страницы допишем код преобразования полученной по методу **GET** информации в переменную:

```
$id=$_GET['id'];
```

16. Чтобы выбрать информацию из БД для статьи следует преобразовать запрос, который остался от файла `index.php`. Запрос должен в переменную `rez` выбирать все поля по указанному `id`:

```
SELECT * FROM stat WHERE id='$id'
```

17. Далее через преобразованный в массив результат запроса выведите в нужных местах страницы:

```
<td valign="top"><?php echo "<p>", $mas['title'], "</p> <p> Дата публика-  
ции:", $mas['date'], "</p> <p>Автор: ", $mas['author']</p>,$mas['text']; ?>  
</td>
```

18. Для более приятного визуального ощущения создадим стили для заголовка и для даты с автором:

```
.view_title {font-size:16px;  
             font-weight:bold;  
             color:#3300FF;  
             text-align:center;  
             }
```

```
.view_date {font-size:11px;  
            color:#777777;  
            margin-top:7px;  
            margin-bottom:7px;  
            font-weight:bold;  
            }
```

И подключим его на странице `stat_view.php` к соответствующим тегам `<p>` - заголовка, даты и автора. Таким образом итоговый код будет выглядеть так:

```
<td valign="top"><?php echo "<p class='view_title'>", $mas['title'],  
"</p> <p class='view_date'> Дата публикации: ", $mas['date'], "</p>  
<p class='view_date'>Автор: ", $mas['author'], $mas['text']; ?> </td>
```

Разработка зоны администрирования (Адмика)

Для полноценной работы с сайтом заказчику необходимо иметь возможность управлять содержимым сайт – добавлять, удалять и изменять информацию на нем. Использовать для этого стандартные средства – PhpMyAdmin и редактор кода в данном случае не представляется возможными, поскольку для работы с ними надо изучить множество книг, а заказчику это ни к чему. Таким образом, следует создать некоторую форму на сайте - админку, через которую владелец сайта (или модератор) сможет управлять контентом (содержимым) страниц сайта. Само собой, что давать любому пользователю доступ к админке недопустимо. Для этого необходимо разработать технологию авторизации (входа через логин и пароль) в эту зону.

1. На первом этапе создадим папку `admin` в корневой папке сайта и скопируем в нее:

- файл `index.php`
- файл `style.css`
- папку `blocks`
- папку `img`

Теперь вся работа будет проходить в папке `admin`!

2. Из вновь скопированного файла **`index.php`** следует удалить соединение с БД, т.к. оно нам не понадобится. Также удалите мета-теги и вывод содержимого страницы. В качестве **`title`** страницы напишите «Админка»;
3. Теперь необходимо изменить меню, поскольку навигация по страницам здесь ни к чему следует преобразовать имена страниц в действия **Добавить, Редактировать, Удалить**. А вместо слова **Навигация** написать

раздел, с которым предполагается работа. Чтобы охватить все разделы достаточно лишь скопировать код навигации несколько раз. Ссылки пока что менять не следует – это мы сделаем позже.

ПОМНИТЕ, чтобы изменить меню необходимо менять его в папке blocks папки admin. За меню отвечает файл menu.php, который через include подключается к файлу index.php!

```
<td width="156" valign="top" class="left">
  <p align="center" class="title">Статьи</p>
  <div id="coolmenu">
    <a href="index.php">Добавить</a>
    <a href="articles.php">Удалить</a>
    <a href="lessons.php">Редактировать</a>
  </div>
  <div>
    <p align="center" class="title">Ссылки</p>
    <div id="coolmenu">
      <a href="index.php">Добавить</a>
      <a href="articles.php">Удалить</a>
      <a href="lessons.php">Редактировать</a>
    </div>
    <div>
      <p align="center" class="title">Тексты страниц</p>
      <div id="coolmenu">
        <a href="lessons.php">Редактировать</a>
      </div>
    </div>
  </td>
```

Общий вид меню на сайте будет такой (рис. 39):

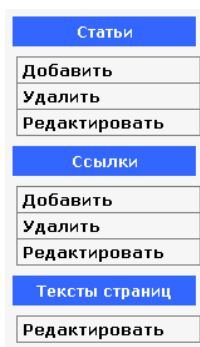


Рисунок 39. Меню зоны администрирования

Добавление новой статьи

Форма

1. Для добавления статей на сайт необходимо организовать форму, через которую эта информация будет попадать в БД. Создадим новый файл **new_stat.php** из файла **index.php**. В меню изменим соответствующую ссылку на **new_stat.php**;
2. Создадим на странице **new_stat.php** форму вида (Рис.40):

Добавление новой статьи

Название статьи

Дата создания

Краткое содержание статьи (с тегами)

Текст статьи (с тегами)

Автор статьи

Рисунок 40. Форма добавления новой статьи

Для создания элементов формы используйте следующие настройки:

Форма: Action add_stat.php, method POST;

Название статьи: title;

Дата создания: date, исходное значение 2019-02-17. Это необходимо для того, чтобы пользователь не ошибся при вводе даты. Здесь ему будет достаточно изменить уже имеющийся текст;

Краткое описание статьи: opis;

Текст статьи: text;

Автор: author;

Длина элементов text 60 символов, а textarea – 57.

Обработчик

1. Чтобы создать файл обработчика (**add_stat.php**) воспользуемся файлом **new_stat.php**, сохранив его с новым именем;
2. Для добавления урока нам понадобится соединяться с БД, поэтому добавьте код подключения к БД (**через include**) в самый верх кода

```
<?php include("blocks/db.php"); ?>
```

3. Теперь необходимо преобразовать данные из формы в переменные:

```
$title=$_REQUEST['title'];  
$date=$_REQUEST['date'];  
$opis=$_REQUEST['opis'];  
$text=$_REQUEST['text'];  
$author=$_REQUEST['author'];
```

4. Теперь следует удалить код формы и на его месте продолжить написание кода обработчика;
5. На следующем шаге следует проверить, введены ли были данные во все поля формы, и если нет – отказать в добавлении статьи в БД, если же все данные в наличии – добавить статью в БД. В начале следует определить, как проводить проверку состояния переменных на заполненность. Это делается через конструкцию `$id!="`, т.е. значение переменной сравнивается с пустотой и если это не так (`!="`) – выполняется действие. В общем виде условие будет выглядеть так:

```
if ($title!=" and $date!=" and $opis!=" and $text!=" and $author!=")  
{  
    Код добавления в БД  
}  
else  
{  
    echo "Статья не добавлена, поскольку не все поля формы были за-  
полнены";  
}
```

6. Код добавления в БД будет выглядеть следующим образом:

```
$pr=mysql_query("INSERT INTO stat(title,date,opis,text,author)
VALUES ('$title', '$date', '$opis', '$text', '$author')");
```

```
if ($pr==1) { echo "Статья добавлена!"; }
```

\$pr – проверка, выполнен ли запрос добавления и если да – вывод сообщения.

Редактирование статьи

Технология редактирования текстов будет выглядеть следующим образом: на первой странице будет выведен список всех статей сайта и по клику на любой из них открывается на этой же странице форма, в полях которой содержатся все записи этой статьи. По щелчку на кнопке в нижней части окна открывается обработчик, который обновляет информацию в БД. В общем виде это будет выглядеть так (Рис. 41):

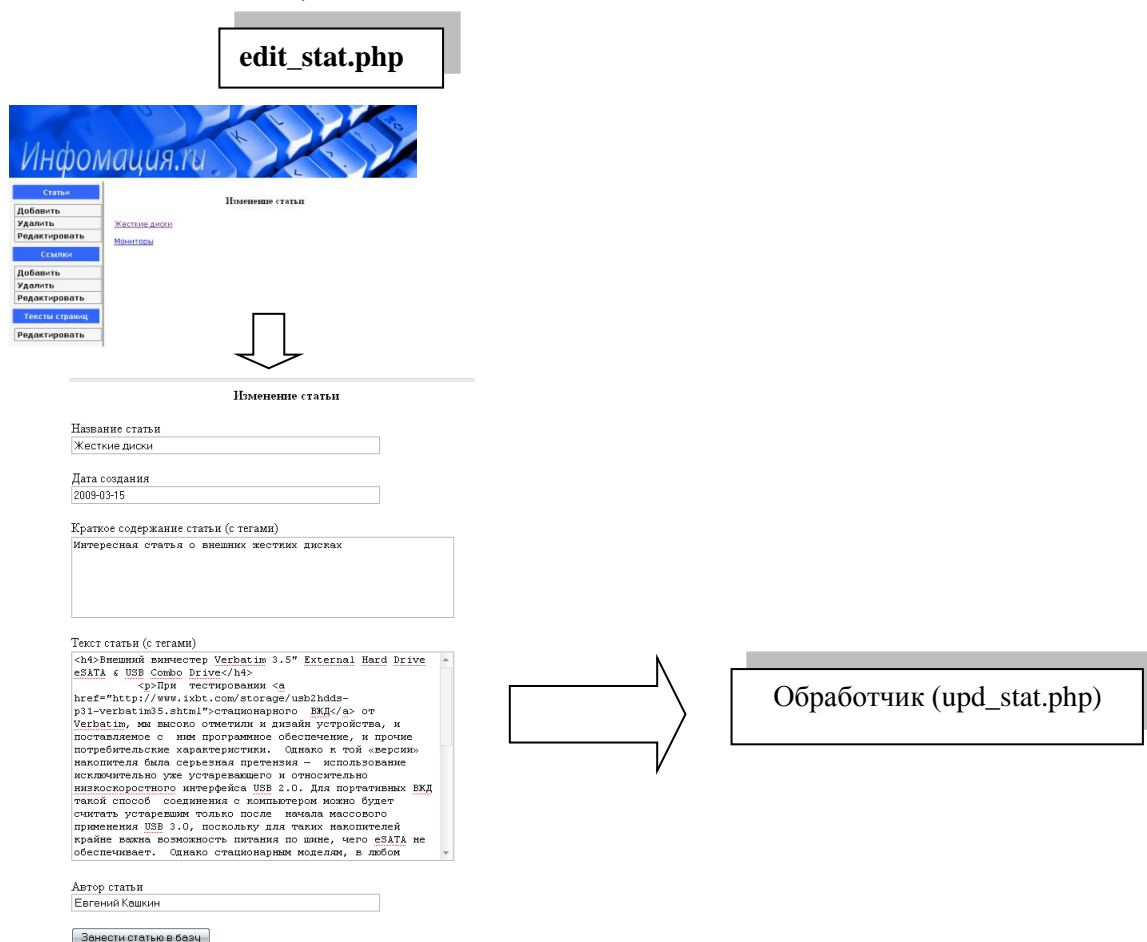


Рисунок 41. Алгоритм редактирования статьи

Страница редактирования

1. Для списка статей и редактирования текстов нам понадобится файл **edit_stat** (который легко создать из файла **new_stat**);
2. В самом верху кода необходимо соединиться с БД (т.к. необходимо будет выбирать оттуда статьи) и создать запрос к БД для выборки всех статей из базы. Нам понадобится заголовок статьи и его **id**, для того чтобы потом по нему выбрать нужную статью из базы:

```
$rez=mysql_query("SELECT title,id FROM stat");
```

3. Теперь временно, чтобы пока не мешалась, следует форму убрать в отдельный файл, а после вернуть назад;
4. После этого на месте формы в цикле производится вывод заголовков статей с ссылкой на этот же файл (**edit_stat.php**) и **id** статьи:

```
<?php
    While ($mas=mysql_fetch_array($rez))
    {
        printf("<p><a
href='edit_stat.php?id=%s'>%s</a></p>", $mas['id'], $mas['title']);
    }
?>
```

5. После того, как названия статей выводятся следует сделать такую модель, при которой пока **\$id** не существует – выводить на странице список статей, а как только появляется (клик на статье) – выводить форму с заполненными полями. Для этого воспользуемся конструкцией:

```
if (!isset($id))
{
    вывод списка статей
}
else
{
    вывод заполненной формы
}
```

6. Само собой разумеется, что для того, чтобы переменная \$id начала существовать в файле – ее надо получить по методу GET в самом начале кода. Но предварительно проверить, а есть ли она(для случая, пока клика по статье еще не было):

```
if (isset($_GET['id'])) { $id=$_GET['id']; }
```

7. Для вывода большого количества HTML кода внутри PHP используется конструкция:

```
print <<<HERE
```

HTML код

```
HERE;
```

Где HERE – маркер, указывающий на начало и конец HTML-блока (может быть любым словом). Важное замечание – после слова HERE и перед началом HTML кода не должно быть не единого пробела!!! иначе работать не будет!

8. Перед кодом формы следует произвести выборку соответствующей записи из БД в массив:

```
$rez=mysql_query("SELECT * FROM stat WHERE id='$id'");  
$mas=mysql_fetch_array($rez);  
print <<<HERE  
код формы  
HERE;
```

9. Для того, чтобы в поля формы ввести данные из БД используйте параметр value (для input type=text), а для textarea – размещайте содержимое соответствующего поля БД между открывающим и закрывающим тегами. **Конструкция, в которой мы выводим форму имеет свою особенность – в ней у индексов ассоциативного массива не ставятся кавычки! А вот сам параметр (\$mas[title]) должен быть в кавычках, иначе будет выводиться лишь первое слово!**

```
value="$mas[title]"
```

10. Помимо этого нам еще понадобится вместе с формой передать значение

id (для того, чтобы обновлять потом не все записи таблицы БД, а лишь ту которая соответствует id). Для этого воспользуемся скрытым полем – полем **input type='hidden'**. Имя поля – **id**, **value='\$mas[id]'**

11.Перепишите обработчик формы на upd_stat.php в action формы;

12.Создайте файл-обработчик upd_stat.php из add_stat.php;

13.В верхней части кода необходимо получить в переменную \$id данные из скрытого поля id:

```
$id=$_REQUEST['id'];
```

14.Далее следует изменить запрос с INSERT на UPDATE в качестве условия указав id:

```
$pr=mysql_query("UPDATE stat SET title='$title',date='$date',opis='$opis',text='$text',author='$author' Where id='$id'");
```

```
if ($pr==1) { echo "Статья обновлена!"; }
```

15.Измените информационные сообщения с «добавлена» на «обновлена» аналогичным образом как в случае «Статья обновлена».

Удаление статьи

В общем виде технология будет выглядеть так (Рис. 41): на странице (**del_stat.php**) вместе с элементами выбора (**input type=radio**) отображаются названия статей. Установив выбор напротив одной из них по клику на кнопке запускается обработчик (**del_obr.php**) и удаляет статью из БД.

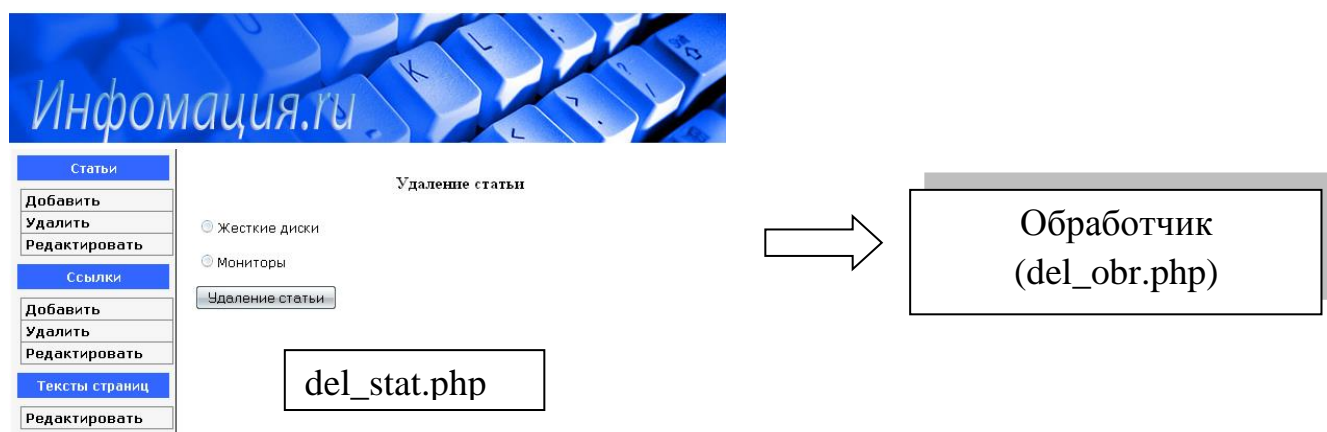
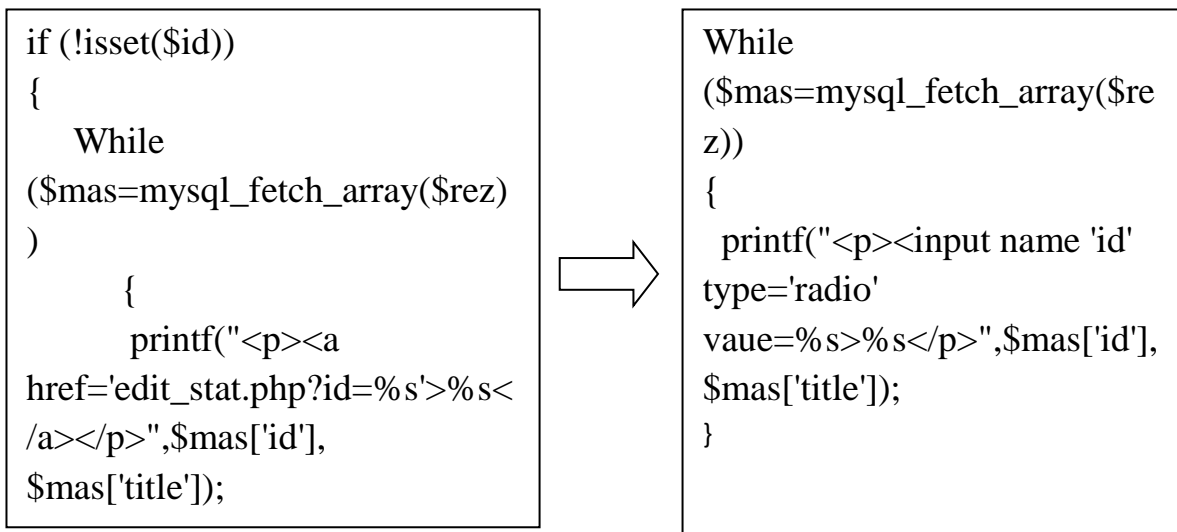


Рисунок 42. Алгоритм удаления статьи

1. Создайте файл **del_stat.php** из edit_stat.php;
2. Удалите **блок else** вместе с формой;
3. Удалите в верхней части **блок проверки** наличия переданного по ссылке id:

```
if (isset($_GET['id'])) {$id=$_GET['id'];}
```

4. В блоке вывода названия статей необходимо вывести в цикле список статей с кнопками. Причем каждая новая кнопка должна иметь собственное возвращаемое значение (value) = текущему id статьи. Для этого замените код вывода:



5. Перед и после блока (в HTML) добавьте тег формы и кнопку submit. В качестве обработчика укажите del_obr.php. А также не забудьте поместить тег кнопки в тег абзаца (для выравнивания ее по тексту);
6. Создайте файл обработчика del_obr.php из файла upd_stat.php;
7. Удалите из него:
 - а. Получение данных из формы (кроме id);
 - б. Проверку наличия данных в полях формы:

```
if ($title!=" and $date!=" and $opis!=" and $text!=" and $author!=")
```

8. Измените запрос к БД на удаление статьи по ее id:

```
<?php
$pr=mysql_query("DELETE FROM stat Where id='$id'");
```

```

        if ($pr==1) { echo "Статья удалена!"; }

        else
        {
            echo "Статья не удалена, поскольку она не выбрана из списка";
        }
    ?>

```

9. Измените информационные сообщения с «обновлена» на «удалена».

Настройка меню

Далее необходимо настроить меню в админке и на всем сайте. Для этого в файле menu.php (в папке blocks для соответствующего раздела сайта – основной части или админки) измените ссылки на те, которые вы установили в своем сайте:

```

<td width="156" valign="top" class="left"><p align="center"
class="title">Навигация</p>

<div id="coolmenu">

<a href="index.php">Главная</a>
<a href="stat.php">Статьи</a>
<a href="link.php">Ссылки</a>
<a href="contacts.php">О нас</a>

</div>

</td>

```

ЛИТЕРАТУРА

1. Р. Никсон. Создаем динамические веб-сайты с помощью PHP, mySQL, JavaScript, CSS и HTML 4-е изд. – Питер, - 2016.
2. Мейер Эрик А. CSS. Карманный справочник. 5-е издание. - Диалектика, - 2019, ISBN 978-5-6041394-1-7.
3. Мэтью Мак-Дональд. HTML5. Недостающее руководство. – БХВ-Петербург, - 2012, ISBN 978-5-9775-0805-6.
4. Дэвид Скляр. Изучаем PHP 7: руководство по созданию интерактивных веб-сайтов. – М.: Диалектика. – 2017. ISBN 978-5-9908462-3-4
5. Бен Фрейн. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. 2-е изд. – СПб.: Питер. – 2017. ISBN 978-5-496-02271-2

СВЕДЕНИЯ ОБ АВТОРАХ

Антонова Ирина Игоревна, к.т.н., доцент, доцент кафедры «Автоматизированные системы управления» Института комплексной безопасности и специального приборостроения Российского технологического университета (МИРЭА).

Кашкин Евгений Владимирович, к.т.н., доцент кафедры «Автоматизированные системы управления» Института комплексной безопасности и специального приборостроения Российского технологического университета (МИРЭА).