ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

ЮТИ ТПУ

Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

РАЗДЕЛ 1. ОРГАНИЗАЦИЯ WEB-САЙТА

ЛЕКЦИЯ 1. ОСНОВЫ РАЗРАБОТКИ WEB-САЙТОВ

ЮТИ ТПУ

Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

Направления развития web-индустрии. Зачем нужен сайт?

Web-сайт - это Ваш электронный офис. Сайты в основном создаются для ведения бизнеса, т.е. получения прибыли.

В настоящее время существует два основных направления использования Интернет в бизнесе: Internet как средство коммуникации, источник справочной информации, средство рекламы и маркетинга для ведения бизнеса (хозяйственной деятельности) вне электронных сетей и Internet как инструмент ведения электронного бизнеса, основанного на принципах сетевой экономики.

Сайты используются для предоставления финансовых услуг (онлайновые платежные системы, обменные пункты и т.п.) и так далее. Кроме того, сайты необходимы при дистанционном обучении, которое является одной из форм получения высшего образования. Таким образом, ведение электронного бизнеса (электронной коммерции) без сайта не представляется возможным.

Что такое сайт?

Web-сайт – это набор Web-страниц связанных между собой гиперссылками. Web-страницы или гипертекстовые документы представляют собой текст, в котором содержатся специальные команды, называемые тегами (tags). Эти теги обеспечивают форматирование элементов страницы и позволяют размещать на ней графические объекты, рисунки, гиперссылки и т.д.

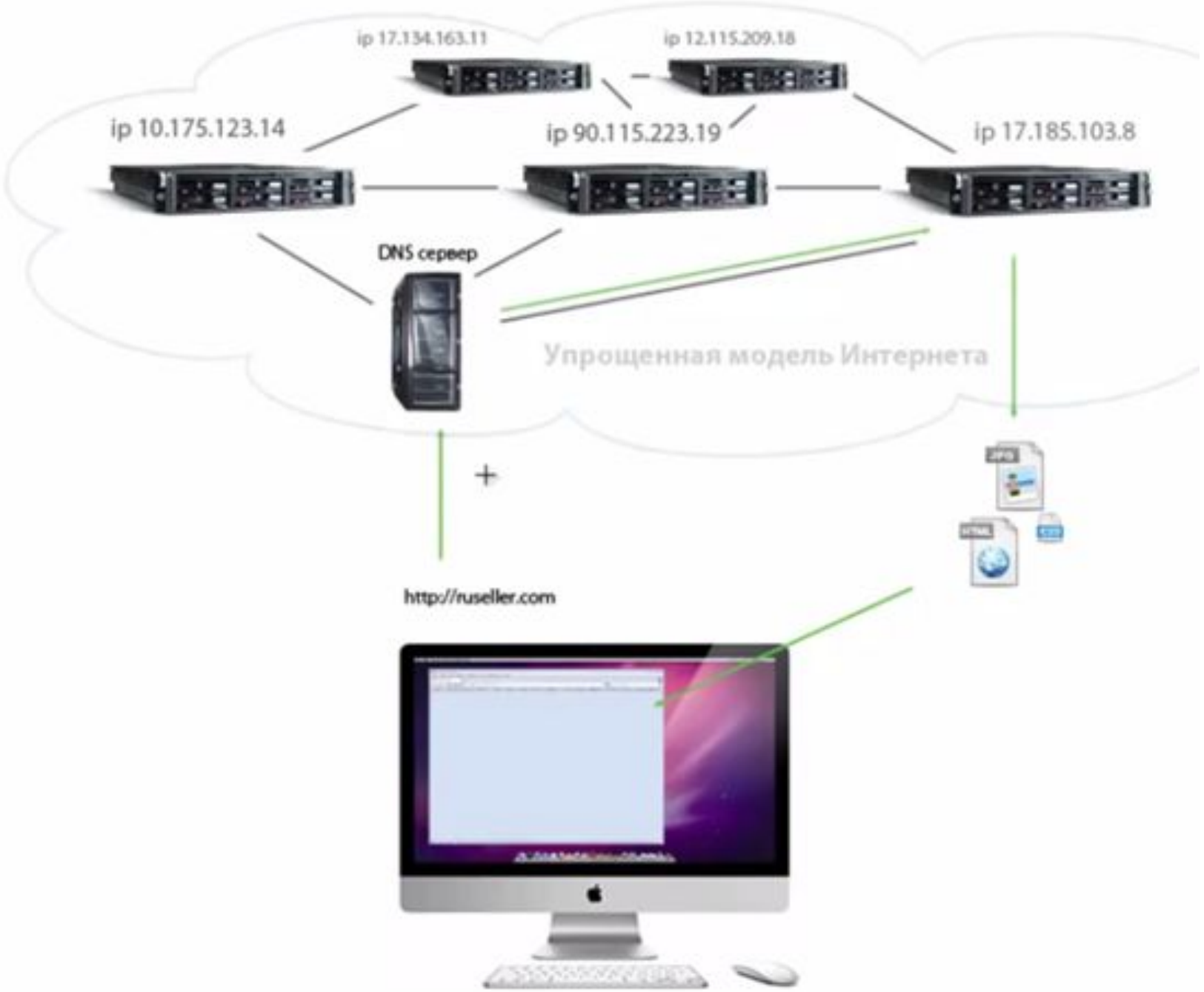
Web-страницы создаются с помощью специального языка HTML. HTML или Hyper Text Markup Language является языком разметки гипертекста, разметка осуществляется с помощью тегов. Сегодня кроме HTML применяются и другие языка разметки: WML, XML.

В настоящее время для создания интерактивных сайтов применяются различные современные технологии: PHP, ASP, Perl, JSP, CSS, базы данных DB2, MsSQL, Oracle, Access и т.д. Современные сайты, как правило, управляемые сайты, т.е. сайты, которые оснащены CMS (Системой Управления Контентом - Content Management Systems).

Как осуществляется передача Web-страниц в сети Интернет?

Возможность работы с Web-страницами обеспечивает один из видов сервиса Internet, который называется World Wide Web или сокращенно WWW. В основу World Wide Web был положен протокол прикладного уровня http, который обеспечивает прием и передачу Web-страниц.

WWW работает по принципу клиент-серверы: серверы Internet, по запросу клиента, который осуществляется с помощью Web-броузера, установленного на компьютере пользователя, направляют ему копии документов. Получив затребованные документы, Web-броузер ПК пользователя, интерпретирует данные и отображает содержание документов на экране.


ip 17.134.163.11
ip 12.115.209.18
ip 10.175.123.14
ip 90.115.223.19
ip 17.185.103.8
DNS сервер
Упрощенная модель Интернета
+
http://ruseller.com

Для создания Веб–сайта компании необходимо:

- определить цель создания сайта;
- разработать ТЗ;
- зарегистрировать домен сайта в определенной зоне (com, ru, ua, net и т.д.);
- разработать сайт;
- разместить сайт на хостинге;
- зарегистрировать в поисковых системах и тематических каталогах;
- выполнить поисковую оптимизацию сайта;
- осуществлять постоянную поддержку сайта.

Определение цели создания сайта

Сайты создаются для различных целей, например: для ведения электронного бизнеса, для поддержки учебного процесса, для предоставления информации, предоставления финансовых услуг и т.д.

Разработка технического задания

В техническом задании учитываются все этапы разработки и сопровождения сайта, цели и назначение сайта, его дизайн, методы навигации, указывается язык разметки страниц и т.д.

Обычно сайт должен включать:

- Информацию о компании, реквизиты: почтовый адрес. Телефон, адрес электронной почты.
- Каталог предлагаемой продукции или услуг.
- Информационный раздел (новости, статьи, аналитические обзоры по тематике предлагаемой продукции или услуг).
- Гостевую книгу.
- Способы оплаты.
- Счетчики числа посетителей (счетчики рейтингов).

Регистрация домена

Регистрация домена осуществляется в выбранной пользователем зоне ua, ru, com, net, info и так далее. В зависимости от назначения сайта выбирается его зона регистрации. Для регистрации сайта желательно выбрать домен второго уровня или третьего уровня, например www.lessons-tva.info.

Домен второго уровня регистрируется у регистратора – организации занимающейся администрированием доменных имен, например Регистрация доменов. Домен третьего уровня приобретается, как правило, вместе с хостингом у хостинговой компании. Имя сайта выбирают исходя из вида деятельности, названия компании или фамилии владельца сайта.

maps.google.com

Разработка сайта - важнейший этап создания сайта

При разработке сайта необходимо уделять большое внимание содержимому, структуре и дизайну (графическому оформлению) Web-страниц, а также структуре Web-сайта и методам навигации по Web-узлу.

Главное на сайте – это его содержание или контент, структурированность информации, навигация, а затем графическое оформление или дизайн сайта.

Для разработки сайта используются различные средства: конструкторы сайтов (дизайнеры), WebCoder 1.6.0.0, профессиональные приложения: Macromedia HomeSite Plus, Macromedia Dreamweaver, Microsoft FrontPage и т.д. Для создания сайтов целесообразно использовать редакторы на русском языке.

При создании сайта необходимо оптимизировать его для поисковых систем, так как целевой посетитель приходит на сайты в основном с поисковых систем, поэтому необходимо стремиться к высокому рейтингу в поисковых системах.

Особое внимание необходимо уделять таким мета - тегам как Title (заголовок), Keywords (ключевые слова) и Description (описание), а также расположению ключевых слов в тексте Web-страниц.

Размещение сайта на хостинге

Веб-хостинг - это место для размещения сайта на сервере в сети Internet, который предоставляет доступ к Web-страницам посетителям сайта. Серверы предлагают как платные, так и бесплатные хостинги. Отличие этих хостингов состоит в качестве предоставляемых услуг.

Для размещения сайта на хостинге необходимо зарегистрироваться на одном из серверов, который предоставляет услуги по размещению. Интернет-адрес или доменный адрес сайта зависит от того, какой Вы уровень домена приобрели. При работе в Internet используются не доменные имена, а универсальные указатели ресурсов, называемые URL (Universal Resource Locator).

URL - это адрес любого ресурса (документа, файла) в Internet, он указывает, с помощью какого протокола следует к нему обращаться, какую программу следует запустить на сервере и к какому конкретному файлу следует обратиться на сервере. Общий вид URL: протокол://хост-компьютер/имя файла (например, http://www.lessons-tva.info/book.html).

Для загрузки файлов сайта на сервер можно использовать файловый менеджер; с помощью браузера; WC или Total Commander.

Регистрация сайта в поисковых системах и тематических каталогах

После размещения сайта на хостинге необходимо зарегистрироваться в поисковых системах и тематических каталогах Yahoo, Rambler, Апорт и осуществить раскрутку сайта. Для раскрутки применяются различные средства.

Контроль посещаемости сайта осуществляется по счетчикам. Поисковые машины, как правило, имеют рейтинговые системы, которые ранжируют ресурсы по их посещаемости. Для участия в рейтинге установите на главной странице своего сайта счетчики рейтингов.

Поддержка и регулярные обновления (развитие) сайта

Далее Вы должны осуществлять поддержку и регулярные обновления сайта. Причем чем чаще Вы будете обновлять информацию на сайте, тем больше будет целевых посетителей сайта и естественно потребителей Вашей продукции или услуг.

Разработка концепции сайта. Этапы работы.

В голове у разработчика уже должен находиться определенный план действий, он должен четко представлять направленность и тематику сайта, основные ключевые моменты, характер целевой аудитории, цели и задачи, поставленные перед сайтом. Именно для этого и создается концепция, которая является ключевым моментом при разработке сайтов.

Гораздо проще создать концепцию, разбив процесс на несколько этапов. И первым из них будет идейная направленность сайта.

После этого вы сможете выбрать для сайта название. Оно будет одновременно и названием интернет-ресурса, и слоганом компании, для которой создается сайт. Название должно быть не длинным, легко читаемым и запоминающимся, а также должно отображать концепцию компании или направленность ее деятельности. Например: «Наша работа – ваш успех».

ЛЕКЦИЯ 2. ТЕХНОЛОГИИ СОЗДАНИЯ САЙТОВ

ЮТИ ТПУ

Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

На данный момент сайты есть уже практически у всех достаточно крупных компаний. А те, у кого сайта нет, мечтают его создать. И, в последнее время, большинство пользователей начали понимать, что *создание сайта* – не такое уж легкое дело. Существуют определенные **технологии создания сайтов**, которыми необходимо отлично владеть, чтобы создать хороший, работающий сайт.

PHP-скрипт

Это скриптовый язык программирования, созданный для генерации HTML-страниц на веб-сервере и работы с базами данных. На данный момент он поддерживается практически всеми представителями хостинга, входит в «стандартный» набор для создания сайтов (LAMP – Linux, Apache, MySQL, PHP).

Благодаря своей простоте, скорости выполнения, богатой функциональности, распространению исходных кодов на основе лицензии PHP, этот язык является чуть ли не самым популярным в области **технологий создания сайтов**. Отличается наличием ядра и подключаемых модулей, «расширений»: для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF и т.п. Есть возможность разработать, а также подключить дополнительное расширение.

Возможности PHP очень обширны. Главным образом, PHP применяется при написании скриптов, работающих на стороне сервера; таким образом, PHP способен выполнять всё то, что выполняет любая другая программа CGI (например, обрабатывать данных форм, генерировать динамические страницы, отсылать и принимать cookies). Но PHP дает возможность выполнять также множество других задач. Существуют три основных области, где используется PHP:

Создание скриптов для выполнения на стороне сервера.

Создание скриптов для выполнения в командной строке.

Создание приложений GUI, выполняющихся на стороне клиента.

Помимо этого PHP: - доступен для большинства операционных систем, включая Linux

Java Script

Это пока еще относительно молодой язык программирования, но уже очень популярный в области **технологий создания сайтов**. На данный момент, работа над ним еще не закончена. Он постоянно дорабатывается и совершенствуется. Технический комитет работает над существенными расширениями, включая механизмы для сценариев, которые будут созданы для применения в Internet, а также более жесткой координацией с другими основными стандартами групп World Wide Web Консорциум и Wireless Application Protocol Форум.

HTML

Этот язык является базовым в области *технологий создания сайтов*, так как относительно легок в освоении. Но чрезмерная простота является и его недостатком. HTML (от английского Hyper Text Markup Language – язык разметки гипертекста) прекрасно отвечал требованиям раннего периода развития технологий создания сайтов, но с дальнейшим его развитием возникли существенные проблемы. HTML предоставляет следующие возможности:

Издавать сетевые документы с заголовками, текстом, таблицами, списками, фотографиями и т.п.

Получать информацию из Сети через ссылки гипертекста при нажатии кнопки.

Создавать формы для посылки запросов на удаленные компьютеры, чтобы производить поиск информации, осуществлять бронирование, заказывать товары и т.п.

Включать электронные таблицы, видео клипы, аудио клипы, и другие программные приложения непосредственно в их документы.

История разработки HTML довольно длительна. В каждой его версии разработчики пытались добиться того, чтобы HTML-страницы читались всеми браузерами, на всех компьютерных платформах.

СУБД и MySQL

SQL (от Structured Query Language – структурированный язык запросов) – создан для работы с реляционными базами данных. Он позволяет пользователям взаимодействовать с базами данных (просматривать, искать, добавлять, управлять данными). MySQL – многопользовательский, многопоточный сервер базы данных SQL. Имеет хорошую скорость и гибкость, если использовать его для хранения изображений и файлов.

Его преимущества:

- Поддержка нескольких одновременных запросов (многопоточность).
- Возможность записи фиксированной, а также переменной длины.
- Оптимизация связей с присоединением многих данных за один проход.
- Гибкая система паролей и доступов.
- ODBC драйвер в комплекте с исходником.

ASP.NET

Одним из самых современных подходов в области создания сайтов является asp.net технология. Стоит особо отметить, что в asp.net имеется предустановленная мощная защита от хакеров, что в связи с участившимися случаями взлома имеет огромное значение в защите данных. Подобная защита просто неоценима особенно для крупных компаний, которые дорожат своей репутацией. Платформа asp.net работая совместно с сервером MS SQL - базы данных позволяет создать самые разнообразные сайты, начиная с простых одностраничников и заканчивая крупнейшими порталами.

При создании сайтов с использованием данной платформы используются разнообразные технологии, такие как графическое наполнение, верстка HTML, программирование серверной составляющей и клиентской части ресурса.

Стоит особо отметить, что разработка сайтов cms с использованием технологии ASP.NET в значительной степени облегчает работу программиста, поскольку в ней содержится полнейший комплекс технологий. К сожалению, споры о том, насколько быстродейственны сайты на разных технологических платформах не умолкают, и по сей день. Большая часть специалистов утверждает, что сайты на PHP намного быстрее, чем на ASP.NET. Однако по факту быстродействие технологии ASP.NET, которая совместно использует MS SQL, практически ни чем не уступает связке PHP и MySQL.

В любом случае выбор технологии при помощи, которой будет создаваться сайт, остается за заказчиком, а если знаний не достаточно, то сотрудники компании по созданию сайта подскажут наиболее оптимальный вариант.

ЛЕКЦИЯ 3. СТРУКТУРА WEB-САЙТА

ЮТИ ТПУ

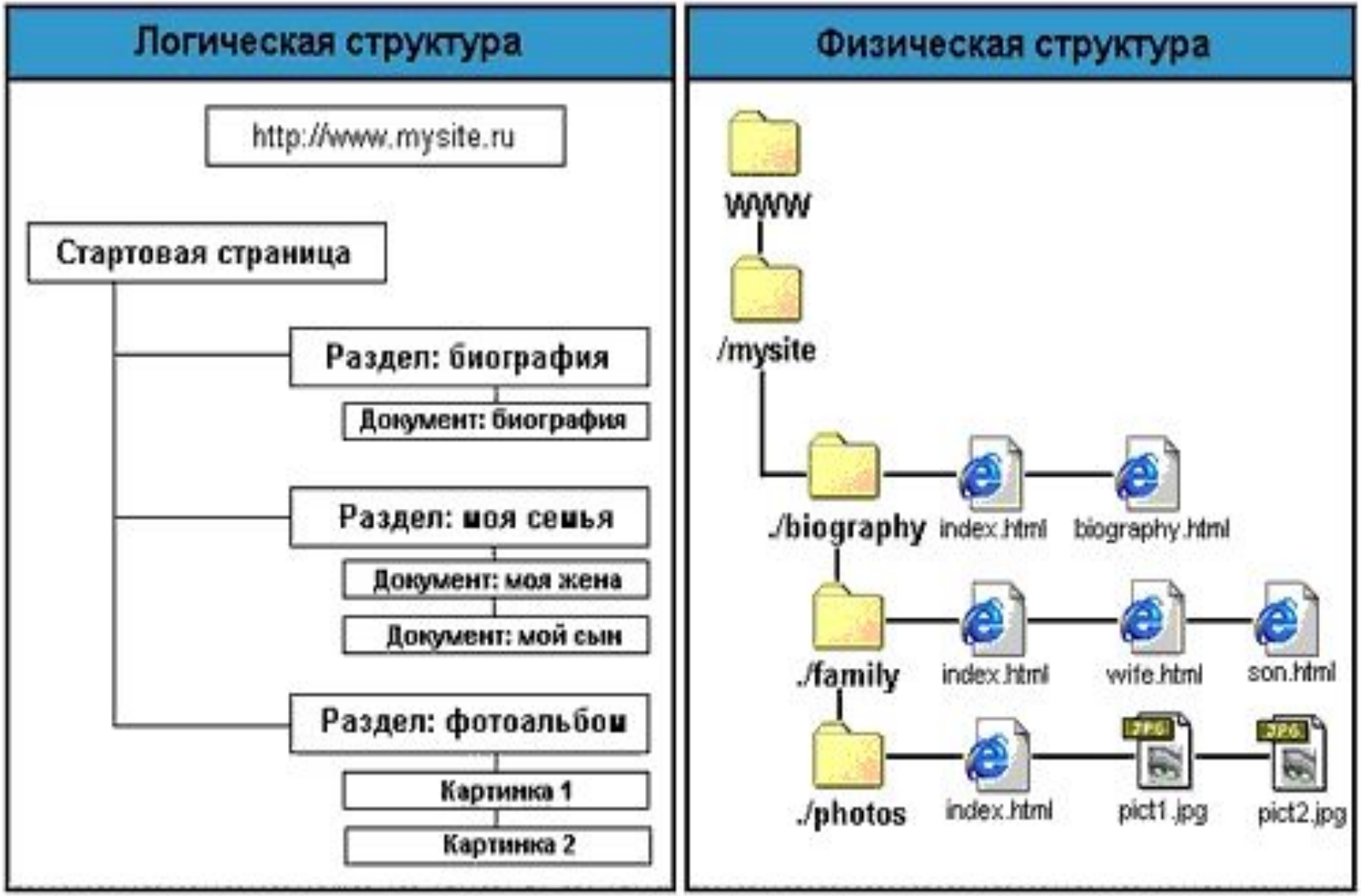
Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

Логическая и физическая структура сайта

Каждый ресурс Интернета, от любительской домашней странички до большого информационного портала, содержит несколько тематических рубрик, соединенных между собой гиперсвязями. Как правило, ссылки на все разделы сайта с краткими анонсами их содержимого приводится на первой, так называемой стартовой странице, которой присваивается имя index.htm (.html). Если тематические рубрики содержат собственные подразделы, каждая из них также имеет свою стартовую страницу, называющуюся index.html.

Подобный набор тематических рубрик с распределенными по соответствующим разделам документами и заранее спроектированными гиперсвязями между всеми страницами ресурса и называется логической структурой сайта. Физическая структура, напротив, подразумевает алгоритм размещения физических файлов по поддиректориям папки, в которой опубликован сайт.


Логическая структура
http://www.mysite.ru
Стартовая страница
Раздел: биография
Документ: биография
Раздел: моя семья
Документ: моя жена
Документ: мой сын
Раздел: фотоальбом
Картинка 1
Картинка 2
Физическая структура
www
/mysite
./biography
index.html
biography.html
./family
index.html
wife.html
son.html
./photos
index.html
pict1.jpg
pict2.jpg

Физическая структура сайта скрыта от посетителей вашего ресурса: они могут наблюдать только логическую структуру, причем именно так, как она представлена при помощи элементов навигации. Отсюда следует вполне логический вывод: строение системы навигации должно если не полностью повторять, то хотя бы максимально соответствовать разработанной вами логической структуре сайта.

Заглавная страница представляет собой html-документ, который не включает в себя какую-либо содержательную информацию и элементы навигации.

Файлу заглавной страницы присваивается имя index.html, при этом стартовая страница называется иначе и вызывается посредством организации гиперссылки с заглавной страницы, загружающейся при обращении к сайту первой. Заглавная страница содержит, как правило, логотип компании - владельца данного ресурса, счетчик посещений и предложение выбора кодировки кириллицы, либо выбора между английской и русской версиями сайта.

Динамическая или статическая компоновка сайта

Для того чтобы избежать «съезжания» элементов html-документа друг относительно друга и, как следствие, деформации web-страницы в целом при изменении параметров экрана, применяется достаточно простой и действенный прием: все компоненты web-страницы заключаются в соответствующие ячейки невидимой таблицы, при этом каждому объекту назначается одно, строго определенное положение. Таким образом, появляется второй критерий, по которому можно разделить все существующие web-сайты на две условные категории. Данной таблице можно назначить строго определенную ширину в пикселах, например, 640 точек, после чего жестко позиционировать ее по центру экрана или «прижать» к левому его краю. Такой вариант компоновки сайта можно назвать статическим, поскольку ширина таблицы не меняется в зависимости от экранного разрешения. Разумеется, при изменении параметров экрана не происходит ни малейшего смещения элементов дизайна страницы.

Иной подход — когда ширину невидимой таблицы, содержащей фрагменты web-страницы, задают в процентах от текущей ширины экрана. При увеличении экранного разрешения таблица «растягивается» по горизонтали, и все размещенные в ее ячейках элементы, позиционированные либо по центру, либо по краям столбцов, смещаются согласно установленному алгоритму. В силу того, что параметры таблицы изменяются в зависимости от настроек экрана, такой принцип компоновки html-документа можно назвать динамическим. И тот и другой подход обладает как достоинствами, так и недостатками, которые перечислены ниже.

Элементы web-страницы

Любая web-страница содержит определенный набор стандартных элементов, являющихся обязательными компонентами каждого ресурса Интернета. Безусловно, ассортимент и количество подобных объектов могут варьироваться в зависимости от тематической направленности сайта, объема опубликованных на нем материалов, а также от целей и задач, которые ставит перед собой создатель данного ресурса. Компоновка таких элементов, проектирование их взаимного расположения и составляет одну из главных задач web-мастера.

Элементы web-страницы: заголовок, рекламный баннер, логотип, текстовое поле, элементы навигации, информация о разработчиках сайта и адрес электронной почты, счетчик посещений.

Пример компоновки сайта

Пример компоновки сайта

Пример компоновки сайта

Пример компоновки сайта

ЛЕКЦИЯ 4. ОСНОВЫ ЯЗЫКА HTML

ЮТИ ТПУ

Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

Язык HTML (от англ. Hyper Text Markup Language – «язык разметки гипертекста») служит для создания веб-страниц. Большинство сайтов созданы именно с помощью HTML.

Синтаксис *HTML*

HTML-документы представляют собой файлы с текстом и дополнительными инструкциями языка HTML, называемыми тегами. Теги позволяют задавать форматирование текста, а также размещать в документе мультимедийные файлы (изображения, звук, Flash-анимацию), гипертекстовые ссылки на другие документы, табличные данные, формы ввода данных. HTML-документы имеют расширение имени файла htm или html. Редактирование HTML кода производят в текстовом редакторе (например, в обычном блокноте), а про-смотр – в браузере.

Структура тега:

<имя тега атрибут1 атрибут2="значение2" ...>

Тег состоит из имени тега, за которым может следовать список атрибутов, помещаемых между открывающей и закрывающей угловыми скобками (< и >). Атрибуты позволяют управлять поведением тега. Они могут иметь конкретные значения, задаваемые после знака равенства. Значения атрибутов заключаются в одиночные или двойные кавычки ("). Атрибуты отделяются друг от друга пробелом, порядок следования атрибутов значения не имеет. Имена тэгов и атрибутов нечувствительны к регистру.

Пример: <FONT color="red" face="Arial">

Теги подразделяются на парные и непарные. Парные теги имеют закрывающий тег, непарные – не имеют. Закрывающий тег содержит косую черту перед именем и не имеет атрибутов. Между открывающим и закрывающим тегами помещается текст и другие теги. Атрибуты указываются только в открывающем теге.

Для выделения текста жирным используется тег . Пример: HTML-код: текст жирный текст текст

Примером непарного тега является тег
 – перевод строки. Обычный перевод строки клавишей {Enter} браузер игнорирует (как и несколько постав-ленных подряд пробелов или знаков табуляции).

Тег используется для выделения текста курсивом.

Неправильно:	HTML-код:	жирный курсив
Правильно:	HTML-код:	жирный курсив
В браузере:	**жирный курсив**	

Неправильно: HTML-код:

первая строка
вторая строка

В браузере:
первая строкавторая строка

Правильно:

HTML-код:

первая строка
вторая строка

В браузере:

первая строка
вторая строка

Структура документа *HTML*

```
<html>
<head>
…  заголовок документа
</head>
<body>
…  тело документа
</body>
</html>
```

HTML-документ заключен в тег <html> и состоит из заголовка и тела. Заголовок документа лежит внутри тега <head> и содержит название документа и некоторые другие параметры. Тело документа заключено в тег <body> и содержит текст и теги, которые должен обработать и вывести браузер. Текст из тега <title> обычно отображается в заголовке окна браузера, а также в результатах поиска поисковых систем.

Пример: простейший HTML-документ

```
<html>
<head>
<title>Заголовок</title>
</head>
<body>
Мой первый <i>HTML-документ!</i><br>
(это пример)
</body>
</html>
```

Представление цвета в *HTML*

Цвет в HTML может быть задан ключевыми словами – названиями цветов на английском языке.

Название в HTML	Название на русском	Код в RGB
aqua	морская волна	#00ffff
black	черный	#000000
blue	синий	#0000ff
fuchsia	фуксия	#ff00ff
grey	серый	#808080
green	зеленый	#008000
lime	ярко-зеленый	#00ff00
maroon	темно-бордовый	#800000
navy	темно-синий	#000080
olive	оливковый	#808000
purple	пурпурный	#800080
red	красный	#ff0000
silver	серебряный	#c0c0c0
teal	бирюзовый	#00808
white	белый	#ffffff
yellow	желтый	#ffff00

Но компьютер может отобразить гораздо больше – около 16 миллионов – цветов. Альтернативным способом задания цвета является указание кода цвета в системе RGB (от англ. *red, green, blue* – красный, зеленый, синий). Суть системы заключается в том, что любой цвет может быть представлен как смешение основных цветов – красного, зеленого и синего. Цвет записывается в виде 6-символьного кода.

Код представляет собой шестнадцатеричное число от 000000 до FFFFFF. Первые две цифры соответствуют красной компоненте, следующие две – зеленой, последние две – синей. Значение 00 означает полное отсутствие составляющей, значение FF (255) – максимум составляющей. В качестве шестнадцатеричных цифр используются десятичные цифры от 0 до 9 и латинские буквы от A до F для обозначения цифр от 10 до 15. Таким обра-зом, получается $256^3 \approx 16.7$ млн. цветов – этого достаточно, чтобы вос-произвести любой цвет, который различает человеческий глаз.

Например:
FF0000 – ярко-красный (red)
00FF00 – ярко-зеленый (green)
0000FF – ярко-синий (blue)
FFFF00 – желтый (yellow) – смесь красного и зеленого
000000 – черный (black)
FFFFFF – белый (white)

Значение цвета указывается в теге после символа решетки (#).

Например для текста:

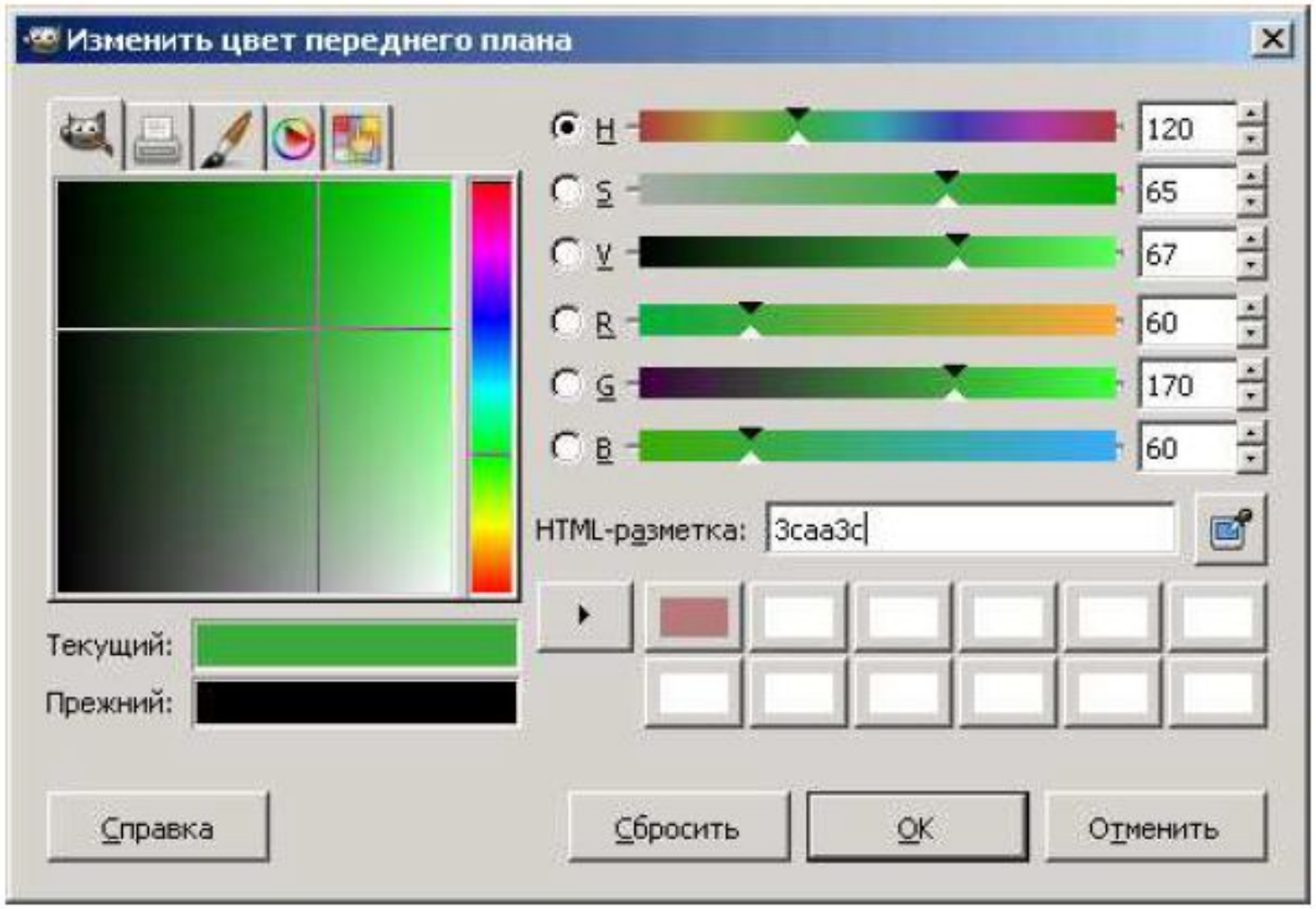
```
<font color="#808080">серый текст</font>
```

Для фона всей страницы в теге body артибут bgcolor:

```
<body bgcolor="# FFFF00">желтый фон</body>
```

Значение кода сложно подобрать самому, поэтому используются специальные инструменты.

Например, можно набрать в Яндексе запрос «подбор цвета»:


Изменить цвет переднего плана
H
S
V
R
G
B
120
65
67
60
170
60
HTML-разметка:
3caa3c
Текущий:
Прежний:
Справка
Сбросить
OK
Отменить

Основные теги, работа с текстом, списки

Теги структуры документа

Эти теги предназначены для определения структуры HTML документа и не влияют на его отображение в браузере. Тем не менее, правильно сформиро-ванный документ обязательно должен их содержать.

<html>…</html> – включает в себя все содержимое веб-страницы, в том числе теги <head> и <body>

<head>…</head> – содержит теги со служебной информацией о странице, например название в теге <title>.

<title>…</title> – задает название документа. Это название обычно ото-бражается в заголовке окна браузера.

<body>…</body> – хранит содержимое документа.

Атрибуты:

`bgcolor="цвет"` – назначает цвет фона документа

`text="цвет"` – указывает цвет обычного текста в документе

! Теги <html>, <head>, <title> и <body> задаются в документе только 1 раз!

Теги для работы с текстом

HTML позволяет управлять отображением текста на странице.

… – выделение текста жирным

… – выделение текста курсивом

<u>…</u> – подчеркивание текста

_… – форматировать текст как подстрочный индекс

Пример:

HTML-код: `101<sub>2</sub> = 5`

В браузере: $101_2 = 5$

[…] – форматировать текст как надстрочный индекс

Пример:

HTML-код: 2⁸ = 256

В браузере: $2^8 = 256$

** … ** – выравнивание текста по центру

… – устанавливает размер, цвет и гарнитуру текста

Атрибуты:

`color="цвет"` – задает цвет текста

`face="шрифт"` – определяет гарнитуру текста; значением атрибута может быть список шрифтов, перечисленных через запятую – в этом случае выбирается первый доступный шрифт

`size="1-7"` – устанавливает размер шрифта (от 1 до 7)

Пример:

HTML-код:

```
<FONT face="Tahoma" size="2" color="gray">текст</FONT>
```

В браузере: текст

** … ** – задает начало и конец параграфа

Атрибут:

`align="…"` – определяет режим выравнивания текста

`left` – по левому краю (по умолчанию)

`center` – по центру

`right` – по правому краю

`justify` – по ширине

<hN>…</hN> – вложенный текст, является заголовком документа уровня N, N принимает значения от 1 до 6. Наибольшим заголовком является `<h1>`, наименьшим `<h6>`.

Тег *HR*

<hr> – выводит горизонтальную разделительную линию
Атрибуты:

align="…" – определяет режим выравнивания линии left – по левому краю

center – по центру (по умолчанию) right – по правому краю

noshade – использовать сплошную линию вместо объемной size="N" – толщина линии в пикселах

width="N" – ширина линии в пикселах или процентах по отношению к ширине экрана.

Размеры объектов в HTML часто указываются в пикселях. Пиксель – наименьший элемент изображения на экране (точка). Количество пикселей на экране по горизонтали и вертикали называют разрешением (например, 1024 по горизонтали на 768 по вертикали).

Работа со списками

В HTML есть возможность создавать нумерованные и маркированные списки.

**
…
** – создает нумерованный список элементов

Атрибуты:

start="N" – начать нумерацию с числа N type="…" - определяет формат нумерации

1 – арабские цифры (по умолчанию)

A – прописные буквы (A, B, C)

a – строчные буквы (a, b, c)

I – прописные римские цифры (I, II, III) i – строчные римские цифры (i, ii, iii)

**
…
** – создает маркированный список элементов

Атрибут:

type="…" – определяет формат маркера

disk – диск (по умолчанию)

circle – окружность

square – квадрат

**
…
** – задает элемент списка в нумерованном или маркированном списке Атрибуты:

type="…" – формат номера или маркера

value="N" – задает номер элемента списка

```
<ol>
<li>арабские цифры (по умолчанию)</li>
<li type="A">прописные буквы</li>
<li type="a">строчные буквы</li>
<li type="I">прописные римские цифры</li>
<li type="i">строчные римские цифры</li>
</ol>
<ul>
<li>диск (по умолчанию)</li>
<li type="circle">окружность</li>
<li type="square">квадрат</li>
</ul>
```

В браузере:

1. арабские цифры (по умолчанию)

B. прописные буквы

c. строчные буквы

IV. прописные римские цифры

v. строчные римские цифры

- диск (по умолчанию)

○ окружность

▪ квадрат

Создание ссылок

Для создания ссылок используется тег **<a>…**.

Обязательный атрибут `href` указывает абсолютный или относительный адрес, на который ведет ссылка. Ссылка может указывать на HTML-документ, изображение, файл для сохранения на диск и пр. Текст ссылки записывается между открывающим и закрывающим тегом.

Абсолютный адрес содержит в себе имя хоста и полный путь к ресурсу, например: http://www.example.com/docs/about.html. С помощью абсолютного адреса можно ссылаться на любой открытый ресурс в Интернете. Если нужно поставить ссылку на главную страницу сайта, указывают его адрес и слеш.

Пример для абсолютного адреса:
HTML-код: <A href="http://www.yandex.ru">Яндекс</A>
В браузере: Яндекс

В браузере ссылка обычно представляется как подчеркнутый текст. При клике по ссылке браузер загружает страницу, указанную в атрибуте href.

Также для документов, расположенных на том же сайте, можно использовать относительный адрес.

Создание ссылок

Например, чтобы поставить ссылку из файла file1.html на файл file2.html необходим следующий HTML-код:

```
<A href="folder1/file2.html">файл file2.html</A>
```

А чтобы ссылка в файле file2.html указывала на file1.html:

```
<A href="../file1.html">файл file1.html</A>
```

Две точки (..) означают переход к родительскому каталогу.

Замечание: для файлов в пределах одного сайта рекомендуется использовать только относительные пути. В этом случае ссылки сохранят работоспособность при изменении адреса сайта, переносе в другую папку и т.п.

Для открытия ссылки в новом окне используется атрибут `target` со значением `_blank`.

Пример: `<a href="http://ya.ru/" target="_blank">Яндекс</a>`

Цвет ссылок в документе можно указать атрибутами тега `<body>`:

`alink="цвет"` – устанавливает цвет активных ссылок

`link="цвет"` – задает цвет непосещенных ссылок

`vlink="цвет"` – определяет цвет посещенных ссылок

Вставка изображений на странице

Осуществляется непарным тегом **<img>**. Обязательный атрибут `src` указывает абсолютный или относительный URL изображения . Стандартными форматами изображений являются GIF, PNG и JPEG.

GIF – формат, реализующий сжатие без потери качества с ограниченной цветностью (от 2 до 256 цветов) и поддержкой анимации – используется для хранения графики, когда достаточно 256 (и меньше) цветов. Обычно это небольшие изображения. Также GIF поддерживает прозрачность.

JPEG реализует сжатие изображений с потерями качества, при этом ограничения на цвет отсутствуют (поддерживается 16 миллионов цветов). Размер JPEG-файла зависит от параметра «качество», который указывается при его сохранении: от 0 до 100. Чем выше качество, тем больше размер файла. Оптимальная степень качества зависит от изображения, в большинстве случаев она равна 70-80. Не стоит выставлять этот параметр меньше 50 – на изображении появятся заметные дефекты или больше 95 – размер файла сильно возрастет без видимого улучшения качества.

Формат **PNG** существует в двух вариантах: PNG-8 и PNG-24. PNG-8, как и GIF, поддерживает 256 цветов, обеспечивает по сравнению с ним лучшее сжатие, но не поддерживает анимацию. Формат PNG-24, как и JPEG, не имеет ограничений на количество цветов, но проигрывает ему в размере файла. Осуществляет сжатие изображений без потери качества, поэтому его стоит применять для изображений, содержащих мелкие детали.

Особенности изображения	Предпочтительный формат
анимированное изображение	только GIF
маленькое изображение с небольшим количеством цветов	GIF или PNG-8
изображение с полупрозрачностью	только PNG
изображение с большим количеством цветов, например фотография	JPEG
изображение с большим количеством цветов с мелкими деталями, например скриншот (снимок экрана)	PNG-24

`align="…"` – определяет режим выравнивания изображения относительно текста в строке:

`top` – по верхнему краю

`middle` – по центру строки

`bottom` – по нижнему краю (по умолчанию)

`left` – по левому краю окна

`right` – по правому краю окна

`alt="…"` – определяет текст, описывающий изображение для браузеров без поддержки графики (или с отключенной графикой), поисковых машин и т.п.

`border="N"` – устанавливает толщину рамки вокруг изображений, равной N пикселей, 0 – для отключения рамки

`height="N"` – высота изображения в пикселях или процентах

`width="N"` – ширина изображения в пикселях или процентах

Изображение может быть сделано ссылкой, путем помещения внутрь тега <a>. В этом случае вокруг изображения автоматически появляется рамка. Толщина рамки задается атрибутом border. Обычно рамку убирают, указывая border="0" в теге <img>.

Примеры:

1. HTML-страница находится в папке site, а изображение picture.jpg находится в папке site/images/.

```
<img src="images/picture.jpg" alt="фотография">
```

2. Изображение находится на другом сайте в Интернет

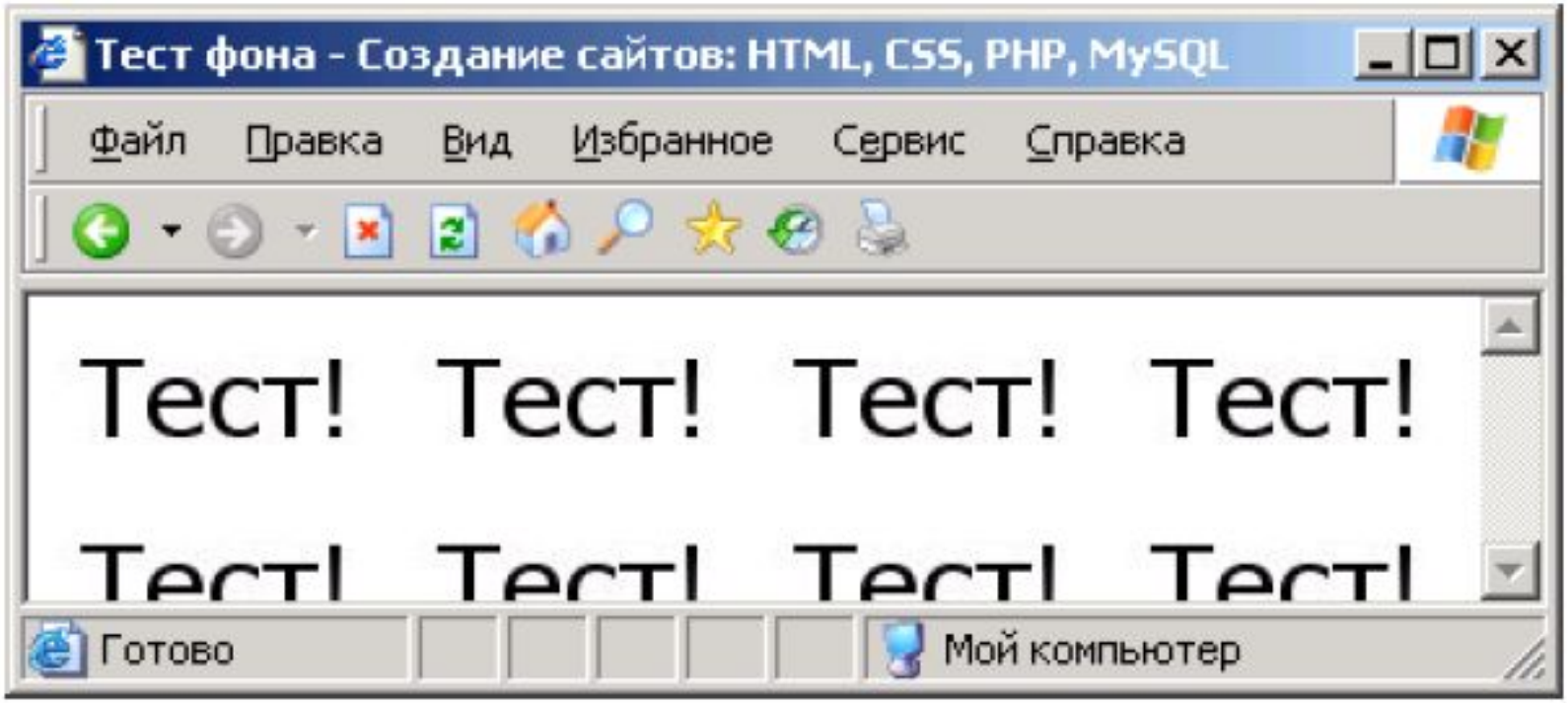
```
<img src="http://example.com/pics/tree.gif" alt="дерево">
```

Фоновое изображение страницы

Можно задавать адрес фонового изображения для страницы в атрибуте `background` тега `<body>`. Фоновое изображение отображается в натуральную величину. Если размер изображения меньше размера окна браузера, то рисунок повторяется по горизонтали вправо и по вертикали вниз. Например, зададим фоновым изображением страницы рисунок bg1.jpg.

Рисунок. 2.4. Файл ***bg1.jpg***

```
<html>
<head>
<title>Тест фона</title>
</head>
<body background="bg1.jpg">
</body>
</html>
```

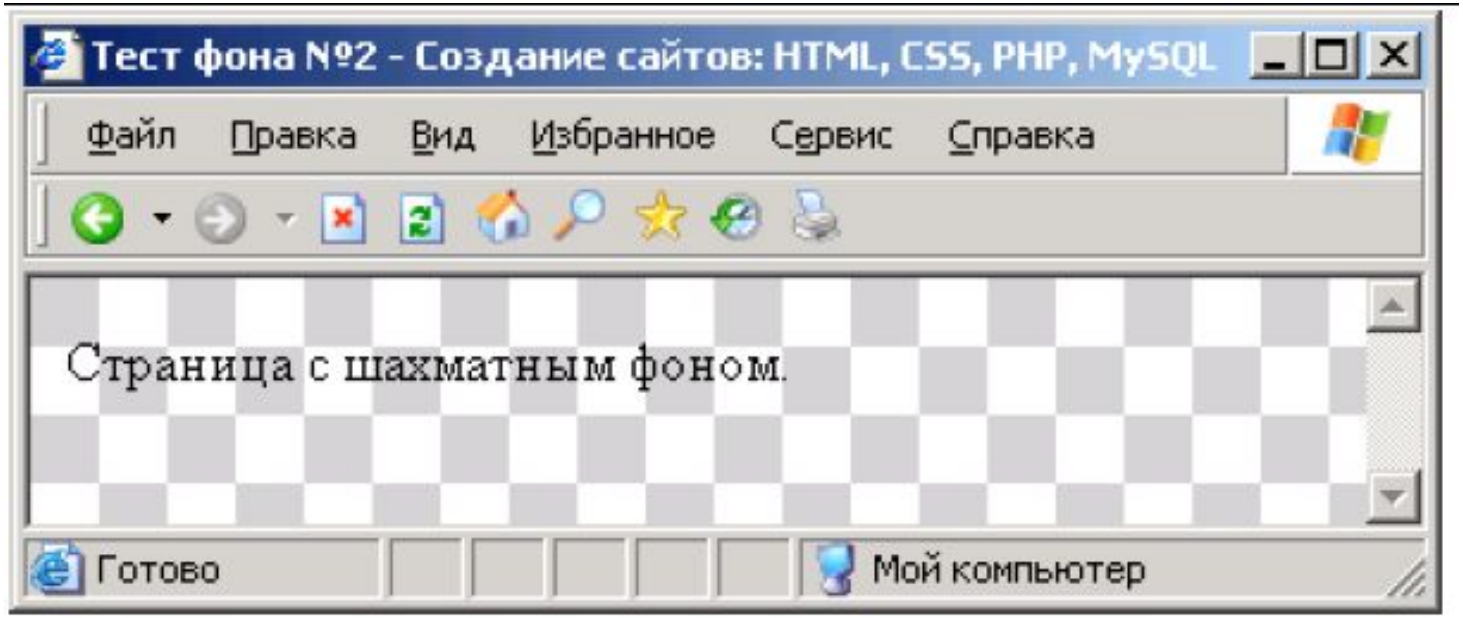

Тест фона - Создание сайтов: HTML, CSS, PHP, MySQL
Файл Правка Вид Избранное Сервис Справка
Тест! Тест! Тест! Тест!
Готово
Мой компьютер

Чтобы сделать неповторяющийся фон, необходимо выбрать картинку заведомо большую, чем размер страницы по ширине и высоте.

Можно использовать повторяющийся фон:

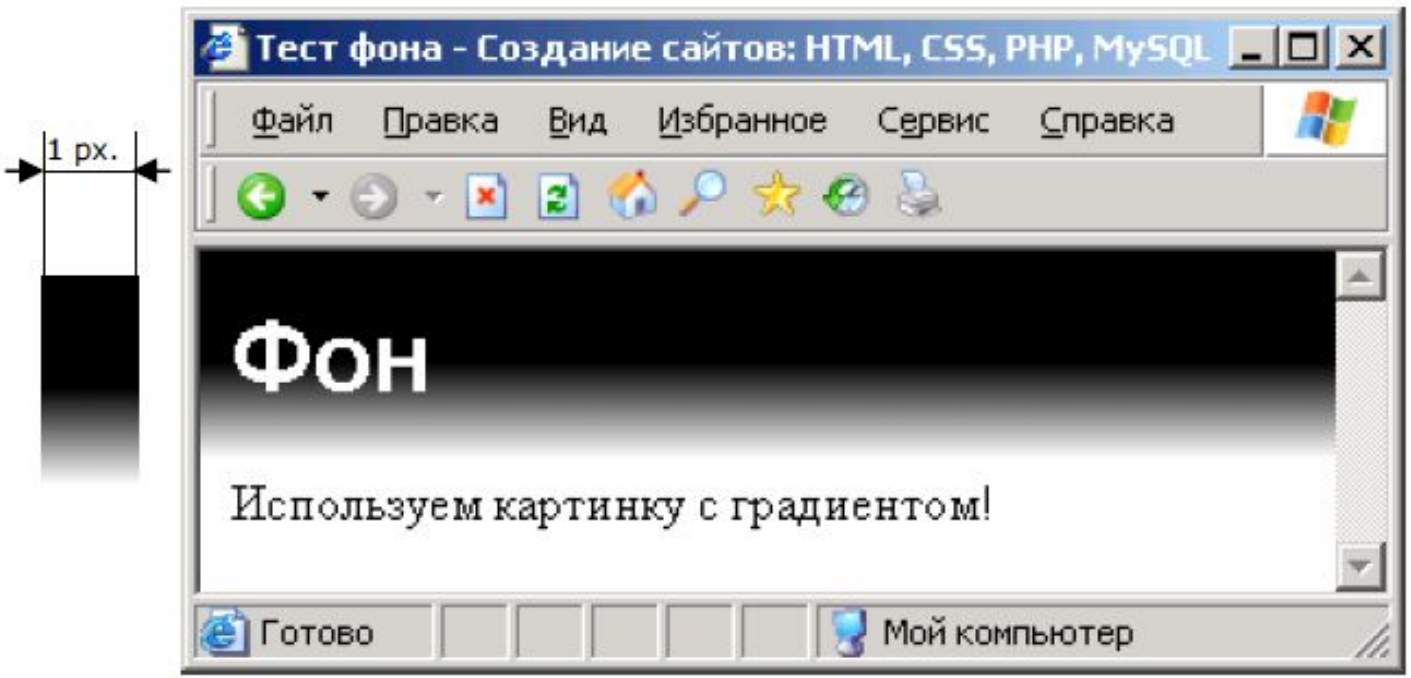
Рисунок 2.6. «Шахматный» фон

```
<html>
<head>
<title>Тест фона №2</title>
</head>
<body background="checkmate.gif">
Страница с шахматным фоном.
</body>
</html>
```


Тест фона №2 - Создание сайтов: HTML, CSS, PHP, MySQL
Файл Правка Вид Избранное Сервис Справка
Страница с шахматным фоном.
Готово
Мой компьютер

Если взять картинку шириной 1 пиксель и высотой, например, 2000 пикселей на экране она будет размножаться только по горизонтали.

```
<html>
<head>
<title>Тест фона</title>
</head>
<body background="1px.gif">
<h1><font color="white" face="Arial">Фон</font></h1>
Используем картинку с градиентом!
</body>
</html>
```



Создание таблиц

Таблица в HTML – это совокупность данных, расположенных и связанных между собой при помощи ячеек, размещаемых в строках и колонках. Таблица заполняется данными построчно. Для вставки таблиц определено 3 основных тега. Содержимое ячеек помещается в теги **<td>…</td>**, которые, в свою очередь, помещаются в теги строк **<tr>…</tr>**, а они уже – в тег **<table>…</table>**.

Пример:

```
<table>
<tr> <td>1</td> <td>2</td> <td>3</td> </tr>
<tr> <td>4</td> <td>5</td> <td>6</td> </tr>
</table>
```

В браузере:

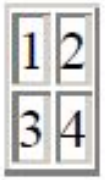
1	2	3
4	5	6

Атрибуты:

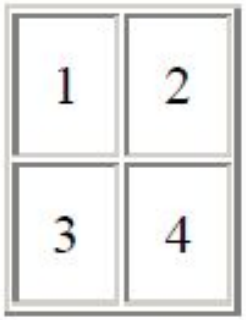
align="…" – определяет режим выравнивания таблицы относительно текста в строке
left – по левому краю
right – по правому краю
background="URL" – задает фоновый рисунок в таблице bgcolor="цвет"– цвет фона таблицы
border="N" – устанавливает толщину границ таблицы, равную N пикселей (0 для отключения)
bordercolor ="цвет" – цвет рамки
cellpadding="N" – размер поля вокруг содержимого каждой ячейки

Пример:

`cellpadding="0"`



`cellpadding="15"`



`cellspacing="N"` – размер свободного пространства между ячейками

Пример:

`cellspacing="0"`

`cellspacing="15"`

`width="N"` – ширина таблицы в пикселях или процентах от ширины окна

Объединение ячеек

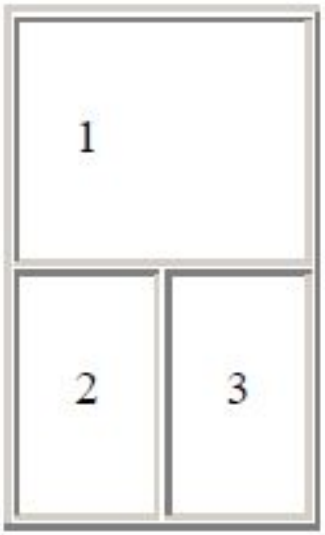
colspan="N" – растягивает ячейку на N столбцов влево

Пример:

HTML-код:

```
<TABLE cellpadding="15" border="1">
<TR><TD colspan="2">1</TD></TR>
<TR><TD>2</TD><TD>3</TD></TR>
</TABLE>
```

В браузере:



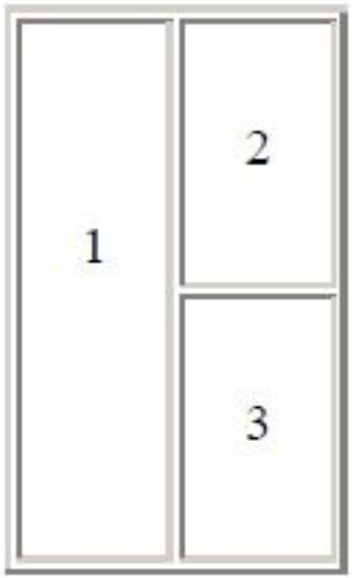
`rowspan="N"` – растягивает ячейку на N строк вниз

Пример:

HTML-код:

```
<TABLE cellpadding="15" border="1">
<TR><TD rowspan="2">1</TD><TD>2</TD></TR>
<TR><TD>3</TD></TR>
</TABLE>
```

В браузере:



Ширина таблицы

Если ширина таблицы изначально не задана, то она вычисляется исходя из содержимого ячеек.

```
<table border="1">
<tr><td>Ширина таблицы не задана!</td></tr>
</table>
```

Ширина таблицы не задана!

Максимальная ширина таблицы в таком случае равна ширине окна.

Если же ширина задана атрибутом width, то браузер расставляет переносы слов в тексте ячеек таким образом, чтобы соблюсти заданный размер.

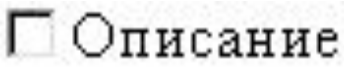
```
<table width="100" border="1">
<tr><td>Если задать атрибут width, текст начинает
переноситься по словам</td></tr>
</table>
```

Если задать атрибут width, текст начинает пе-реноситься по словам

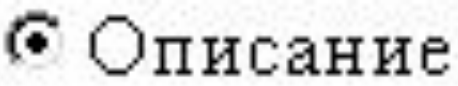
ФОРМЫ

Формы применяются для передачи данных от html-документа интерактивным элементам сайта, например сценариям CGI. Поместив в форму какие-либо значения, посетитель сервера нажимает мышью на соответствующую кнопку, после чего введенная им информация передается CGI-скрипту, который принимает управление процессом обработки данных. Эти данные претерпевают те или иные изменения, алгоритм которых записан в файле сценария CGI, например, встраиваются в другую web-страницу или передаются по электронной почте. Подобный принцип реализован в многочисленных электронных конференциях, досках объявлений, гостевых книгах и web-чатах Всемирной сети.

Элемент формы TEXT



Элемент формы CHECKBOX



Элеент формы RADIO

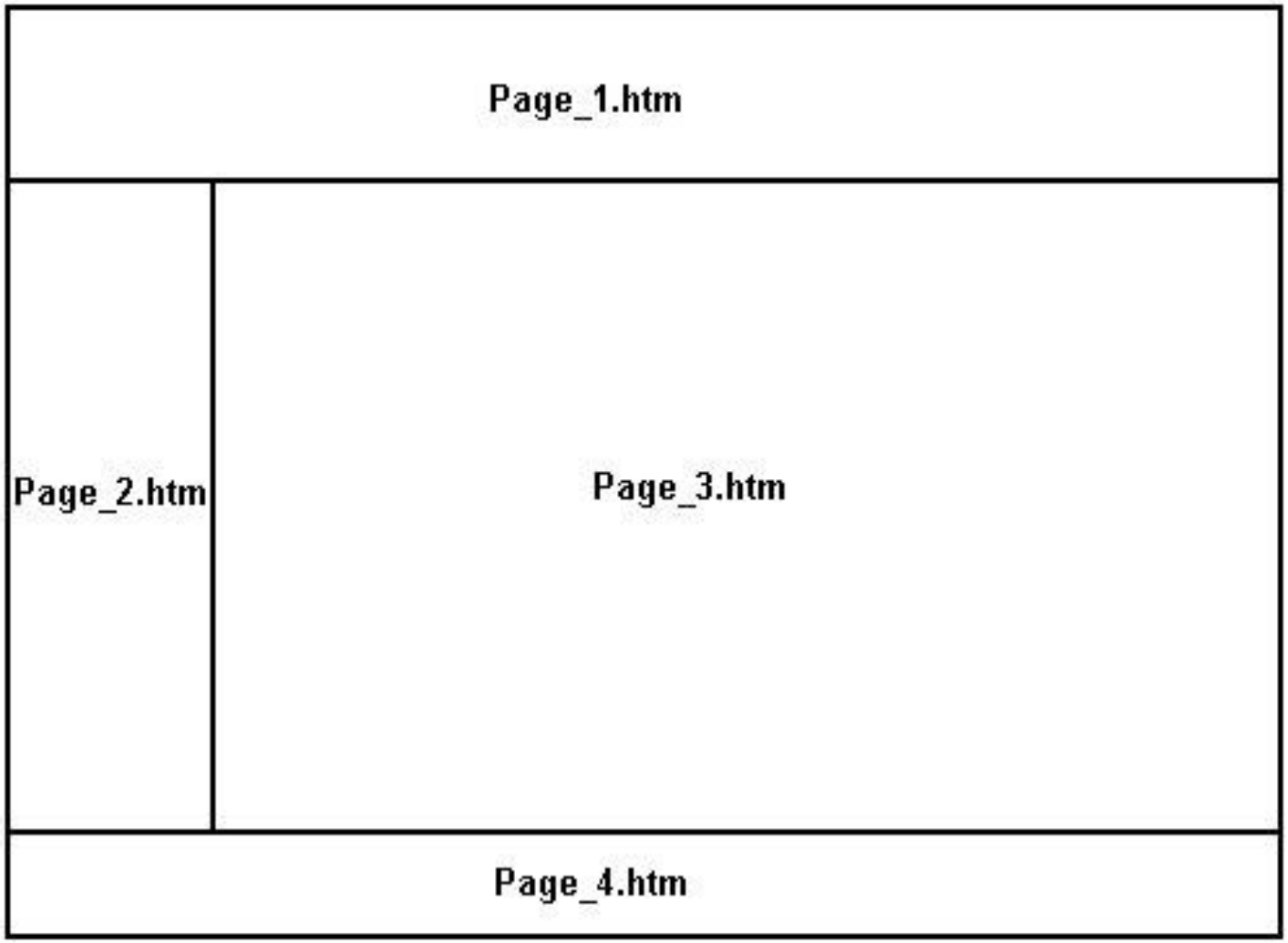
Элемент форм BUTTON

Элемент формы TEXTAREA

Элемент формы SELECT

ФРЕЙМЫ

Фреймы — способ организации структуры сайта, при котором web-страница дробится на ряд отдельных составляющих и «собирается» в главном окне браузера из нескольких независимых или вложенных окон. При таком представлении каждый компонент страницы является самостоятельным документом HTML и встраивается в ту область экрана, которая задается директивой <FRAMESET>. Данный способ применяется в основном для дробления web-страницы на логические разделы: например, в верхнем фрейме выводится рекламный баннер, в левом — элементы навигации, в правом — основной текст страницы, в нижнем — сообщение об авторских правах и адрес электронной почты разработчика ресурса. При этом нажатие на любую из навигационных кнопок приводит к изменению содержимого лишь одного окна, все остальные фреймы остаются без изменений. Если содержимое фрейма не умещается в видимые границы окна, браузер отображает полосы прокрутки.


Page_1.htm
Page_2.htm
Page_3.htm
Page_4.htm

Кодировки текста и специальные символы

Кодировка в HTML

Кодировка документа HTML задается в текстовом редакторе. Например, Блокнот в ОС Windows по умолчанию сохраняет текстовые файлы в кодировке Windows-1251.

Для того чтобы браузер правильно отобразил HTML-страницу, необходимо задать правильную кодировку в специальном теге **<meta>**[1].

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
```

или

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

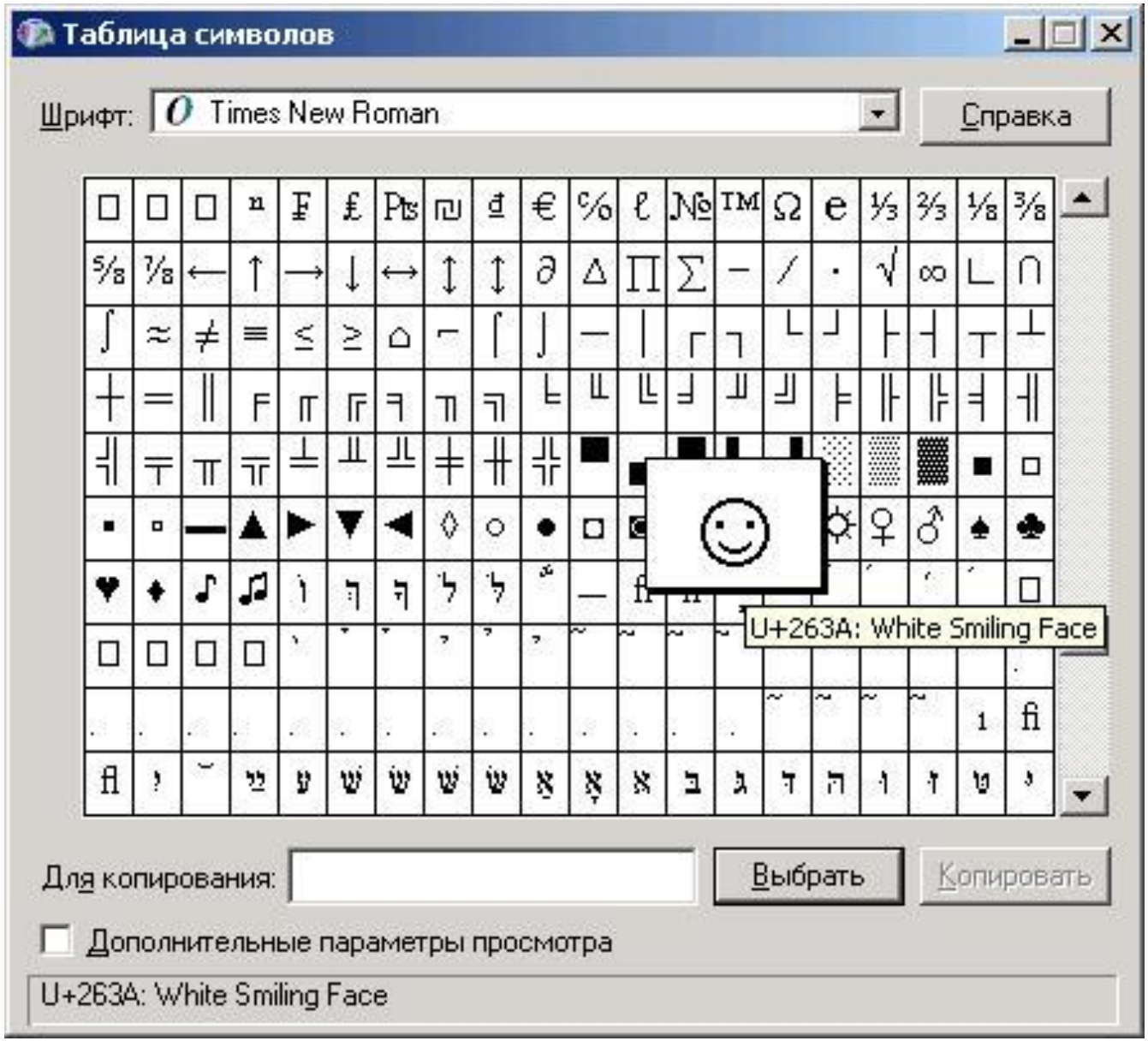
Если кодировка не будет указана, браузер попытается «угадать» ее, но не всегда это заканчивается успехом. Пользователь может выбрать кодировку самостоятельно в меню браузера (в Internet Explorer и Mozilla Firefox: Вид → Кодировка). При разработке сайта проблем с кодировкой следует избегать, т.к. большинство пользователей сразу же покинет страницу, увидев нечитаемый набор букв на экране.

Кодировки текста и специальные символы

Специальные символы в *HTML*

В HTML предусмотрен механизм вставки в документ любых символов Юни-код – подстановки или сущности (англ. entities). Подстановки позволяют употреблять символы, отсутствующие на клавиатуре или даже в используе-мой кодировке (т.е. даже используя кодировку Windows-1251 можно вставить букву греческого алфавита). Подстановки начинаются с символа амперсанда и записываются в виде &#DDDD; где DDDD – код символа в Юникоде в де-сятеричной системе счисления. Также можно записывать код в шестнадцате-ричной системе счисления в форме &#xHHHH; Для некоторых символов за-даны специальные названия – мнемоники. Например, знак копирайта © мо-жет быть задан кодом © или © или мнемоникой ©.

Кодировки текста и специальные символы



РАЗДЕЛ 2. ДИНАМИЧЕСКИЕ ЯЗЫКИ РАЗМЕТКИ ГИПЕРТЕКСТА

ЛЕКЦИЯ 5. ОСНОВЫ СОЗДАНИЯ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ

ЮТИ ТПУ

Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

CSS (Cascading Style Sheets, каскадные таблицы стилей). Стили представляют собой набор параметров, управляющих видом и положением элементов веб-страницы.

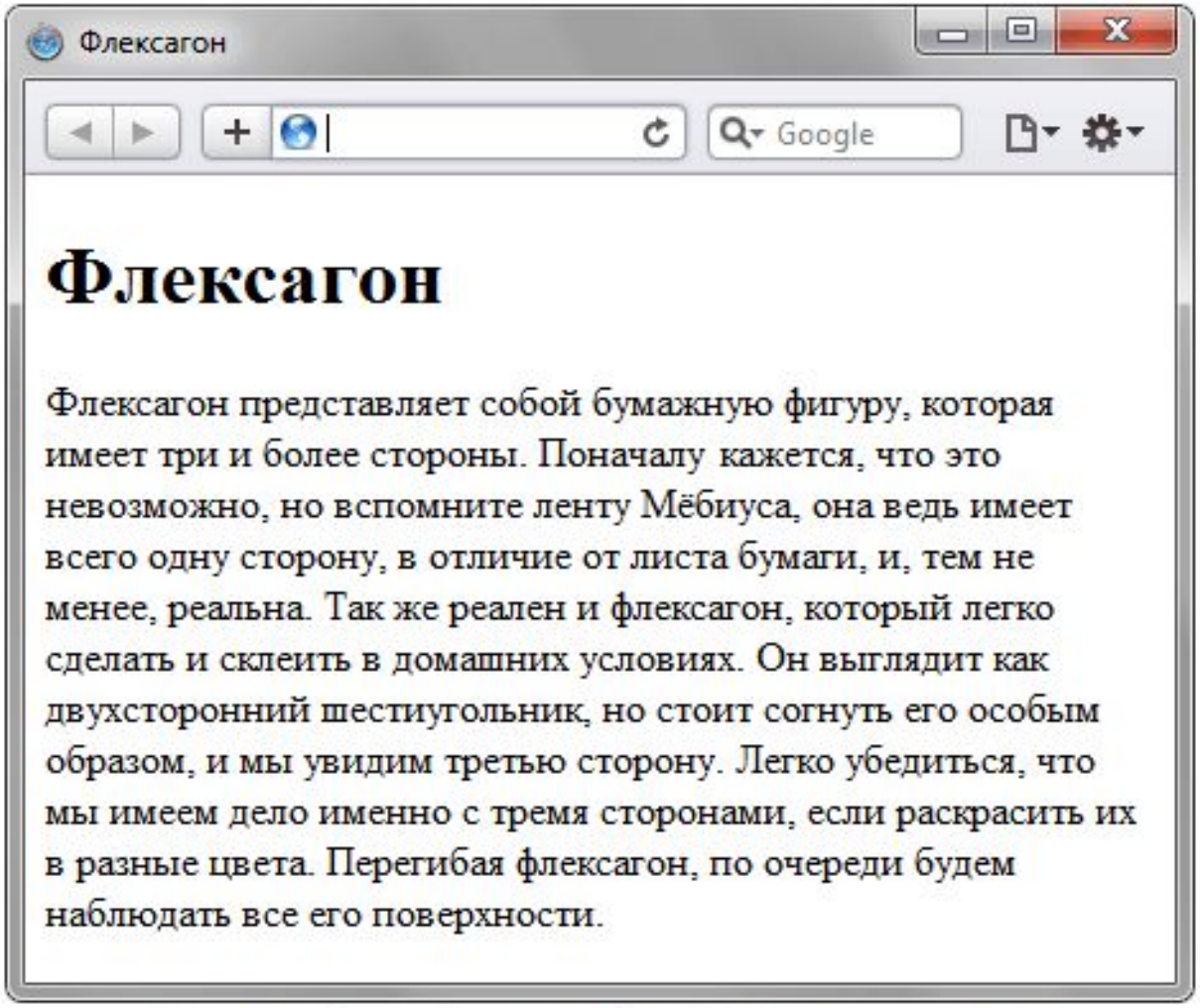


Рис. 1.1. Веб-страница, созданная только на HTML

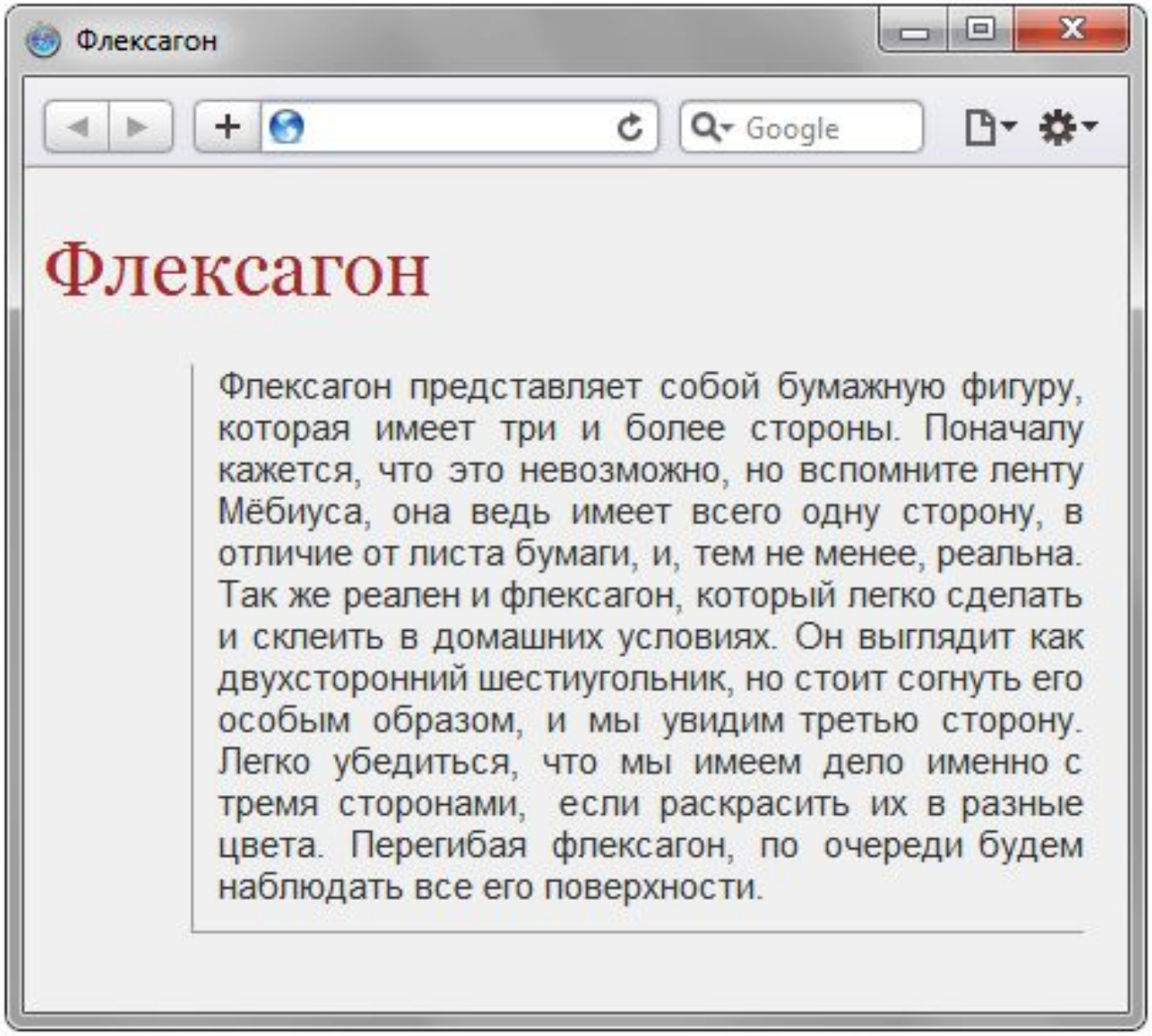


Рис. 1.2. Веб-страница, созданная на HTML и CSS

Пример 1.1. Исходный код документа

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>Флексагон</title>
  <meta charset="utf-8">
  <link rel="stylesheet" href="style.css">
 </head>
 <body>
  <h1>Флексагон</h1>
  <p>Флексагон представляет собой бумажную фигуру, которая имеет
   три и более стороны. Поначалу кажется, что это невозможно, но вспомните
   ленту Мёбиуса, она ведь имеет всего одну сторону, в отличие от листа  бумаги,
   и, тем не менее, реальна. Так же реален и флексагон, который легко сделать и
   склеить в домашних условиях. Он выглядит как двухсторонний шестиугольник, но
   стоит согнуть его особым образом, и мы увидим третью сторону. Легко убедиться,
   что мы имеем дело именно с тремя сторонами, если раскрасить их в разные цвета.
   Перегибая флексагон, по очереди будем наблюдать все его поверхности.</p>
 </body>
</html>
```

Пример 1.2. Содержимое стилевого файла style.css

```
body {
  font-family: Arial, Verdana,  sans-serif; /* Семейство шрифтов */
  font-size: 11pt; /* Размер основного шрифта в пунктах  */
  background-color: #f0f0f0; /* Цвет фона веб-страницы */
  color: #333; /* Цвет основного текста */
}
h1 {
  color: #a52a2a; /* Цвет заголовка */
  font-size: 24pt; /* Размер шрифта в пунктах */
  font-family: Georgia, Times, serif; /* Семейство шрифтов */
  font-weight: normal; /* Нормальное начертание текста  */
}
p {
  text-align: justify; /* Выравнивание по ширине */
  margin-left: 60px; /* Отступ слева в пикселах */
  margin-right: 10px; /* Отступ справа в пикселах */
  border-left: 1px solid #999; /* Параметры линии слева */
  border-bottom: 1px solid #999; /* Параметры линии снизу */
  padding-left: 10px; /* Отступ от линии слева до текста  */
  padding-bottom: 10px; /* Отступ от линии снизу до текста  */
}
```

В файле style.css описаны все параметры оформления таких тегов как <body>, <h1> и <p>. Заметим, что сами теги в коде HTML пишутся как обычно.

Поскольку на файл со стилем можно ссылаться из любого веб-документа, это приводит в итоге к сокращению объёма повторяющихся данных. А благодаря разделению кода и оформления повышается гибкость управления видом документа и скорость работы над сайтом.

CSS представляет собой свой собственный язык, который совпадает с HTML только некоторыми значениями, например способом определения цвета.

Типы стилей

Различают несколько типов стилей, которые могут совместно применяться к одному документу. Это стиль браузера, стиль автора и стиль пользователя.

Стиль браузера

Оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей. Например, заголовок страницы, формируемый тегом <H1>, в большинстве браузеров выводится шрифтом с засечками размером 24 пункта.

Стиль автора

Стиль, который добавляет к документу его разработчик.

Стиль пользователя

Это стиль, который может включить пользователь сайта через настройки браузера. Такой стиль имеет более высокий приоритет и переопределяет исходное оформление документа. В браузере Internet Explorer подключение стиля пользователя делается через меню Сервис > Свойство обозревателя > Кнопка «Оформление».

Указанные типы стилей могут спокойно существовать друг с другом, если они не пытаются изменить вид одного элемента. В случае возникновения противоречия вначале имеет приоритет стиль пользователя, затем стиль автора и последним идёт стиль браузера.

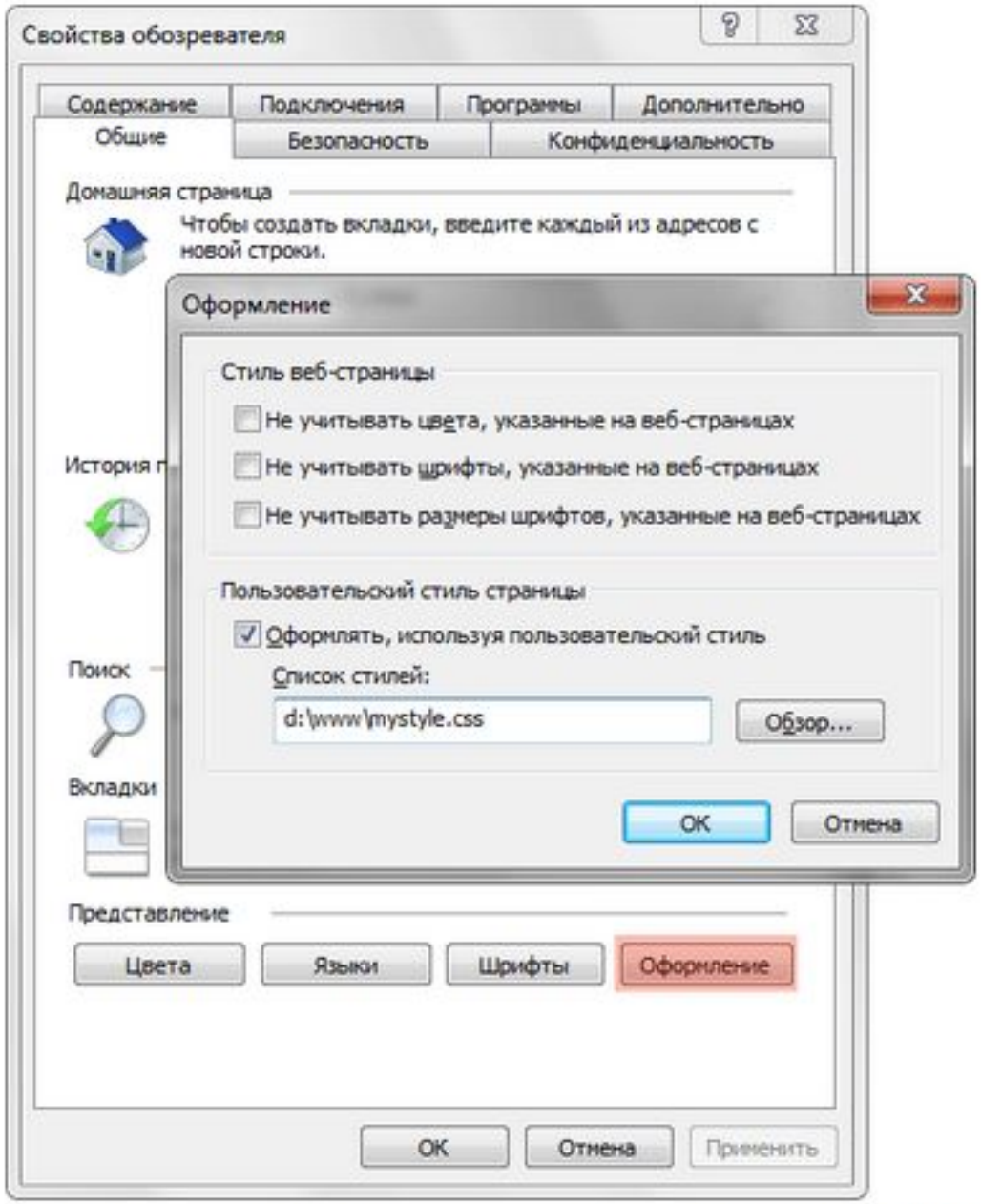


Рис. 1.3. Подключение стиля пользователя в браузере Internet Explorer

В браузере Opera аналогичное действие происходит через команду Инструменты > Общие настройки > Вкладка «Расширенные» > Содержимое > Кнопка «Параметры стиля».

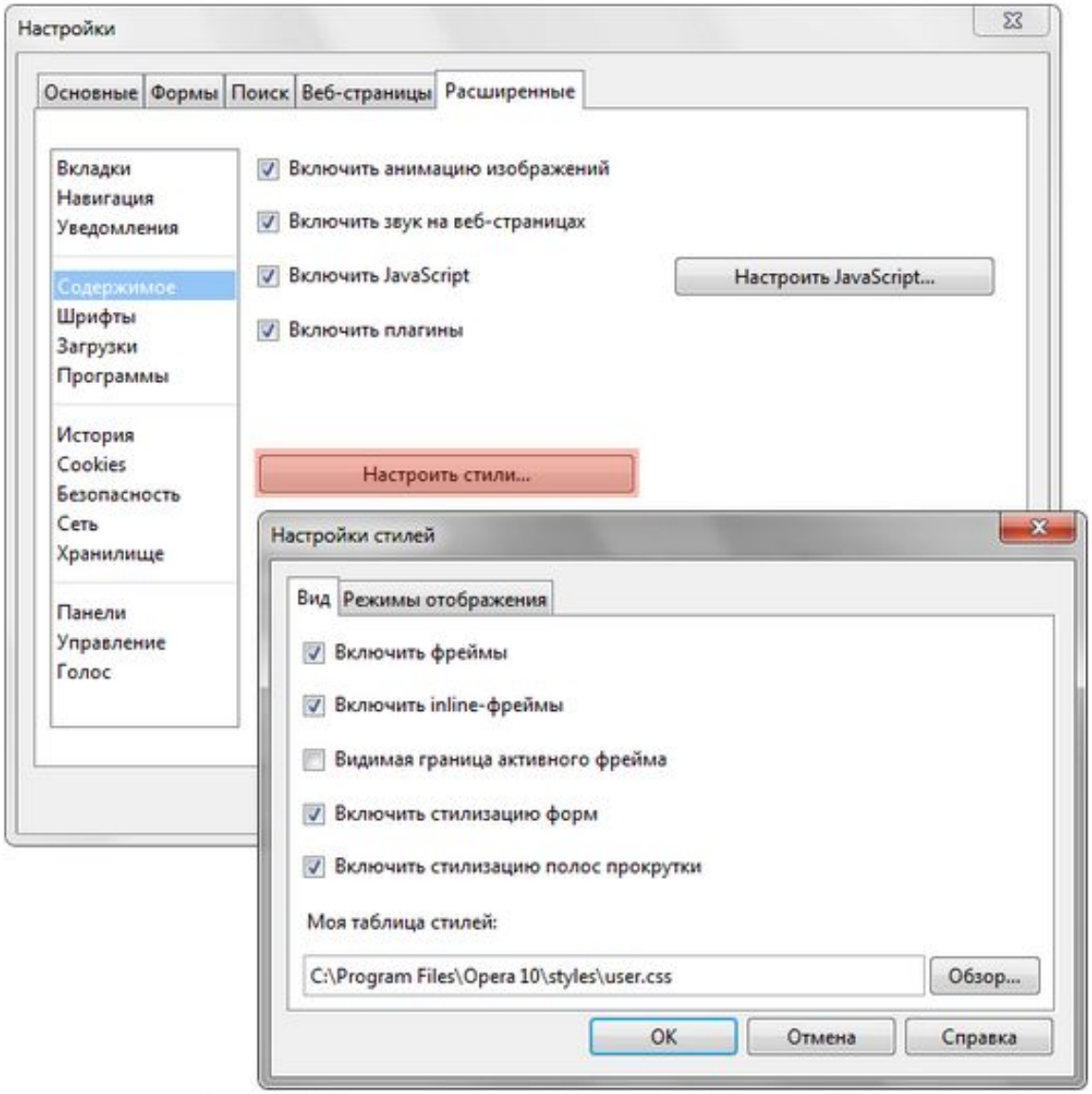


Рис. 1.4. Подключение стиля пользователя в браузере Opera

Преимущества стилей

Стили являются удобным, практичным и эффективным инструментом при вёрстке веб-страниц и оформления текста, ссылок, изображений и других элементов.

Разграничение кода и оформления

Идея о том, чтобы код HTML был свободен от элементов оформления вроде установки цвета, размера шрифта и других параметров, стара как мир. В идеале, веб-страница должна содержать только теги логического форматирования, а вид элементов задаётся через стили. При подобном разделении работа над дизайном и версткой сайта может вестись параллельно.

Разное оформление для разных устройств

С помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, планшета и др. Например, на экране монитора отображать страницу в одном оформлении, а при её печати — в другом. Эта возможность также позволяет скрывать или показывать некоторые элементы документа при отображении на разных устройствах.

Расширенные по сравнению с HTML способы оформления элементов

В отличие от HTML стили имеют гораздо больше возможностей по оформлению элементов веб-страниц. Простыми средствами можно изменить цвет фона элемента, добавить рамку, установить шрифт, определить размеры, положение и многое другое.

Преимущества стилей

Ускорение загрузки сайта

При хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из кэша браузера. За счёт кэширования и того, что стили хранятся в отдельном файле, уменьшается код веб-страниц и снижается время загрузки документов. Кэшем называется специальное место на локальном компьютере пользователя, куда браузер сохраняет файлы при первом обращении к сайту. При следующем обращении к сайту эти файлы уже не скачиваются по сети, а берутся с локального диска. Такой подход позволяет существенно повысить скорость загрузки веб-страниц.

Единое стилевое оформление множества документов

Сайт это не просто набор связанных между собой документов, но и одинаковое расположение основных блоков, и их вид. Применение единообразного оформления заголовков, основного текста и других элементов создает преемственность между страницами и облегчает пользователям работу с сайтом и его восприятие в целом. Разработчикам же использование стилей существенно упрощает проектирование дизайна.

Централизованное хранение

Стили, как правило, хранятся в одном или нескольких специальных файлах, ссылка на которые указывается во всех документах сайта. Благодаря этому удобно править стиль в одном месте, при этом оформление элементов автоматически меняется на всех страницах, которые связаны с указанным файлом. Вместо того чтобы модифицировать десятки HTML-файлов, достаточно отредактировать один файл со стилем и оформление нужных документов сразу же поменяется.

Способы добавления стилей на страницу

Для добавления стилей на веб-страницу существует несколько способов, которые различаются своими возможностями и назначением.

Связанные стили

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле, как правило, с расширением css, а для связывания документа с этим файлом применяется тег <link>. Данный тег помещается в контейнер <head>, как показано в примере 3.1.

Пример 3.1. Подключение связанных стилей

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Стили</title>
  <link rel="stylesheet" href="http://htmlbook.ru/mysite.css">
  <link rel="stylesheet" href="http://www.htmlbook.ru/main.css">
 </head>
 <body>
   <h1>Заголовок</h1>
   <p>Текст</p>
 </body>
</html>
```

Значение атрибута тега <link> — rel остаётся неизменным независимо от кода, как приведено в данном примере. Значение href задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицу стилей, которая находится на другом сайте.

Содержимое файла mysite.css подключаемого посредством тега <link> приведено в примере 3.2.

Пример 3.2. Файл со стилем

```
H1 {
  color: #000080;
  font-size: 200%;
  font-family: Arial, Verdana, sans-serif;
  text-align: center;
}
P {
  padding-left: 20px;
}
```

Как видно из данного примера, файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т. е. таким способом в полной мере реализуется принцип разделения кода и оформления сайта. Поэтому использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

Глобальные стили

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает предыдущему, но также позволяет хранить стили в одном месте, в данном случае прямо на той же странице с помощью контейнера <style>, как показано в примере 3.3.

Пример 3.3. Использование глобального стиля

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Глобальные стили</title>
  <style>
   H1 {
    font-size: 120%;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #333366;
   }
  </style>
 </head>
 <body>
  <h1>Hello, world!</h1>
 </body>
</html>
```

В данном примере задан стиль тега <h1>, который затем можно повсеместно использовать на данной веб-странице

Внутренние стили

Внутренний или встроенный стиль является по существу расширением для одиночного тега используемого на текущей веб-странице. Для определения стиля используется атрибут style, а его значением выступает набор стилевых правил (пример 3.4).

Пример 3.4. Использование внутреннего стиля

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Внутренние стили</title>
 </head>
 <body>
  <p style="font-size: 120%; font-family: monospace; color: #cd66cc">Пример текста</p>
</body>
</html>
```

В данном примере стиль тега <p> задаётся с помощью атрибута style, в котором через точку с запятой перечисляются стилевые свойства (рис. 3.2).

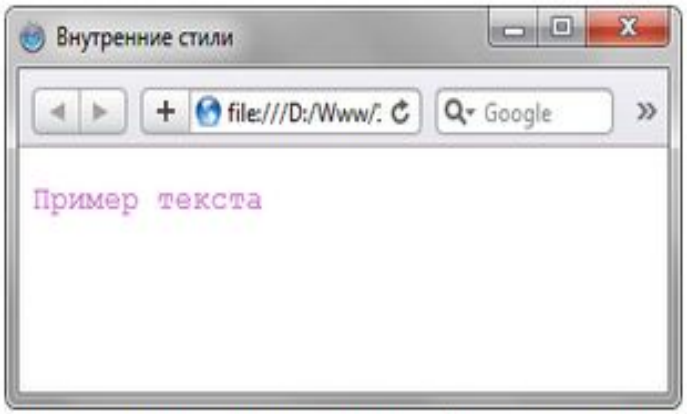


Рис. 3.2. Использование внутренних стилей для изменения вида текста

Все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии. Первым имеет приоритет внутренний стиль, затем глобальный стиль и в последнюю очередь связанный стиль. В примере 3.5 применяется сразу два метода добавления стиля в документ.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Подключение стиля</title>
  <style>
   H1 {
    font-size: 120%;
    font-family: Arial, Helvetica, sans-serif;
    color: green;
   }
  </style>
 </head>
 <body>
   <h1 style="font-size: 36px; font-family: Times, serif; color: red">Заголовок 1</h1>
   <h1>Заголовок 2</h1>
 </body>
</html>
```

В данном примере первый заголовок задаётся красным цветом размером 36 пикселов с помощью внутреннего стиля, а следующий — зелёным цветом через таблицу глобальных стилей (рис. 3.3).

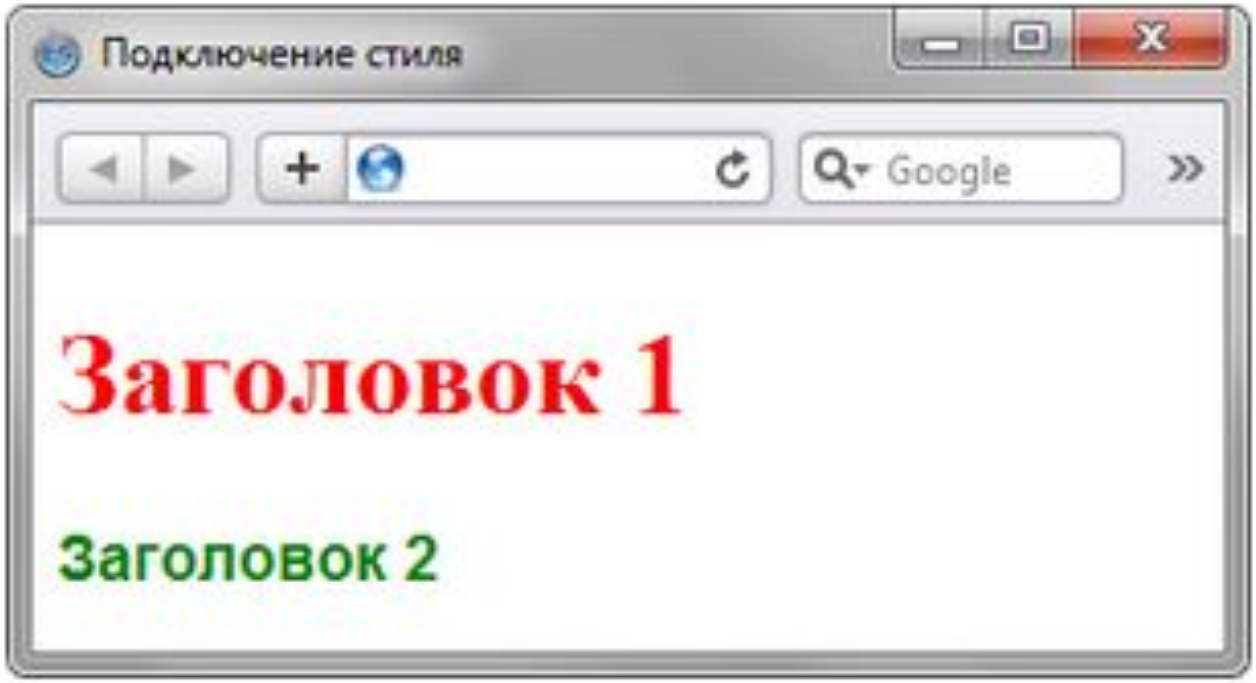


Рис. 3.3. Результат применения стилей

Импорт CSS

В текущую стилевую таблицу можно импортировать содержимое CSS-файла с помощью команды @import. Этот метод допускается использовать совместно со связанными или глобальными стилями, но никак не с внутренними стилями. Общий синтаксис следующий.

```
@import url("имя файла") типы носителей;
@import "имя файла" типы носителей;
```

После ключевого слова @import указывается путь к стилевому файлу одним из двух приведенных способов — с помощью url или без него. В примере 3.6 показано, как можно импортировать стиль из внешнего файла в таблицу глобальных стилей.

В данном примере показано подключение файла header.css, который расположен в папке style.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Импорт</title>
  <style>
   @import url("style/header.css");
   H1 {
    font-size: 120%;
    font-family: Arial, Helvetica, sans-serif;
    color: green;
   }
  </style>
 </head>
 <body>
   <h1>Заголовок 1</h1>
   <h2>Заголовок 2</h2>
 </body>
</html>
```

Типы носителей

Широкое развитие различных платформ и устройств заставляет разработчиков делать под них специальные версии сайтов, что достаточно трудоёмко и проблематично. Вместе с тем, времена и потребности меняются, и создание сайта для различных устройств является неизбежным и необходимым звеном его развития. С учетом этого в CSS введено понятие типа носителя, когда стиль применяется только для определённого устройства. В табл. 4.1 перечислены некоторые типы носителей.

Тип	Описание
all	Все типы. Это значение используется по умолчанию.
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры.
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
handheld	Наладонные компьютеры и аналогичные им аппараты.
print	Печатающие устройства вроде принтера.
projection	Проектор.
screen	Экран монитора.
tv	Телевизор.

В CSS для указания типа носителей применяются команды @import и @media, с помощью которых можно определить стиль для элементов в зависимости от того, выводится документ на экран или на принтер.

При импортировании стиля через команду @import тип носителя указывается после адреса файла. При этом допускается задавать сразу несколько типов, упоминая их через запятую, как показано в примере 4.1.

Пример 4.1. Импорт стилевого файла

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Импорт стиля</title>
  <style>
    @import "/style/main.css" screen; /* Стиль для вывода результата на монитор */
    @import "/style/smart.css" print, handheld; /* Стиль для печати и смартфона */
  </style>
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере импортируется два файла — main.css предназначен для изменения вида документа при его просмотре на экране монитора, и smart.css — при печати страницы и отображении на смартфоне.

Команда @media позволяет указать тип носителя для глобальных или связанных стилей и в общем случае имеет следующий синтаксис.

```
@media тип носителя 1 {
  Описание стиля для типа носителя 1
}
@media тип носителя 2 {
  Описание стиля для типа носителя 2
}
```

После ключевого слова @media идёт один или несколько типов носителя, перечисленных в табл. 4.1, если их больше одного, то они разделяются между собой запятой. После чего следуют обязательные фигурные скобки, внутри которых идёт обычное описание стилевых правил. В примере 4.2 показано, как задать разный стиль для печати и отображения на мониторе.

В данном примере вводится два стиля — один для изменения вида элементов при их обычном отображении в браузере, а второй — при выводе страницы на печать. При этом облик документа для разных носителей может сильно различаться между собой, например, как это показано на рис. 4.1 и рис. 4.2.

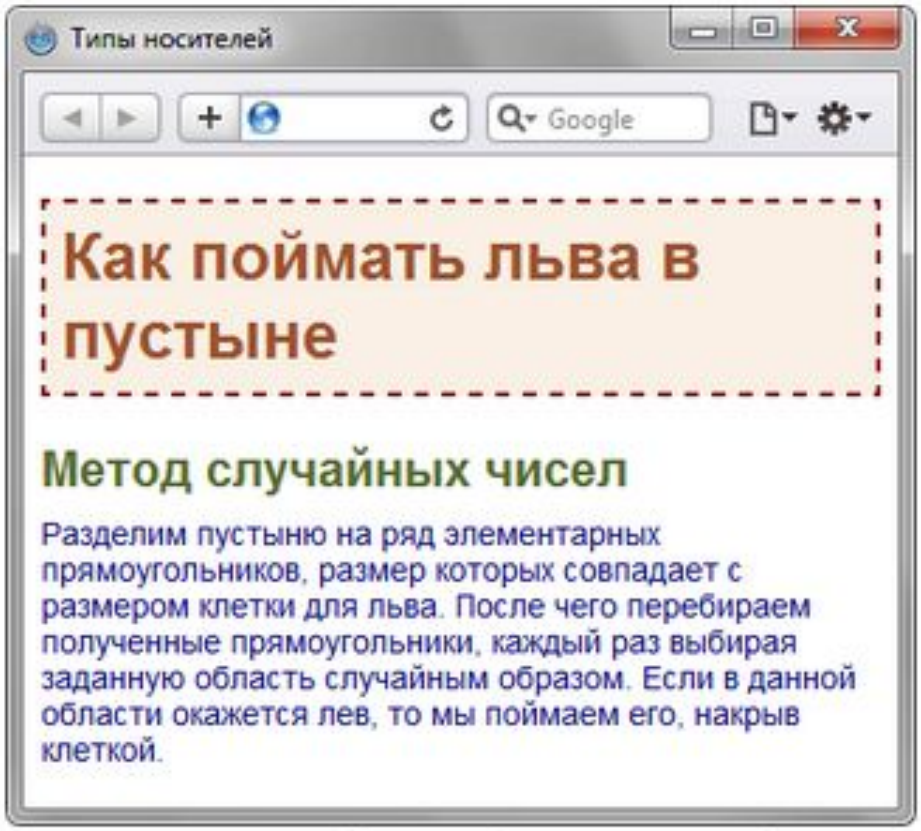


Рис. 4.1. Страница для отображения в окне браузера

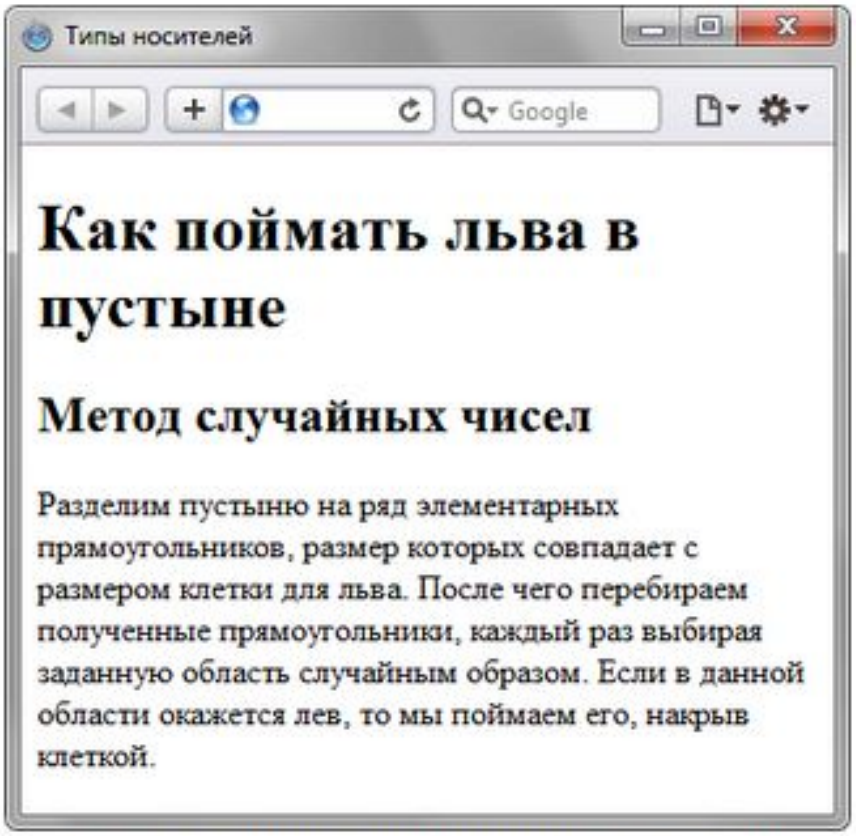


Рис. 4.2. Страница, предназначенная для печати

Просмотреть документ, у которого CSS установлен как тип print можно, если распечатать определенную страницу или воспользовавшись предварительным просмотром в браузере (Файл > Предварительный просмотр). Или пойти на хитрость и временно заменить print на screen, чтобы отобразить итог в браузере. Именно так был получен рис. 4.2.

Команда @media применяется в основном для формирования одного стилевого файла, который разбит на блоки по типу устройств. Иногда же имеет смысл создать несколько разных CSS-файлов — один для печати, другой для отображения в браузере — и подключать их к документу по мере необходимости. В подобном случае следует воспользоваться тегом <link> с атрибутом media, значением которого выступают все те же типы, перечисленные в табл. 4.1.

В примере 4.3 показано, как создавать ссылки на CSS-файлы, которые предназначены для разных типов носителей.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Разные носители</title>
  <link media="print, handheld" rel="stylesheet" href="print.css">
  <link media="screen" rel="stylesheet" href="main.css">
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере используются две таблицы связанных стилей, одна для отображения в браузере, а вторая — для печати документа и его просмотре на смартфоне. Хотя на страницу загружаются одновременно два разных стиля, применяются они только для определённых устройств.

Аналогично можно использовать и глобальные стили, добавляя атрибут media к тегу <style> (пример 4.4).

В данном примере ширина для устройств типа handheld ограничена размером 320 пикселов.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Разные носители</title>
  <style media="handheld">
   BODY {
    width: 320px; /* Ширина страницы */
   }
  </style>
 </head>
 <body>
   <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem
   nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat
   volutpat. </p>
 </body>
</html>
```

Базовый синтаксис CSS

Стилевые правила записываются в своём формате, отличном от HTML. Основным понятием выступает селектор — это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид.

Вначале пишется имя селектора, например, TABLE, это означает, что все стилевые параметры будут применяться к тегу <table>, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить.

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика.

В примере 5.1 показаны две разновидности оформления селекторов и их правил.

Пример 5.1. Использование стилей

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Заголовки</title>
  <style>
   h1 { color: #a6780a; font-weight: normal; }
   h2 {
    color: olive;
    border-bottom: 2px solid black;
   }
  </style>
 </head>
 <body>

  <h1>Заголовок 1</h1>
  <h2>Заголовок 2</h2>

 </body>
</html>
```

Форма записи

Для селектора допускается добавлять каждое стилевое свойство и его значение по отдельности, как это показано в примере 5.2.

Пример 5.2. Расширенная форма записи

```
td { background: olive; }
td { color: white; }
td { border: 1px solid black; }
```

Однако такая запись не очень удобна. Приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Поэтому пишите все свойства для каждого селектора вместе. Указанный набор записей в таком случае получит следующий вид (пример 5.3).

Пример 5.3. Компактная форма записи

```
td {
  background: olive;
  color: white;
  border: 1px solid black;
}
```

Эта форма записи более наглядная и удобная в использовании.

Имеет приоритет значение, указанное в коде ниже

Если для селектора вначале задаётся свойство с одним значением, а затем то же свойство, но уже с другим значением, то применяться будет то значение, которое в коде установлено ниже (пример 5.4).

Пример 5.4. Разные значения у одного свойства

```
p { color: green; }
p { color: red; }
```

В данном примере для селектора p цвет текста вначале установлен зелёным, а затем красным. Поскольку значение red расположено ниже, то оно в итоге и будет применяться к тексту.

На самом деле такой записи лучше вообще избегать и удалять повторяющиеся значения. Но подобное может произойти случайно, например, в случае подключения разных стилевых файлов, в которых содержатся одинаковые селекторы.

Значения

У каждого свойства может быть только соответствующее его функции значение. Например, для color, который устанавливает цвет текста, в качестве значений недопустимо использовать числа.

Комментарии

Комментарии нужны, чтобы делать пояснения по поводу использования того или иного стилевого свойства, выделять разделы или писать свои заметки. Комментарии позволяют легко вспоминать логику и структуру селекторов, и повышают разборчивость кода. Вместе с тем, добавление текста увеличивает объём документов, что отрицательно сказывается на времени их загрузки. Поэтому комментарии обычно применяют в отладочных или учебных целях, а при выкладывании сайта в сеть их стирают.

Чтобы пометить, что текст является комментарием, применяют следующую конструкцию /* ... */ (пример 5.5).

Пример 5.5. Комментарии в CSS-файле

```
/*
  Стиль для сайта htmlbook.ru
  Сделан для ознакомительных целей
*/

div {
  width: 200px; /* Ширина блока */
  margin: 10px; /* Поля  вокруг элемента */
  float: left; /* Обтекание по правому краю */
}
```

Значения стилевых свойств

Всё многообразие значений стилевых свойств может быть сведено к определённому типу: строка, число, проценты, размер, цвет, адрес или ключевое слово.

Строки

Любые строки необходимо брать в двойные или одинарные кавычки. Если внутри строки требуется оставить одну или несколько кавычек, то можно комбинировать типы кавычек или добавить перед кавычкой слэш (пример 6.1).

Пример 6.1. Допустимые строки

```
'Гостиница "Турист"'
"Гостиница 'Турист'"
"Гостиница \"Турист\""
```

В данном примере в первой строке применяются одинарные кавычки, а слово «Турист» взято в двойные кавычки. Во второй строке всё с точностью до наоборот, в третьей же строке используются только двойные кавычки, но внутренние экранированы с помощью слэша.

Числа

Значением может выступать целое число, содержащее цифры от 0 до 9 и десятичная дробь, в которой целая и десятичная часть разделяются точкой (пример 6.2).

Пример 6.2. Числа в качестве значений

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Числа</title>
  <style>
   p {
    font-weight: 600; /* Жирное начертание */
    line-height: 1.2; /* Межстрочный интервал */
   }
  </style>
 </head>
 <body>
  <p>Пример текста</p>
 </body>
</html>
```

Если в десятичной дроби целая часть равна нулю, то её разрешается не писать. Запись .7 и 0.7 равнозначна.

Проценты

Процентная запись обычно применяется в тех случаях, когда надо изменить значение относительно родительского элемента или когда размеры зависят от внешних условий. Так, ширина таблицы 100% означает, что она будет подстраиваться под размеры окна браузера и меняться вместе с шириной окна (пример 6.3).

Пример 6.3. Процентная запись

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Ширина в процентах</title>
  <style>
   TABLE {
    width: 100%; /* Ширина таблицы в процентах */
    background: #f0f0f0; /* Цвет фона */
   }
  </style>
 </head>
 <body>
   <table>
    <tr><td>Содержимое таблицы</td></tr>
   </table>
 </body>
</html>
```

Проценты не обязательно должны быть целым числом, допускается использовать десятичные дроби, вроде значения 56.8%.

Размеры

Для задания размеров различных элементов, в CSS используются абсолютные и относительные единицы измерения. Абсолютные единицы не зависят от устройства вывода, а относительные единицы определяют размер элемента относительно значения другого размера.

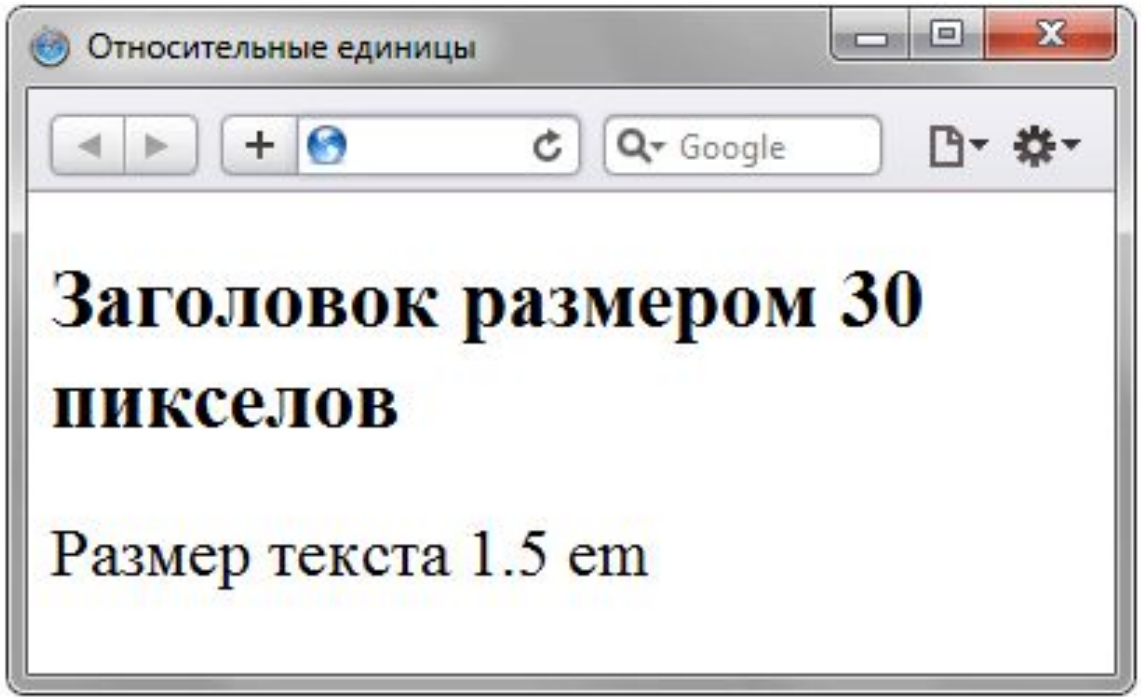
Относительные единицы

Относительные единицы обычно используют для работы с текстом, либо когда надо вычислить процентное соотношение между элементами. В табл. 6.1 перечислены основные относительные единицы.

Единица	Описание
em	Размер шрифта текущего элемента
ex	Высота символа x
px	Пиксел
%	Процент

Пример 6.4. Использование относительных единиц

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Относительные единицы</title>
  <style>
    H1 { font-size: 30px; }
    P { font-size: 1.5em; }
  </style>
 </head>
 <body>
  <h1>Заголовок размером 30 пикселов</h1>
  <p>Размер текста 1.5 em</p>
 </body>
</html>
```



Абсолютные единицы

Абсолютные единицы применяются реже, чем относительные и обычно при работе с текстом. В табл. 6.2 перечислены основные абсолютные единицы.

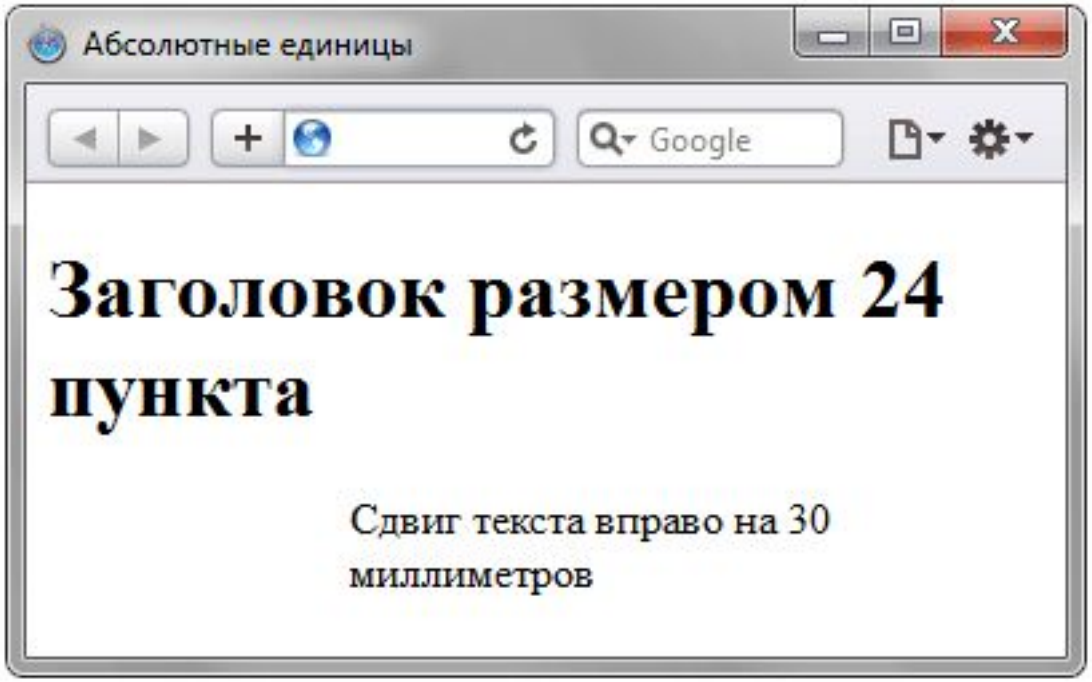
Табл. 6.2. Абсолютные единицы измерения

Единица	Описание
in	Дюйм (1 дюйм равен 2,54 см)
cm	Сантиметр
mm	Миллиметр
pt	Пункт (1 пункт равен 1/72 дюйма)
pc	Пика (1 пика равна 12 пунктам)

Самой, пожалуй, распространенной единицей является пункт, который используется для указания размера шрифта. Хотя мы привыкли измерять все в миллиметрах и подобных единицах, пункт, пожалуй, единственная величина из не метрической системы измерения, которая используется у нас повсеместно. И все благодаря текстовым редакторам и издательским системам. В примере 6.5 показано использование пунктов и миллиметров.

Пример 6.5. Использование абсолютных единиц

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Абсолютные единицы</title>
  <style>
   H1 { font-size: 24pt; }
   P { margin-left: 30mm; }
  </style>
 </head>
 <body>
   <h1>Заголовок размером 24 пункта</h1>
   <p>Сдвиг текста вправо на 30
 </body>
</html>
```



Цвет

Цвет в стилях можно задавать тремя способами: по шестнадцатеричному значению, по названию и в формате RGB.

С помощью RGB

Можно определить цвет, используя значения красной, зелёной и синей составляющей в десятичном исчислении. Значение каждого из трех цветов может принимать значения от 0 до 255. Также можно задавать цвет в процентном отношении. Вначале указывается ключевое слово rgb, а затем в скобках, через запятую указываются компоненты цвета, например rgb(255, 0, 0) или rgb(100%, 20%, 20%).

Ключевые слова

В качестве значений активно применяются ключевые слова, которые определяют желаемый результат действия стилевых свойств. Ключевые слова пишутся без кавычек.

```
Правильно: P { text-align: right; }
Неверно: P { text-align: "right"; }
```

Селекторы тегов

В качестве селектора может выступать любой тег HTML, для которого определяются правила форматирования, такие как: цвет, фон, размер и т. д. Правила задаются в следующем виде.

```
Тег { свойство1: значение; свойство2: значение; ... }
```

Вначале указывается имя тега, оформление которого будет переопределено, заглавными или строчными символами не имеет значения. Внутри фигурных скобок пишется стилевое свойство, а после двоеточия — его значение. Набор свойств разделяется между собой точкой с запятой и может располагаться как в одну строку, так и в несколько (пример 7.1).

```
<head>
 <meta charset="utf-8">
 <title>Селекторы тегов</title>
 <style>
  P {
   text-align: justify; /* Выравнивание по ширине */
   color: green; /* Зеленый цвет текста */
  }
 </style>
</head>
<body>

 <p>Более эффективным способом ловли льва в пустыне
 является метод золотого сечения. При его использовании пустыня делится
 на две неравные части, размер которых подчиняется правилу золотого
 сечения.</p>
```

В данном примере изменяется цвет и выравнивание текста абзаца. Стиль будет применяться только к тексту, который располагается внутри контейнера <p>.

Следует понимать, что хотя стиль можно применить к любому тегу, результат будет заметен только для тегов, которые непосредственно отображаются в контейнере <body>.

Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

```
Тег.Имя класса { свойство1: значение; свойство2: значение; ... }
```

Внутри стиля вначале пишется желаемый тег, а затем, через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определённым классом, к тегу добавляется атрибут class="Имя класса" (пример 8.1).

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Классы</title>
  <style>
   P { /* Обычный абзац */
    text-align: justify; /* Выравнивание текста по ширине */
   }
   P.cite { /* Абзац с классом cite */
    color: navy; /* Цвет текста */
    margin-left: 20px; /* Отступ слева */
    border-left: 1px solid navy; /* Граница слева от текста */
    padding-left: 15px; /* Расстояние от линии до текста */
   }
  </style>
 </head>
 <body>
  <p>Для искусственного освещения помещения применяются люминесцентные лампы.
     Они отличаются высокой световой отдачей, продолжительным сроком службы,
     малой яркостью светящейся поверхности, близким к естественному спектральным
     составом излучаемого света, что обеспечивает хорошую цветопередачу.</p>
  <p class="cite">Для исключения засветки экрана дисплея световыми потоками
     оконные проемы снабжены светорассеивающими шторами.</p>
 </body>
</html>
```

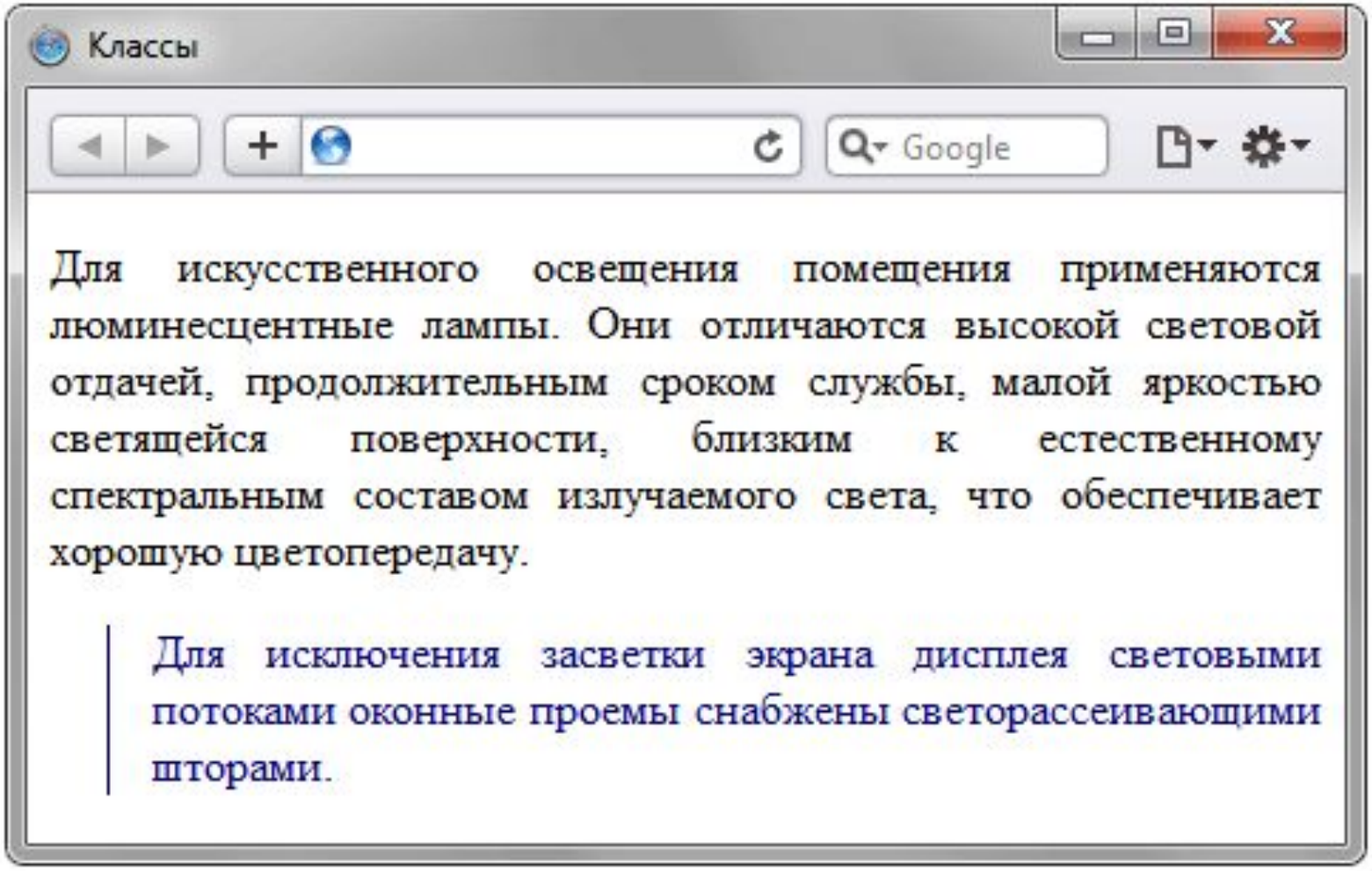
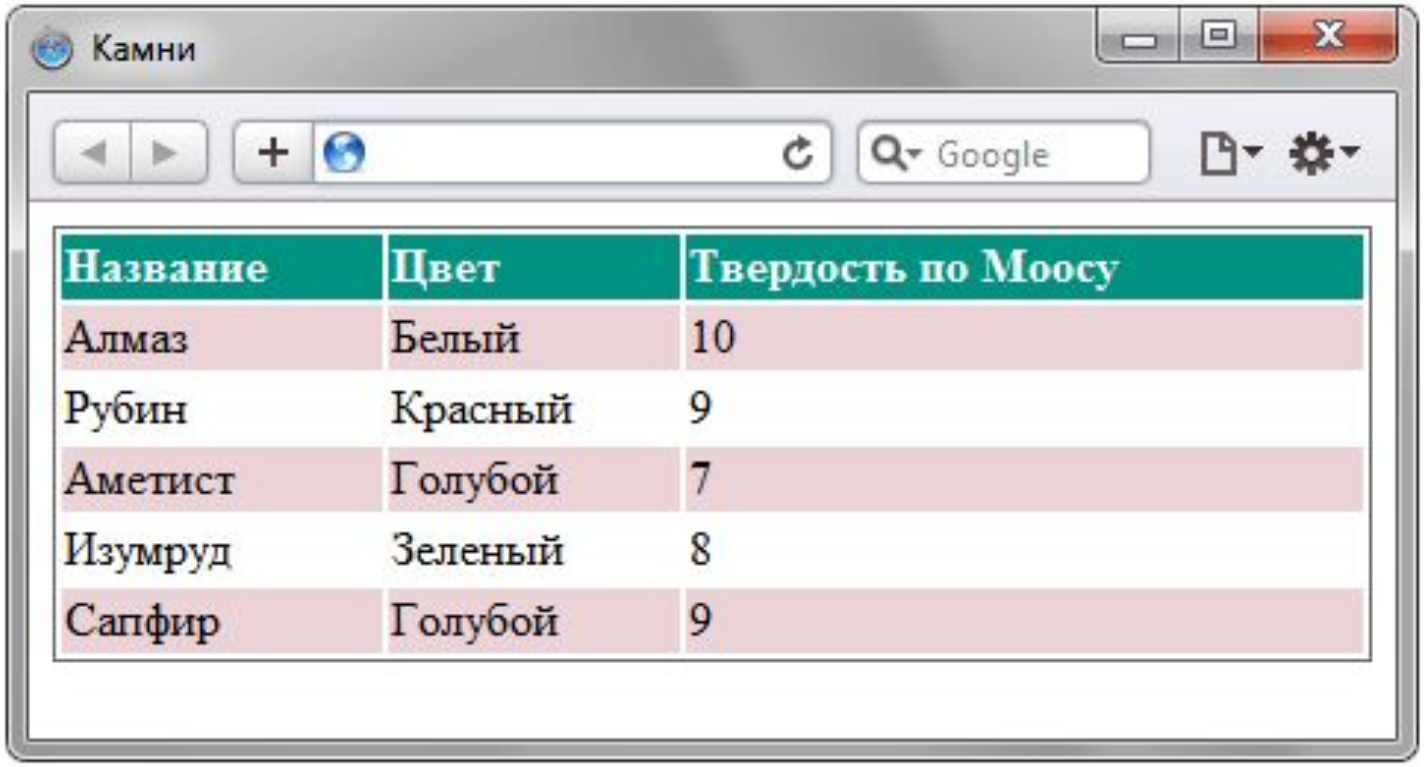


Рис. 8.1. Вид текста, оформленного с помощью стилевых классов

Классы удобно использовать, когда нужно применить стиль к разным элементам веб-страницы: ячейкам таблицы, ссылкам, абзацам и др. В примере 8.3 показано изменение цвета фона строк таблицы для создания «зебры».



Название	Цвет	Твердость по Моосу
Алмаз	Белый	10
Рубин	Красный	9
Аметист	Голубой	7
Изумруд	Зеленый	8
Сапфир	Голубой	9

Идентификаторы

Идентификатор (называемый также «ID селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты.

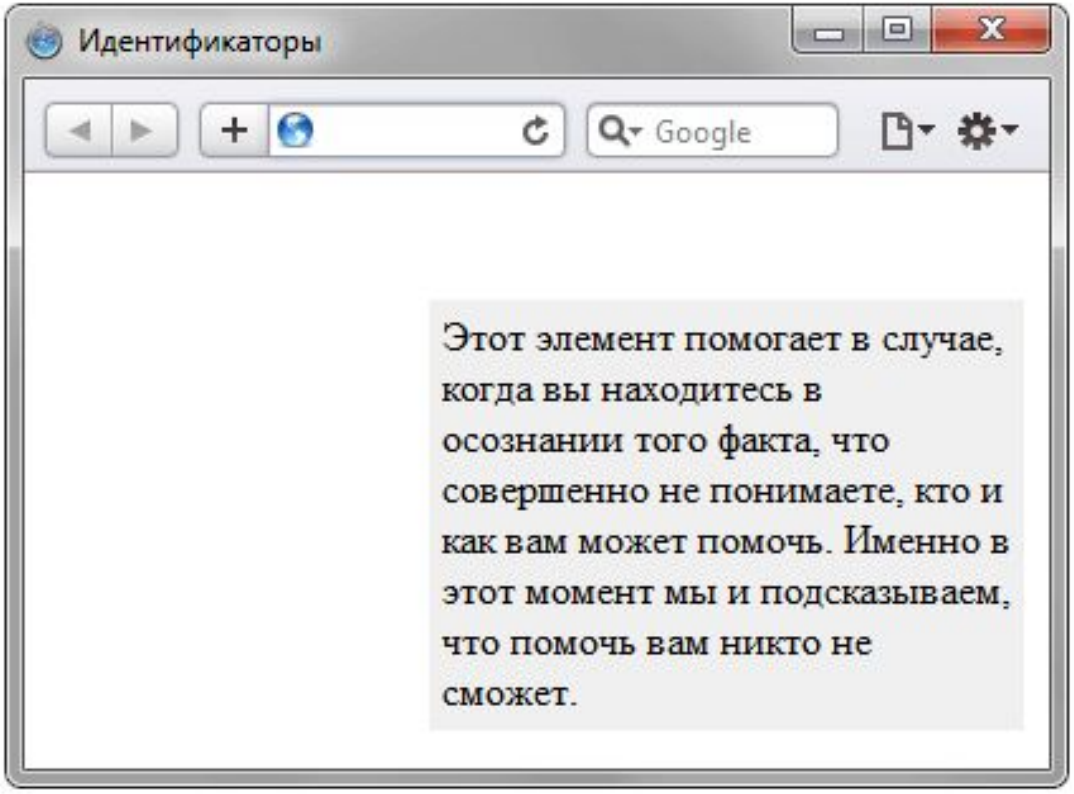
Синтаксис применения идентификатора следующий.

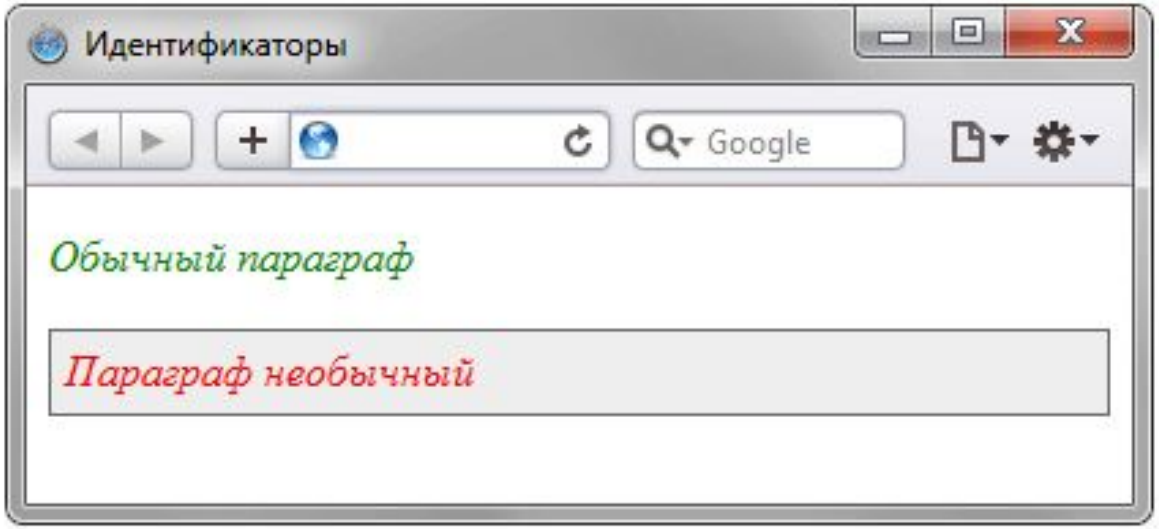
```
#Имя идентификатора { свойство1: значение; свойство2: значение; ... }
```

При описании идентификатора вначале указывается символ решётки (#), затем идет имя идентификатора. Оно должно начинаться с латинского символа и может содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах идентификатора недопустимо. В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз.

Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у тега используется атрибут id, значением которого выступает имя идентификатора (пример 9.1). Символ решётки при этом уже не указывается.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Идентификаторы</title>
  <style>
   #help {
    position: absolute; /* Абсолютное позиционирование */
    left: 160px; /* Положение элемента от левого края */
    top: 50px; /* Положение от верхнего края */
    width: 225px; /* Ширина блока */
    padding: 5px; /* Поля вокруг текста */
    background: #f0f0f0; /* Цвет фона */
   }
  </style>
 </head>
 <body>
  <div id="help">
   Этот элемент помогает в случае, когда вы находитесь в осознании того
   факта, что совершенно не понимаете, кто и как вам может помочь. Именно
   в этот момент мы и подсказываем, что помочь вам никто не сможет.
  </div>

 </body>
</html>
```


Идентификаторы
Google
Этот элемент помогает в случае, когда вы находитесь в осознании того факта, что совершенно не понимаете, кто и как вам может помочь. Именно в этот момент мы и подсказываем, что помочь вам никто не сможет.


Идентификаторы
Google
Обычный параграф
Параграф необычный

Контекстные селекторы

При создании веб-страницы часто приходится вкладывать одни теги внутрь других. Чтобы стили для этих тегов использовались корректно, помогут селекторы, которые работают только в определённом контексте. Например, задать стиль для тега <b> только когда он располагается внутри контейнера <p>. Таким образом можно одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

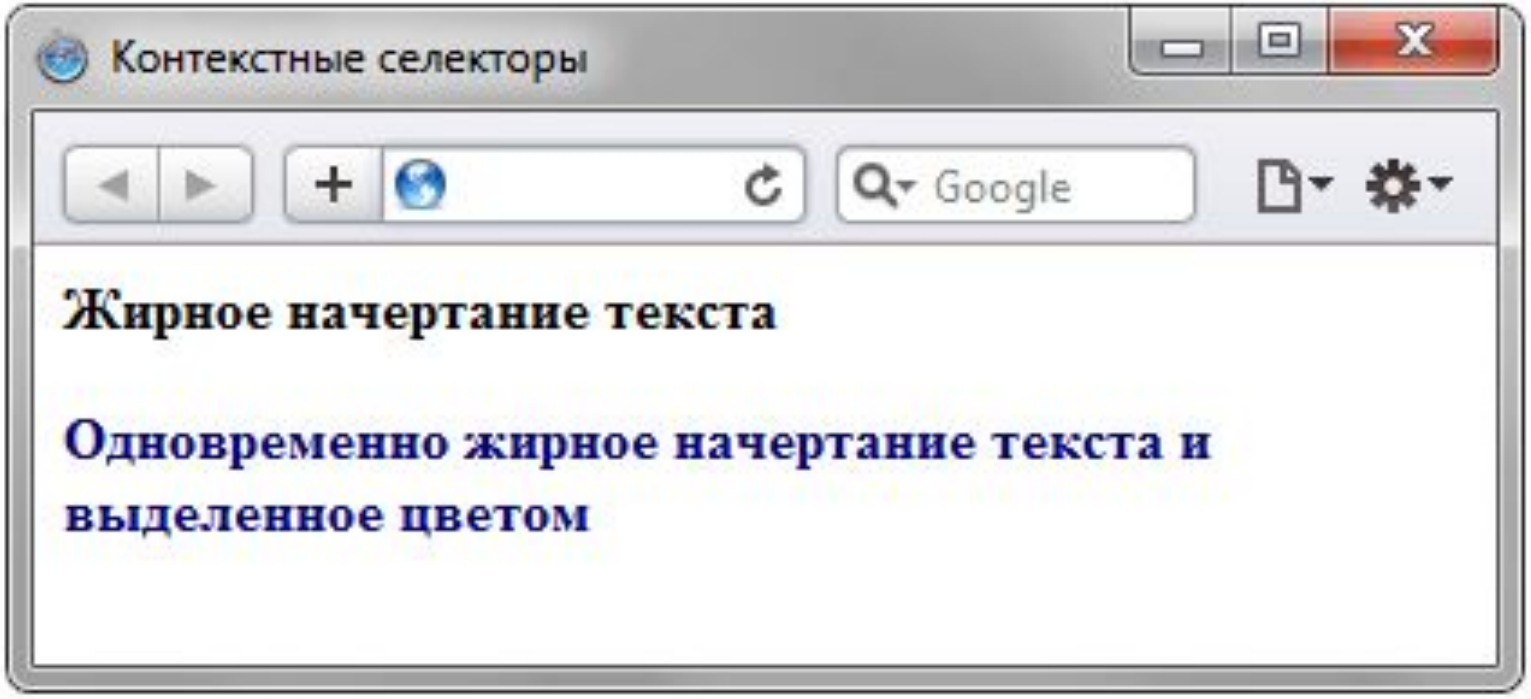
Контекстный селектор состоит из простых селекторов разделенных пробелом. Так, для селектора тега синтаксис будет следующий.

```
Тег1 Тег2 { ... }
```

В этом случае стиль будет применяться к Тегу2 когда он размещается внутри Тега1, как показано ниже.

```
<Тег1>
 <Тег2> ... </Тег2>
</Тег1>
```

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Контекстные селекторы</title>
  <style>
   P B {
    font-family: Times, serif; /* Семейство шрифта */
    color: navy; /* Синий цвет текста */
   }
  </style>
 </head>
 <body>
  <div><b>Жирное начертание текста</b></div>
  <p><b>Одновременно жирное начертание текста
  и выделенное цветом</b></p>
 </body>
</html>
```

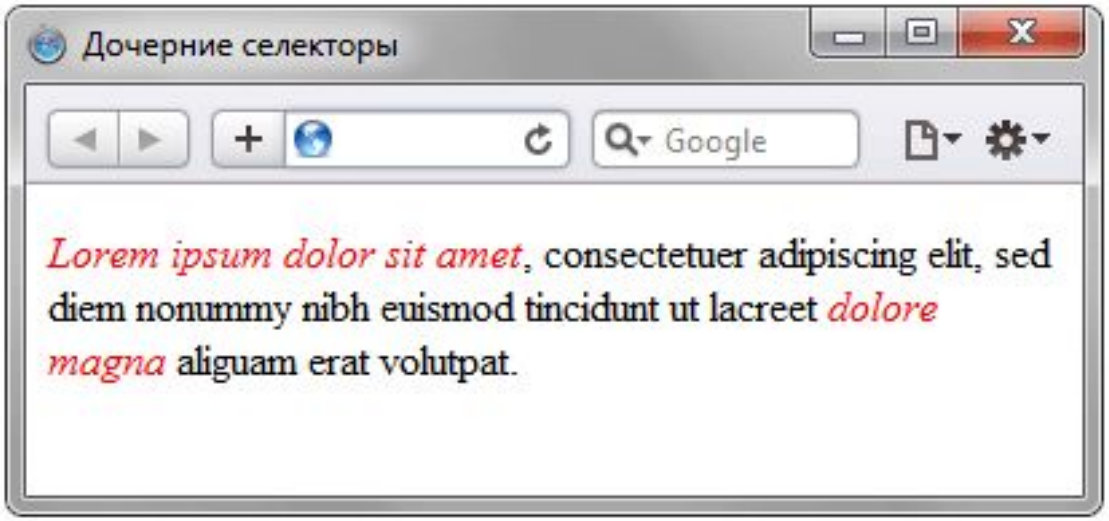

Контекстные селекторы
Google
Жирное начертание текста
Одновременно жирное начертание текста и выделенное цветом

Соседние селекторы

Соседними называются элементы веб-страницы, когда они следуют непосредственно друг за другом в коде документа.

Дочерние селекторы

Дочерним называется элемент, который непосредственно располагается внутри родительского элемента.



Селекторы атрибутов

Многие теги различаются по своему действию в зависимости от того, какие в них используются атрибуты. Например, тег <input> может создавать кнопку, текстовое поле и другие элементы формы всего лишь за счёт изменения значения атрибута type. При этом добавление правил стиля к селектору INPUT применит стиль одновременно ко всем созданным с помощью этого тега элементам. Чтобы гибко управлять стилем подобных элементов, в CSS введены селекторы атрибутов. Они позволяют установить стиль по присутствию определённого атрибута тега или его значения.

Универсальный селектор

Иногда требуется установить одновременно один стиль для всех элементов веб-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу веб-страницы.

Для обозначения универсального селектора применяется символ звёздочки (*) и в общем случае синтаксис будет следующий.

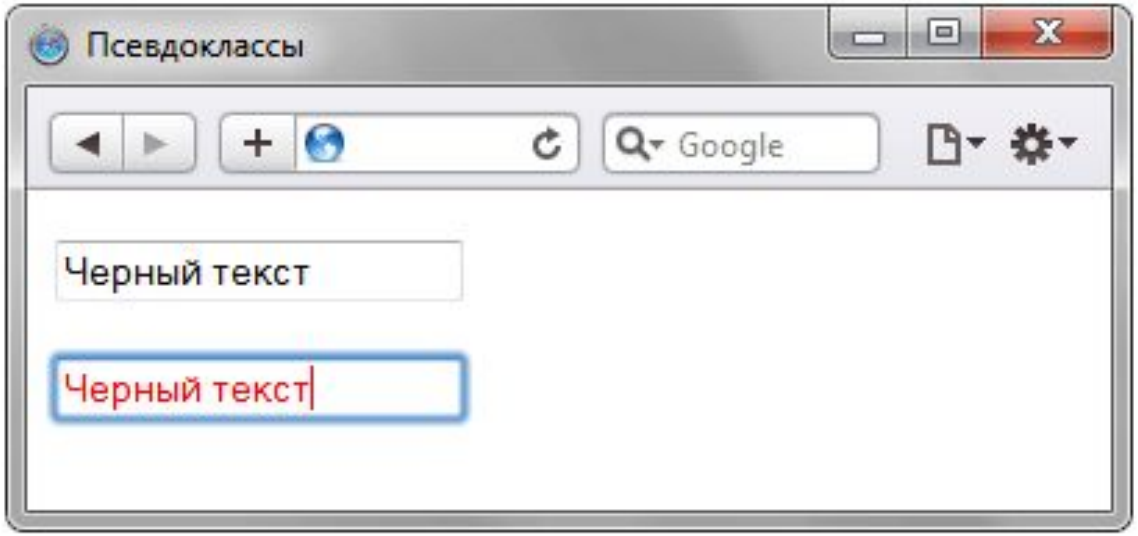
```
* { Описание правил стиля }
```

Псевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице.

Синтаксис применения псевдоклассов следующий.

```
Селектор:Псевдокласс { Описание правил стиля }
```

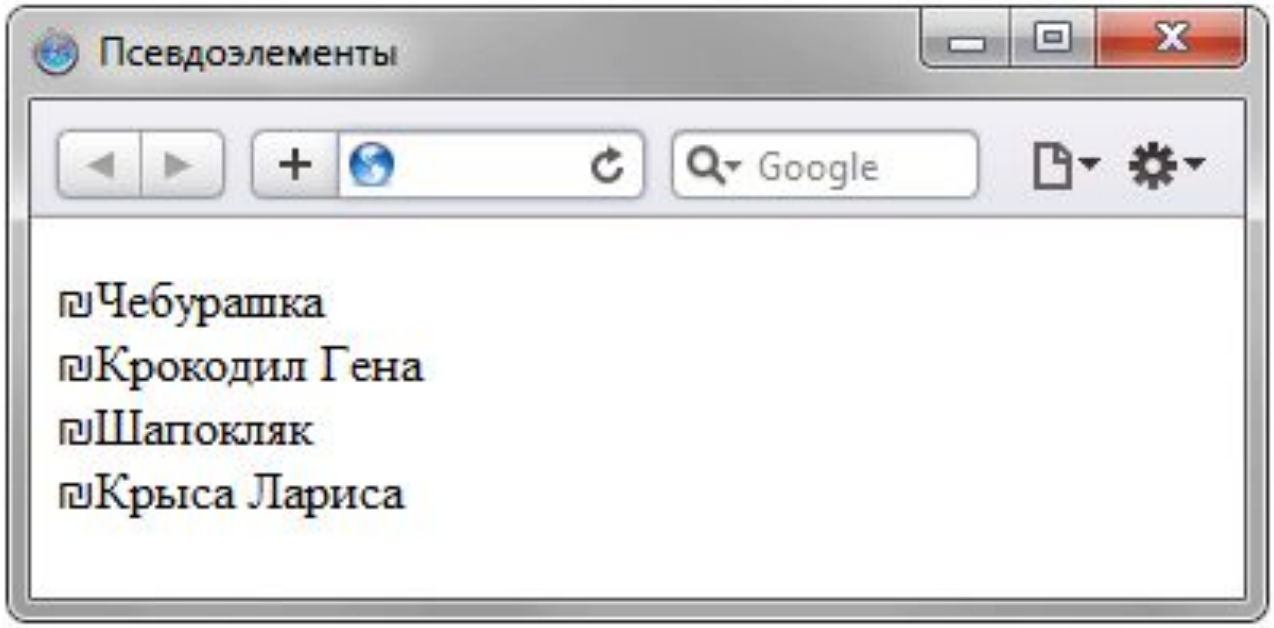


Псевдоэлементы

Псевдоэлементы позволяют задать стиль элементов не определённых в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

Синтаксис использования псевдоэлементов следующий.

```
Селектор:Псевдоэлемент { Описание правил стиля }
```



Написание эффективного кода

В процессе написания CSS следует придерживаться некоторых принципов, которые позволяют сократить код CSS, сделать его более удобным, наглядным и читабельным. Читабельность в данном случае означает, что разработчик спустя какое-то время может легко понять и модифицировать стиль или что в коде разберётся даже сторонний человек.

Размещайте каскадные таблицы стилей в отдельном файле

Размещение стилей в отдельном файле позволяет ускорить загрузку веб-страниц за счёт уменьшения их кода, а также кэширования файла с описанием стиля.

Удаляйте неиспользуемые селекторы

Большое количество селекторов создаёт путаницу в вопросе о том, кто из них за что отвечает, да и просто увеличивает объем документа. Чтобы этого не произошло, удаляйте селекторы, которые никак не применяются на сайте. К сожалению, определить точно, какой селектор используется, а какой нет, довольно сложно, поэтому добавляйте комментарий в код. Это поможет хотя бы не запутаться в большом объёме текста.

Применяйте группирование

Достоинство и удобство группирования состоит в описании одинаковых свойств в одном месте. Тем самым, значение свойства пишется только один раз, а не повторяется многократно.

Используйте универсальные свойства

Вместо того чтобы указывать значения отступа на каждой стороне элемента через свойства margin-left, margin-right, margin-top и margin-bottom, это можно одновременно задать через универсальное свойство margin. Перечисление значений через пробел позволяет установить индивидуальные отступы для каждой стороны. Кроме margin к универсальным свойствам относятся background, border, font, padding. Применение этих свойств сокращает объём кода и повышает его читабельность.

Форматирование кода

Существует множество разных подходов как же писать CSS-код. Кто-то упорядочивает селекторы по блокам, другой согласно структуре документа, третий по алфавиту, в общем, сколько людей, столько и мнений. Вы можете воспользоваться онлайновым инструментом, который форматирует CSS-код сразу четырьмя разными способами. А там уже сами решите, какой из способов вам симпатичнее.

Ссылка на сайт

http://www.cssportal.com/format-css/

http://www.wisdomweb.ru/CSS/css-first.php

РАЗДЕЛ 3. ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA SCRIPT

ЛЕКЦИЯ 6. Основы JAVA SCRIPT

ЮТИ ТПУ

Кафедра информационных систем

Направление 09.03.03 Прикладная информатика

Возможности JavaScript

С помощью **JavaScript** можно создавать интерактивные веб-страницы. Интерактивные страницы могут взаимодействовать с пользователем (выводить сообщения, изменять содержимое после определенных действий и т.д.). JavaScript встраивается прямо в веб-страницы и исполняется браузером во время их загрузки.

JavaScript был создан в 1995 году как инструмент предоставляющий веб-дизайнерам возможности программирования. JavaScript обладает простым синтаксисом и его очень легко изучить.

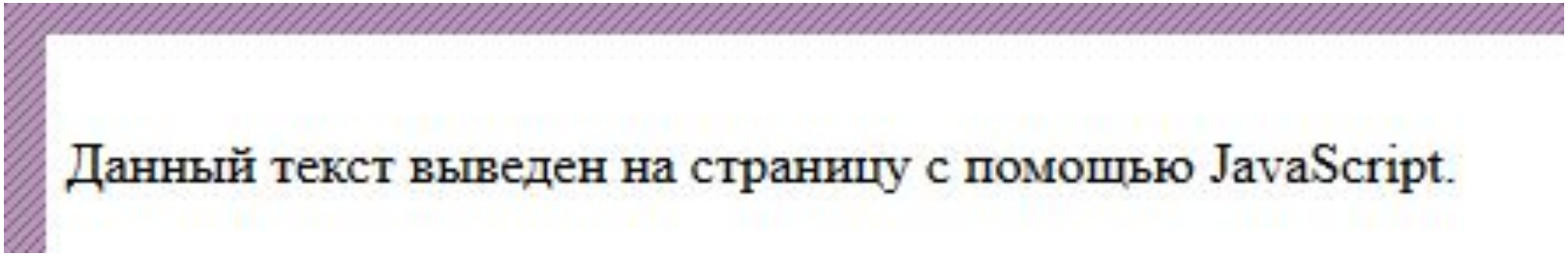
Все современные браузеры имеют поддержку JavaScript.

С помощью JavaScript можно:

- Динамически изменять содержимое веб-страниц;
- Привязывать к элементам обработчики событий (функции которые выполнят свой код только после того, как совершатся определенные действия);
- Выполнять код через заданные промежутки времени;
- Управлять поведением браузера (открывать новые окна, загружать указанные документы и т.д.);
- Создавать и считывать cookies;
- Определять, какой браузер использует пользователь (также можно определить ОС, разрешение экрана, предыдущие страницы, которые посещал пользователь и т.д.);
- Проверять данные форм перед отправкой их на сервер и многое другое.

JavaScript добавляется на веб-страницы с помощью тэга **<script>**.

```
<script type="text/javascript">
document.write("Данный текст выведен на страницу с помощью JavaScript.");
</script>
```



Отладчики веб-страниц

Отладчики веб-страниц упрощают разработку веб-сайтов и написание JavaScript кода. Отладчики доступны для всех современных браузеров.

В Chrome, Safari, Opera и Internet Explorer 8+ имеются встроенные отладчики, которые можно вызвать с помощью нажатия клавиши F12 (Chrome, Safari и IE) или Ctrl+Shift+I (Opera) при просмотре страниц.

В последней версии Firefox был добавлен встроенный отладчик, который не очень неудобный, так что можно скачать Firebug (данный отладчик доступен в виде дополнения).

JS включения

Если Вы хотите, чтобы JavaScript код не смешивался с HTML разметкой размещайте его в секции head.

Пример

```
<html>
<head>
<script type='text/javascript'>
function example(){
   alert('Если Вы видите это сообщение, значит страница была полностью загружена.');
}
</script>
</head>
<body onload='example()'>
</body>
</html>
```

JavaScript может быть размещен в секции body.

Размещайте JavaScript в самом конце секции body (перед </body>) если не хотите, чтобы скрипт начал выполняться до полной загрузки документа и это привело к ошибкам.

Обратите внимание: на странице может быть размещено неограниченное количество скриптов в том числе и в body, и в head одновременно.

Пример

```
<html>
<body>
<p>Данный текст присутствовал на странице.</p>
<hr />
<script type='text/javascript'>
document.write('Данный текст выведен на страницу с помощью JavaScript.');
</script>
</body>
</html>
```

Подключение внешних скриптов

JavaScript код необязательно должен непосредственно содержаться в HTML документе, он также может храниться во внешнем текстовом файле с расширением **.js**.

Использовать внешние файлы скриптов удобно в случаях, когда необходимо определять код, который будет работать на нескольких страницах веб-сайта.

Внешние скрипты также как и обычные подключаются к страницам с помощью тэга <script> однако в этом случае содержимое тэга должно оставаться пустым и к нему должен быть добавлен атрибут src содержащий адрес внешнего .js файла.

Пример

```
<!-- Содержимое файла ex.js: 'document.write("Данный текст был выведен на страницу с
<script type="text/javascript" src="ex.js"></script>
```

JavaScript команды

С помощью **JavaScript команд** Вы можете "общаться" с браузером. Например с помощью команды **document.write('Привет!');** Вы можете сообщить браузеру, что хотите вывести на страницу текст *"Привет!"*.

Точка с запятой сообщает браузеру о конце команды.

Обратите внимание: в JavaScript необязательно явно указывать точку запятой после каждой команды, так как браузер автоматически подставит ее если следующая команда будет написана на новой строке.

В отличие от HTML JavaScript чувствителен к регистру букв. Всегда обращайте внимание на регистр букв при наборе команд в JavaScript.

Пример

```
document.wrITE("<hr /><b>Поздравляем, Вы исправили ошибку! </b><hr />");
```

JavaScript код

JavaScript код является последовательностью JavaScript команд. Команды исполняются браузером сверху вниз. Следующий пример выводит два абзаца, написанных жирным и курсивным текстом, на страницу:

Пример

```
document.write("<p><b>Это абзац написанный жирным шрифтом.</b></p>");
document.write("<p><i>Это абзац написанный курсивным шрифтом.</i></p>");
```

JavaScript блоки команд

JavaScript команды могут объединяться в блоки. Обычно блоки используют при определении функций, создании циклов и в некоторых других случаях. Начало блока команд обозначается открывающейся фигурной скобки ({), а конец закрывающейся (}).

Пример использования блоков:

Пример

```
function mesg()
{
document.write("<p><b>Это абзац написанный жирным шрифтом.</b></p>");
document.write("<p><i>Это абзац написанный курсивным шрифтом.</i></p>");
}
```

JavaScript комментарии

С помощью комментариев Вы можете оставлять в коде различные "заметки", которые помогут разобраться в нем другим лицам.

В JavaScript существует два вида комментариев: **короткие** и **длинные**.

Короткие комментарии - это комментарии, длина которых не превышает длину строки. Все что находится после символа **//** до конца строки будет являться коротким комментарием.

Пример

```
//Команда ниже выведет текст написанный жирным шрифтом
document.write("<p><b>Это абзац написанный жирным шрифтом.</b></p>");
//Команда ниже выведет текст написанный курсивом
document.write("<p><i>Это абзац написанный курсивным шрифтом.</i></p>");
```

При отладке кода комментарии могут использоваться для предотвращения выполнения некоторых команд, например:

Пример

```
//Команда ниже выполнена не будет
//document.write("Привет хозяин");
```

Многострочные комментарии

Многострочные комментарии - это комментарии, длина которых может превышать длину строки. Многострочные комментарии начинаются с /* и заканчиваются */.

Пример

```
/*
Первая команда выводит абзац
жирного, а вторая абзац
курсивного текста
*/
document.write("<p><b>Это абзац написанный жирным шрифтом.</b></p>");
document.write("<p><i>Это абзац написанный курсивным шрифтом.</i></p>");
```

Переменные JavaScript

JavaScript переменные являются "контейнерами", в которые Вы можете загружать различную информацию, а позднее извлекать ее обратно.

Каждая JavaScript переменная должна иметь собственное уникальное имя, которое может начинаться с латинской буквы или символа "_".

Обратите внимание: имя переменных в JavaScript не может начинаться с цифр.

Обратите внимание: так как JavaScript чувствителен к регистру, переменные с одинаковыми именами написанными в разном регистре (например var и VAR), будут являться разными переменными.

Создание переменных

Создание переменных в JavaScript часто называют ***"объявлением"*** переменных. Переменные в JavaScript объявляются с помощью команды **var**.

```
//Создаем переменную с именем ex1
var ex1;
//Создаем переменную с именем ex2
var ex2;
```

Загрузка значений в переменные в JavaScript выполняется с помощью **оператора =**.

Значения могут загружаться в контейнеры также прямо в момент создания как в примере ниже:

```
//Создаем переменную с именем ex1 содержащую значение 4
var ex1=4;
//Создаем переменную с именем ex2 содержащую значение 5
var ex2=5;
```

Для того, чтобы извлечь значение из созданной ранее переменной, необходимо обратится к ее имени.

В этом примере мы будем извлекать содержимое переменных и сразу выводить его на страницу с помощью команды document.write.

Пример

```
//Запишем число 4 в переменную ex1
var ex1=4;
//Запишем число 5 в переменную ex2
var ex2=5;
//Выведем содержимое переменной ex1 на страницу
document.write(ex1+'<br />');
//Выведем содержимое переменной ex2
document.write(ex2+'<br />');
//Изменим содержимое переменной ex2
ex2=200;
//Выведем новое содержимое переменной ex2
document.write(ex2);
```

Строковые переменные

Помимо чисел Вы можете хранить в переменных произвольный текст. Переменные, которые хранят текст, называются **строковыми переменными**.

При записи текста в переменную обязательно заключайте его в двойные (") или одинарные кавычки (').

Пример

```
//Запишем в переменную ex строку 'Привет всем!'
var ex='Привет всем!';
//Выведем значение переменной ex на страницу
document.write(ex);
```

Определение переменных с var и без него

В JavaScript Вы можете определять переменные с var и без него.

```
//Создадим новую переменную с var
var ex=123;
//Создадим новую переменную без var
ex2=20;
```

Вам может показаться, что объявление переменных с var и без него всегда приводят к одинаковому результату, но это действительно так только, когда когда объявление происходит в глобальном контексте (т.е. за пределами всех функций).

Если же объявление происходит в локальном контексте (т.е. в теле какой-либо функции) объявление с var создает локальную переменную (т.е. переменную, которая будет доступна только в теле данной функции и после выполнения функции будет уничтожена), объявление без var создает глобальную переменную (т.е. переменную, которая будет доступна другим функциям внутри данного скрипта).

Чтобы избежать ошибок в коде рекомендуем стараться всегда определять переменные с var.

Об удалении и переопределении переменных

Переопределяя переменные Вы не стираете значение, которое хранятся в них.

```
var ex=123;
var ex;
document.write(ex); // Выведет 123
```

Если Вы хотите удалить переменную в JavaScript и она не была объявлена с помощью var Вы можете использовать оператор **delete**.

```
ex=123;
delete ex;
```

Оператор delete не может удалить переменные объявленные с помощью var, поэтому если переменная была объявлена с помощью var, то единственный способ удалить ее - присвоить ей значение null или undefined.

```
var ex=123;

ex=null;
// или
ex=undefined
```

Арифметические операторы JavaScript

Арифметические операторы используются для выполнения арифметических операций над переменными или значениями.

В таблице ниже перечислены арифметические операторы доступные в JavaScript (предположим, что **x=7**).

Оператор	Описание	Пример	Результат
+	Выполняет сложение чисел	y=x+10	y=17
-	Выполняет вычитание чисел	y=x-3	y=4
*	Выполняет умножение чисел	y=x*4	y=28
/	Выполняет деление чисел	y=x/2	y=3
%	Вычисляет остаток от деления чисел	y=x%2	y=1
++	Увеличивает значение на 1 и возвращает новое значение Увеличивает значение на 1 и возвращает старое значение	y=++x y=x++	y=8, x=8 y=7, x=8
--	Уменьшает значение на 1 и возвращает новое значение Уменьшает значение на 1 и возвращает старое значение	y=--x y=x--	y=6, x=6 y=7, x=6

Пример

```
//Объявим переменные
var x=6;
var y=7;
//Выполним сложение переменных и выведем результат на страницу
z=x+y;
document.write(z);
//Выполним умножение и выведем результат на страницу
g=x*y;
document.write(g);
//Выполним вычитание и выведем результат на страницу
t=y-x;
document.write(t);
```

```
13
42
1
```

Сокращенная запись арифметических операторов

Для того, чтобы уменьшить размер кода Вы можете использовать сокращенную запись арифметических операций.
Условимся что **x=4**, а **y=6**:

Оператор	Сокращенная запись	Полная запись	Результат
+=	y+=x	y=y+x	y=10
-=	y-=x	y=y-x	y=2
*=	y*=x	y=y*x	y=24
/=	y/=x	y=y/x	y=1
%=	y%=x	y=y%x	y=2

Использование оператора + со строковыми переменными

Если оператор + используется со строковыми переменными он выполняет объединение строк хранящихся в них.

Пример

```
//Запишем "Привет " в переменную ex1
ex1="Привет ";
//Запишем "всем" в переменную ex2
ex2="всем";
/* Соединим значения переменных ex1 и ex2, добавим к ним восклицательный
знак и затем запишем результат в ex3 */
ex3=ex1+ex2+"!";
//Выведем содержимое переменной ex3
document.write(ex3);
</script>
```

Привет всем!

Результатом сложения строки и числа всегда будет строка

```
//Сложение двух чисел
var ex1=10+5;
document.write(ex1+"<br />");
//Сложение двух строк
var ex2="10"+"5"
document.write(ex2+"<br />");
//Сложение строки и числа
var ex3="10"+5;
document.write(ex3+"<br />");
var ex4="5"+10;
document.write(ex4);
```

```
15
105
105
510
```

Операторы сравнения

Операторы сравнения позволяют производить над переменными и значениями различные операции сравнения.

В результате выполнения таких операций в зависимости от исхода возвращается true (истина) или false (ложь).

Пример

```
//Произведем сравнение чисел 7 и 10
document.write(7==10) // Выведет false так как числа не равны
//Теперь произведем сравнение чисел 10 и 10
document.write(10==10) // Выведет true так как числа равны
```

```
false
true
```

В таблице ниже перечислены доступные в JavaScript операторы сравнения (предположим, что **x=7**):

Оператор	Описание	Пример	Результат
==	Проверяет переменные или значения на равенство.	x==7	**true**
===	Проверяет переменные или значения на равенство учитывая тип переменной	x===7 x==="7"	**true** **false**
!=	Проверяет различаются ли переменные или значения	x!=9	**true**
>	Проверяет больше ли переменная или значение стоящее слева, чем стоящее справа	x>13	**false**
<	Проверяет меньше ли переменная или значение стоящее слева, чем стоящее справа	x<13	**true**
>=	Проверяет является ли переменная или значение стоящее слева большим или равным стоящему справа	x>=13 x>=7	**false** **true**
<=	Проверяет является ли переменная или значение стоящее слева меньшим или равным стоящему справа	x<=13 x<=7	**true** **true**

Операторы сравнения в основном используются в **условных конструкциях**.

```
//Запишем число 10 в переменную a
var a=10;
//Запишем число 7 в переменную b
var b=7;
//Проверим совпадают ли числа
if (a==b) {
   //Если числа совпадают выведем "Числа совпадают"
   document.write("Числа совпадают");
}
else {
   //Если числа не совпадают выведем "Числа не совпадают"
   document.write("Числа не совпадают");
}
```

Числа не совпадают

Логические операторы

Логические операторы используются для связки нескольких операторов сравнения.

В таблице ниже приведены логические операторы доступные в JavaScript (предположим, что **x=2**, а **y=9**):

Оператор	Значение	Пример	Результат
&&	*И*	(x==2 && y==9) (x==3 && y==9)	**true** **false**
||	*ИЛИ*	(x==2 || y==8) (x==3 || y==9) (x==5 || y==6)	**true** **true** **false**
!	*НЕ*	!(x==3)	**true**

Условные конструкции JavaScript

С помощью условных конструкций Вы можете изменить стандартную очередность выполнения команд (по умолчанию команды исполняются поочередно сверху вниз).

В JavaScript имеются следующие условные конструкции:

конструкция if используйте данную конструкцию если хотите, чтобы блок команд был выполнен только если указанное условие истинно;

конструкция if..else используйте данную конструкцию если хотите, чтобы один блок команд был выполнен если указанное условие истинно и другой блок команд если условие ложно;

конструкция if..else if..else используйте данную конструкцию если хотите, чтобы при определенном условии команды выполнились только в одном из нескольких блоков;

конструкция switch используйте данную конструкцию если хотите, чтобы при определенном условии команды выполнились только в одном из нескольких блоков;

Конструкция if

Синтаксис:

```
if (условие) {
   //Команды расположенные здесь будут выполнены только если условие истинно
}
```

```
//Запишем число 7 в переменную a
var a=7;
//Запишем число 7 в переменную b
var b=7;
//Проверим совпадают ли числа
if (a==b) {
   //Если числа совпадают выведем "Числа совпадают"
   document.write("Числа совпадают");
}
```

Если блок команд состоит из одной команды, то фигурные скобки могут быть опущены. Код ниже также будет исполнен корректно:

```
//Запишем число 7 в переменную a
var a=7;
//Запишем число 7 в переменную b
var b=7;
//Проверим совпадают ли числа и выведем "Числа совпадают" если это так
if (a==b) document.write("Числа совпадают");
```

Конструкция if..else

Синтаксис:

```
if (условие) {
   //Команды расположенные здесь будут выполнены только если условие истинно
}
else {
   //Команды расположенные здесь будут выполнены только если условие ложно
}
```

Пример

```
//Запишем число 10 в переменную a
var a=10;
//Запишем число 7 в переменную b
var b=7;
//Проверим совпадают ли числа
if (a==b) {
   //Если числа совпадают выведем 'Числа совпадают'
   document.write('Числа совпадают');
}
else {
   //Если числа не совпадают выведем 'Числа не совпадают'
   document.write('Числа не совпадают');
}
```

Обратите внимание: всегда пишите условные слова if и else строчными буквами, использование IF и ELSE приведет к ошибке JavaScript.

Конструкция if..else if..else

Синтаксис:

```
if (условие1) {
   //Команды расположенные здесь будут выполнены только если условие1 истинно
}
else if (условие2) {
   //Команды расположенные здесь будут выполнены только если условие2 истинно
}
....
else if (условиеN) {
   //Команды расположенные здесь будут выполнены только если условиеN истинно
}
else {
   //Команды расположенные здесь будут выполнены если ни одно из условий выше не было
}
```

Пример

```
var a=20;
var b=44;
if (a > b) {
   document.write('<b>a больше b.</b>');
}
else if (a==b){
   document.write('<b>a равно b.</b>');
}
else {
   document.write('<b>a меньше b.</b>');
}
```

Альтернативный синтаксис конструкции if..else

Данная конструкция выполняет действия аналогичные оригинальной конструкции if..else, но позволяет значительно сократить размер кода.

```
(условие)?команды1:команды2
```

Если **условие** истинно, будут выполнены **команды1**, если ложно будут выполнены **команды2**.

Пример

```
var a=20;
var b=44;
//Если переменная a больше переменной b, то переменной c будет присвоено 10, а если не
var c=(a>b)?10:20;
//Выведем значение переменной c на страницу
document.write(c);
```

```
20
```

Конструкция Switch

Используйте данную конструкцию если хотите, чтобы при определенном условии команды выполнились только в одном из нескольких блоков.

Обратите внимание: ключевое слово break используется для предотвращения автоматического исполнения кода следующего case.

Обратите внимание: ключевые слова switch, case, break, default всегда должны быть написаны строчными буквами.

Синтаксис:

```
switch (x) {
   case n:
      Команды этого блока будут выполнены если x=n
   break;
   case t:
      Команды этого блока будут выполнены если x=t
   break;
   case j:
      Команды этого блока будут выполнены если x=j
   break;

   default:
      Команды этого блока будет выполнены если x не равен n, t и j
}

/* Если Вы хотите, чтобы при нескольких разных условиях был выполнен один
и тот же блок команд просто объедините блоки: */

switch (x){
case n: case t: case j:
   Команды этого блока будут выполнены если x=n или x=t или x=j
}
```

Пример

```
//Определяем сегодняшний день (Понедельник=1, Среда=3, Воскресенье=7 и т.д.)
dat=new Date();
day=dat.getDay();
//И в зависимости от этого выводим сообщения
switch(day) {
   case 6:
      document.write('<h2>'+'Сегодня выходной!'+'</h2>');
   break;
   case 0:
      document.write('<h2>'+'Сегодня выходной!'+'</h2>');
   break;
   default:
      document.write('<b>'+'Сегодня Вам нужно идти на работу...'+'</b>');
}
```

Обратите внимание: операции с датой и временем, которые были использованы в данном примере будут подробно рассмотрены далее в этом учебнике.

Окна оповещения в JavaScript

Окна оповещения используются в случаях, когда необходимо, чтобы пользователь обязательно обратил внимание на определенную информацию.

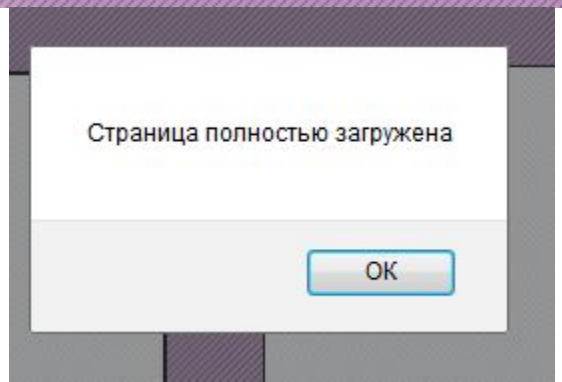
Когда окно оповещения будет вызвано пользователь должен будет нажать кнопку "OK" для, того чтобы продолжить просмотр страницы.

Синтаксис:

```
alert("Текст окна оповещения");
```

Пример

```
//Данная функция будет вызвана после полной загрузки страницы
function popBox(){
   //Выведем окно оповещения
   alert("Страница полностью загружена");
}
```



Окна подтверждения в JavaScript

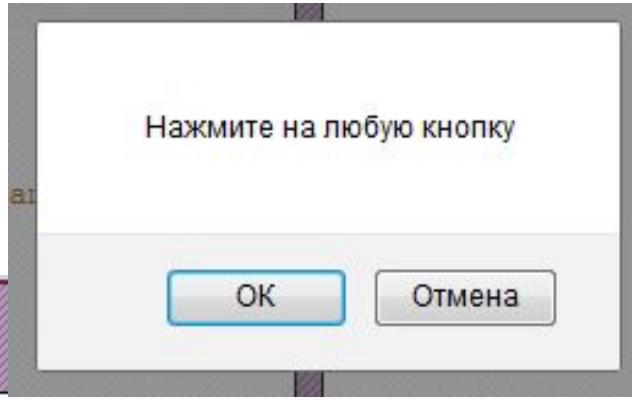
Окна подтверждения используются в случаях когда необходимо, чтобы пользователь подтвердил или отклонил что-либо.

Когда окно подтверждения будет вызвано пользователь должен будет нажать либо "OK", либо "Отмена", чтобы продолжить.

Если пользователь нажмет "OK" вернется true (истина), если пользователь нажмет "Отмена" вернется false (ложь).

Синтаксис:

```
var x=confirm("Текст окна подтверждения");
```



Пример

```
//Данная функция будет вызвана после полной загрузки страницы
function popBox(){
   /* После вызова окна подтверждения в переменную x будет возвращено true или false
   в зависимости от того, какую кнопку нажал пользователь */
   x=confirm("Нажмите на любую кнопку");
   if (x==true){
   //Если пользователь нажал OK вывести соответствующее сообщение
   document.write('Вы нажали OK');
   }
   else {
   //Если пользователь нажал Отмена вывести соответствующее сообщение
   document.write('Вы нажали Отмена.');
   }
}
```

Окна запроса в JavaScript

Окна запроса используются в случаях, когда от пользователя необходимо получить определенную информацию.

Когда окно запроса будет вызвано пользователь должен будет ввести определенные данные и нажать на "OK". Если пользователь не хочет вводить данные он может нажать "Отмена" и окно сразу будет закрыто.

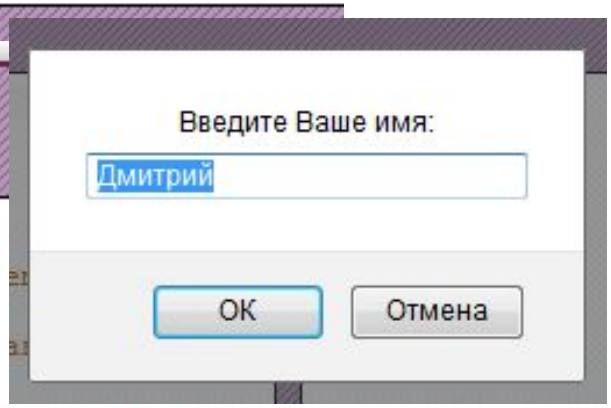
Если пользователь введет что-либо в окно и нажмет "OK" будет возвращено введенное пользователем значение, если пользователь нажмет "Отмена", то будет возвращено null.

Синтаксис:

```
var x=prompt("Текст окна запроса", "Заполнитель поля ввода");
```

Пример

```
//функция будет вызвана после загрузки страницы
function popBox() {
   //Выведет окно запроса
   ex1=prompt("Введите Ваше имя:", "Дмитрий");
   //Отобразит имя введенное пользователем в окно запроса
   document.write('Ваше имя: '+ex1);
}
```



JavaScript функции

Функции являются одним из наиболее важных строительных блоков кода в JavaScript. Функции состоят из набора команд и обычно выполняют какую-то одну определенную задачу (например суммирование чисел, вычисление корня и т.д.).

Код помещенный в функцию будет выполнен только после явного вызова этой функции.

Объявление функций

1. **Синтаксис:**

```
//Объявление функции
function имяфункции(пер1, пер2){
   Код функции
}

//Вызов функции
имяфункции(пер1,пер2);
```

2. **Синтаксис:**

```
//Объявление функции
var имяфункции=function(пер1, пер2){Код функции}

//Вызов функции
имяфункции(пер1,пер2);
```

имяфункции задает имя функции. Каждая функция на странице должна иметь уникальное имя. Имя функции должно быть задано латинскими буквами и не должно начинаться с цифр.

пер1 и **пер2** являются переменными или значениями, которые можно передавать внутрь функции. В каждую функцию может быть передано неограниченное количество переменных.

Обратите внимание: даже если в функцию не передаются переменные не забывайте вставлять круглые скобки "()" после имени функции.

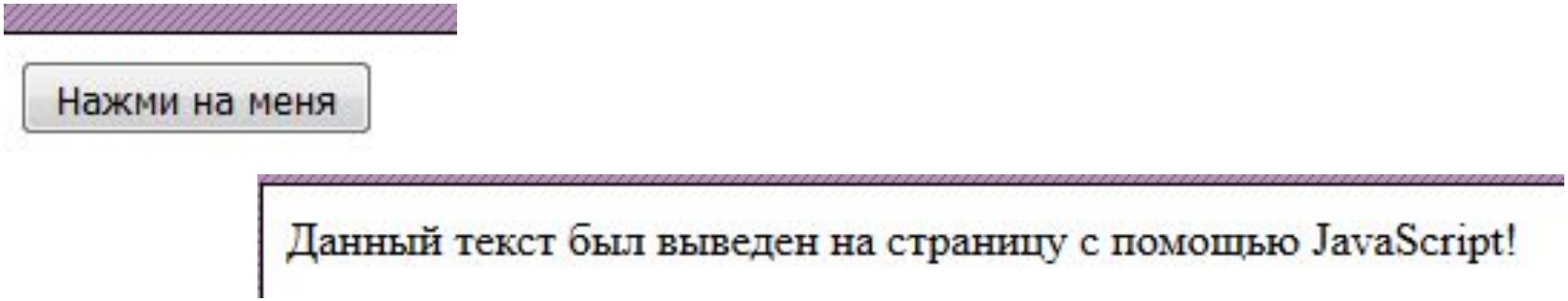
Обратите внимание: имена функций в JavaScript чувствительны к регистру.

Пример JavaScript функции

Функция messageWrite() в примере ниже будет выполнена только после нажатия на кнопку.

Пример

```
<html>
<head>
<script type='text/javascript'>
// Функция выводит текст на страницу
function messageWrite() {
   document.write('Данный текст был выведен на страницу с помощью JavaScript!');
}
</script>
</head>
<body>
<input type='button' value='Нажми на меня' onclick='messageWrite()' />
</body>
</html>
```



Передача функциям переменных

Вы можете передавать функциям неограниченное количество переменных.

Обратите внимание: все манипуляции над переменными внутри функций на самом деле производятся не над самими переменными а над их копией, поэтому содержимое самих переменных в результате выполнения функций не изменяется.

Пример

```
/* Зададим функцию, которая прибавляет к переданной переменной 10 и выводит результат
function plus(a){
   a=a+10;
   document.write('Вывод функции: ' + a+'<br />');
}
var a=25;
document.write('Значение переменной до вызова функции: '+a+'<br />');
// Вызовем функцию передав ей в качестве аргумента переменную a
plus(a);
document.write('Значение переменной после вызова функции: '+a+'<br />');
```

Значение переменной до вызова функции: 25
Вывод функции: 35
Значение переменной после вызова функции: 25

Обратите внимание: как Вы можете видеть содержимое переменной a не было изменено после вызова, так как функция производила операции над ее локальной копией.

Команда return

С помощью команды **return** Вы можете возвращать из функций значения.

Пример

```
<html>
<head>
<script type='text/javascript'>
//функция sum возвращает сумму переданных в нее переменных
function sum(v1,v2){
   return v1+v2;
}
</script>
</head>
<body>
<script type='text/javascript'>
document.write('5+6=' + sum(5,6) + '<br />');
document.write('10+4=' + sum(10,4) + '<br />');
</script>
</body>
</html>
```

```
5+6=11
10+4=14
```

Встроенные функции

Помимо определяемых пользователем функций в JavaScript существуют еще и **встроенные функции**.

К примеру встроенная функция **isFinite** позволяет проверить является ли переданное значение допустимым числом.

Пример

```
document.write(isFinite(40)+'<br />');
document.write(isFinite(-590)+'<br />');
document.write(isFinite(90.33)+'<br />');
document.write(isFinite(NaN)+'<br />');
document.write(isFinite('Это строка')+'<br />');
```

```
true
true
true
false
false
```

Локальные и глобальные переменные

Переменные создающиеся внутри функций называются **локальными переменными**. Вы можете обращаться к таким переменным только внутри функций, в которых они были определены.

После завершения выполнения кода функции такие переменные уничтожаются. Это значит, что в разных функциях могут быть определены переменные с одинаковым именем.

Переменные, которые создаются вне кода функций называются **глобальными переменными** к таким переменным можно обращаться из любого места кода.

Если Вы объявляете переменную без var внутри функции она тоже становится глобальной.

Глобальные переменные уничтожаются только после закрытия страницы.

Использование анонимных функций

Функции, которые не содержат имени при объявлении называются **анонимными**.

Анонимные функции в основном объявляют не для последующего их вызова из кода как обычные функции, а для передачи другим функциям в качестве параметра.

Циклы JavaScript

Цикл - это блок команд, который может повторно выполняться пока определенное условие не будет выполнено.

В JavaScript поддерживает следующие виды циклов:

for
while
do..while

Цикл For

Цикл for исполняет блок команд пока заданное условие является истинным.

Синтаксис:

```
for (блок определения;условие;блок изменения) {
   //Блок команд, который будет повторно выполняться пока условие истинно
}
```

Когда цикл for начинает исполнение происходит следующее:

Выполняются выражения заданные в **блоке определения** (в данном блоке определяются служебные переменные цикла такие как счетчик цикла);

Производится оценка **условия** и если оно истинно (равно true) выполнение переходит к шагу 3. Если условие ложно (равно false) цикл завершается;

Выполняется блок команд;

Выполняются выражения заданные в **блоке изменения** (в данном блоке над счетчиком цикла производятся какие-либо изменения) и выполнение переходит к шагу 2.

Пример

```
for (i=1;i<=30;i++) {
   document.write (i+'<br />');
}
```

Предположим нам нужно написать программу выводящую на страницу числа от 1 до 30.

Если бы мы не воспользовались циклом нам бы пришлось написать 30 почти идентичных строчек кода выводящих каждое из этих чисел на страницу вручную.

С помощью цикла for можно сделать тоже самое написав только 3 строчки кода.

1
2
3
4
5
6
7

Цикл While

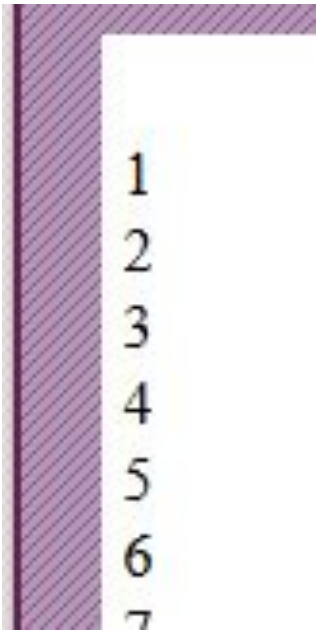
Цикл while выполняет блок кода, пока заданное условие истинно. Цикл while выполняет действия аналогичные циклу for и отличается от него только синтаксисом.

Синтаксис:

```
while (условие) {
  //Блок команд, который будет повторно выполняться если условие истинно
}
```

Пример

```
var i=1;
while (i<=30) {
   document.write (i+'<br />');
   i++;
}
```



Цикл do..while

Цикл do..while часто называют циклом с постусловием, потому что в отличие от предыдущих циклов он вначале исполняет блок команд и только потом проверяет заданное условие.

Если условие истинно блок команд выполняется еще раз, если условие ложно цикл завершает исполнение.

Синтаксис:

```
do {
   //Блок команд, который будет выполнен больше одного раза если условие истинно
}
while (условие);
```

В примере ниже цикл do..while исполнит блок кода несмотря на то, что условие ложно изначально.

Пример

```
var i=20;
do {
   document.write('Если Вы видите этот текст код в цикле был исполнен.');
}
while (i<=3);
```

Команда break

С помощью команды **break** Вы можете досрочно "обрывать" выполнение цикла.

Пример

```
//Задаем цикл, который должен выводить значения от 1 до 20
for (i=1;i<=20;i++) {
   document.write(i + '<br />');
   //Прервем выполнения цикла после того, как он досчитает до 5
   if (i==5) {
      break;
   }
}
```

1
2
3
4
5

Команда continue

Команда **continue** обрывает текущую итерацию выполнения цикла и переходит к следующей.

Пример

```
//Задаем цикл который должен выводить значения от 1
for (i=1;i<=15;i++) {
   //Пропустим 3, 10 и 13 круг выполнения цикла
   if (i==3) {
      continue;
   }
   else if (i==10) {
      continue;
   }
   else if (i==13) {
      continue;
   }
   document.write(i+'<br />');
}
```

1
2
4
5
6
7
8
9
11
12
14
15

Обратите внимание: числа 3, 10 и 13 не были выведены на страницу.

Цикл for..in

Цикл for..in используется для перебора массивов или объектов.

В ходе выполнения цикла **переменная** будет поочередно принимать значения всех элементов массива или всех свойств объекта.

Синтаксис:

```
for (переменная in объект_или_массив) {
   //Код, который будет выполнен для каждого элемента массива или свойства объекта
}
```

переменная данной переменной будет поочередно присвоено каждое свойство "перебираемого" **объекта** или каждый номер элемента **массива**.

Пример

```
//Зададим переменную, которой поочередно будут присваиваться все элементы массива
var y;
//Создадим массив с произвольными элементами
var favcity = new Array();
favcity[1]='Москва';
favcity[2]='Нью-Йорк';
favcity[3]='Лондон';
favcity[4]='Берлин';
//Выведем все элементы массива с помощью for..in
for (y in favcity) {
   document.write('Мой любимый город № '+y+' '+favcity[y]+'.<br />');
}
```

Мой любимый город № 1 Москва
Мой любимый город № 2 Нью-Йорк
Мой любимый город № 3 Лондон
Мой любимый город № 4 Берлин

События

События - это функции, которые могут быть привязаны к элементам HTML страниц.

Код событий выполнится только после того, как произойдет их **активирующее действие**. Разные типы событий имеют разные активирующие действия.

Примеры активирующих действий JavaScript:

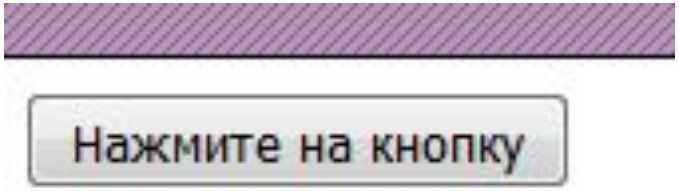
- Щелчок мыши (*событие onclick*);
- Нажатие клавиши (*onkeypress*);
- Отправление формы (*onsubmit*);
- Наведение курсора мыши на элемент (*onmouseover*) или выведение курсора мыши за пределы границ элемента (*onmouseout*);
- Полная загрузка страницы или картинки (*onload*);
- Изменение содержимого элемента, например содержимого текстового поля формы (*onchange*).

Событие onclick

Код события **onclick** будет выполнен после того, как пользователь щелкнет на элемент, к которому привязано это событие.

Пример

```
function messageShow() {
   document.write('Вы нажали на кнопку!');
}
```

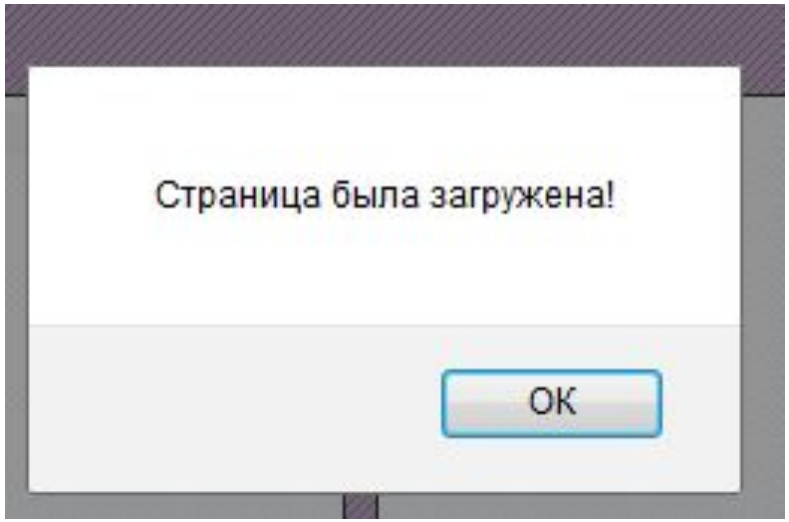


Событие onload

Код события **onload** будет выполнен при полной загрузке страницы.

Пример

```
function messageShow() {
   alert('Страница была загружена!');
}
```



События onmouseover и onmouseout

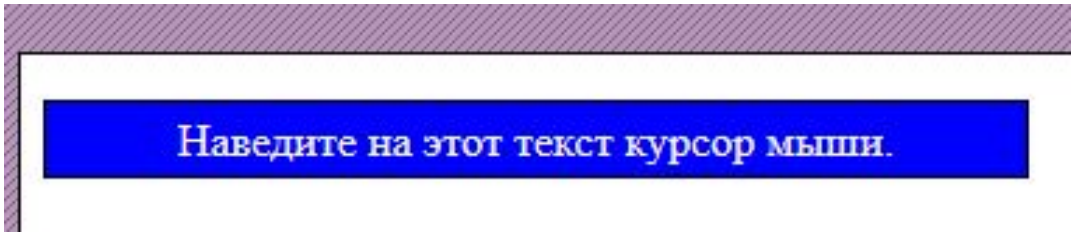
События **onmouseover** и **onmouseout** часто используются, чтобы создать "анимированные" кнопки.

Код события onmouseover будет выполнен, когда на элемент, к которому привязано событие, будет наведен курсор мыши.

Код события onmouseout будет выполнен при выведении курсора мыши за пределы элемента.

Пример

```
function textchange() {
   document.getElementById('mes').style.backgroundColor='green';
}
function textreturn() {
   document.getElementById('mes').style.color='white';
   document.getElementById('mes').style.backgroundColor='blue';
}
```



JS Ошибки

Выражение try..catch

Выражение **try..catch** позволяет проверять участки кода на наличие ошибок.

Блок **try** содержит код, который проверяется на ошибки.

Блок **catch** содержит код, который будет выполнен если в блоке try будет найдена ошибка.

Синтаксис:

```
try  {
   //Код, который проверяется на наличие ошибок
}
catch(ошибка) {
   //Код, который будет выполнен если в блоке try были найдены ошибки
}
```

```
//Проверим участок кода на ошибки
try
{
//В данном коде ошибка (dddocument вместо document)
dddocument.write('Привет всем!');
}
catch (er)
{
//Выведем пояснение к ошибке
document.write(er);
}
```

ReferenceError: dddocument is not defined

Блок finally

Данный блок является необязательной частью конструкции try..catch.

Код находящийся в данном блоке начинает выполняться после исполнения кода в блоках try и catch, но перед исполнением команд, которые следуют за данной конструкцией.

Команда throw

Если автоматически сгенерированное пояснение к ошибке Вас не устраивает можете использовать команду **throw**, чтобы создавать собственные пояснения к возможным ошибкам.

Синтаксис:

Выполнение кода по расписанию.

В JavaScript код может выполняться по расписанию например, Вы можете задать, чтобы код функции начал выполняться только через 30 секунд после ее вызова.

Для выполнения кода по расписанию в JavaScript предусмотрено два специальных метода:

setTimeout(код,промежуток_времени) позволяет выполнить **код** через заданный **промежуток_времени** один раз.

setInterval(код,промежуток_времени) позволяет выполнять **код** через заданный **промежуток_времени** бесконечное количество раз.

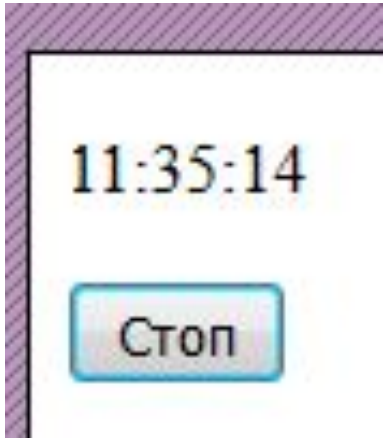
Остановка кода вызванного по расписанию

Методы **setTimeout()** и **setInterval()** после их запуска возвращают **идентификатор вызова**.

Используя этот идентификатор и методы clearTimeout() и clearInterval() Вы можете остановить выполнение кода вызванного по расписанию.

С помощью clearTimeout(идентификатор_вызова). Вы можете остановить выполнение кода вызванного методом setTimeout().

С помощью **clearInterval(идентификатор_вызова)** Вы можете остановить выполнение кода вызванного методом setInterval().



Специальные операторы

Оператор delete

С помощью оператора **delete** Вы можете удалить указанный элемент. Оператор возвращает true если удаление указанного элемента прошло успешно, и false если нет.

Обратите внимание: оператор delete не может удалить, переменную объявленную, с помощью var.

Оператор in

С помощью оператора **in** Вы можете узнать имеется ли произвольное свойство у указанного объекта или массива. Метод вернет true если указанное свойство имеется, и false, если нет.

Оператор instanceof

Оператор instanceof сверяет тип объекта с переданным значением. Если они совпадают, метод возвращает true, если нет false.

Оператор typeof

Оператор typeof возвращает тип указанного объекта.

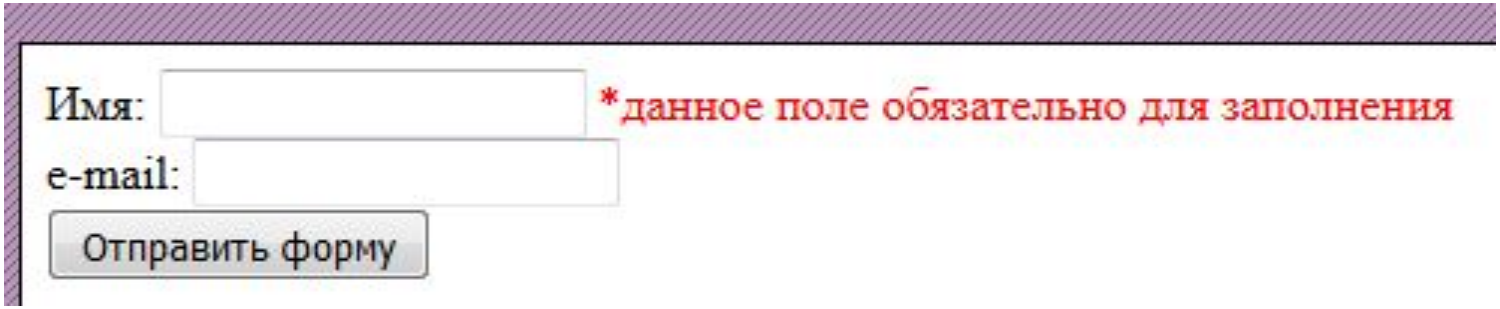
Проверка форм в JavaScript

Примеры проверок, которые возможно реализовать с помощью JavaScript:

- Было ли данное поле заполнено;
- В правильном ли формате пользователь указал свой адрес или email в соответствующем поле;
- Совпадают ли значения введенные в два различных поля (часто используется для полей с паролем);
- Не превышает ли значение введенное в поле максимально допустимую длину и т.д.

Проверка форм в JavaScript возможна благодаря событию **onSubmit**.

Если у тэга form атрибут onsubmit="return true" форма будет отправлена на сервер, если же onsubmit="return false", то форма на сервер отправлена не будет.



Вставка специальных символов

Где могут использоваться специальные символы? Предположим мы хотим вывести окно оповещения, в тексте которого есть переводы строки.

Мы не можем использовать внутри alert() тэг
 (и другие тэги HTML), но в JS есть специальный символ \n позволяющий переводить текст на новую строку.

Пример

```
alert('Сообщение\n с переводами\nстроки');
```

В таблице ниже содержатся специальные символы, которые присутствуют в JavaScript:

Код спецсимвола	Пояснение
\n	Перевод строки
\\	Обратный слэш
\t	Табуляция
\'	Одиночная кавычка
\"	Двойная кавычка
\&	Амперсанд

ЛЕКЦИЯ 7. Объектно-ориентированное программирование в JavaScript

Обратите внимание: объектно-ориентированный стиль программирования сейчас является доминирующим и почти все популярные языки программирования поддерживают его. Этот стиль стал популярным в основном благодаря тому, что он повторяет наш обычный способ мышления.

Объектно-ориентированное программирование - это стиль программирования ориентированный на работу с **объектами**.

В реальном мире мы можем рассматривать как объекты все, что нас окружает. Объектом может быть: компьютер, машина, самолет, дом, гитара. В JavaScript объектами являются строки, массивы, регулярные выражения и т. д.

Объекты могут иметь **свойства** и **методы**.

Обратите внимание: рассмотрим только **встроенные объекты**, однако JavaScript позволяет также создавать свои собственные объекты.

Свойства объектов в JavaScript

Свойства являются значениями, которые связаны с объектами.

Если рассматривать автомобиль как объект, то его свойствами будут: количество дверей, марка автомобиля, год выпуска, объем двигателя и т. д.

Объекты в JavaScript также могут иметь свойства, например объект массив имеет свойство length позволяющее узнать количество элементов в этом массиве.

При обращении к свойству объекта необходимо отделить его точкой от названия объекта (объект.свойство). Если название свойства состоит из двух (или более) слов необходимо удалить пробел между ними и начать второе слово с заглавной буквы или заменить этот пробел на _ (знак нижнего подчеркивания), например объект.маркаАвтомобиля или объект. марка_автомобиля.

Пример

```
//Создадим массив
var x=new Array();
//Запишем в массив элементы
x[0]='1 элемент массива';
x[1]='2 элемент массива';
x[2]='3 элемент массива';
x[3]='4 элемент массива';
//Посчитаем сколько элементов находится в массиве x и выведем результат на страницу
document.write('Массив содержит '+x.length+' элемента.');
```

Массив содержит 4 элемента.

Методы объектов в JavaScript

Методы являются действиями, которые могут быть совершены над объектами.

Если рассматривать автомобиль как объект, то его методами могут быть: поехать, затормозить, переключить скорость.

При обращении к методу объекта необходимо отделить его точкой от названия объекта и добавить после него круглые скобки, например объект.метод(). Если название метода состоит из двух (или более слов) необходимо удалить пробел между ними и начать второе слово с заглавной буквы, или заменить этот пробел на _ (знак нижнего подчеркивания), например объект.переключитьСкорость() или объект.переключить_скорость().

Объекты в JavaScript также могут иметь методы, например объект массив имеет метод reverse() позволяющий изменять порядок следования элементов в массиве на противоположный:

Массивы в JavaScript

Массивы представляют собой контейнеры, в которых может быть сохранено неограниченное количество переменных.

Каждому значению, которое заносится в массив присваивается уникальный идентификатор, по которому Вы затем сможете обращаться к данному элементу внутри массива.

В каких случаях удобно использовать массивы? Предположим у Вас имеется список вещей, которые Вы желаете купить в ближайшем будущем. Вы можете записать его в переменные следующим образом:

```
item1 = "Автомобиль";
item2 = "Микроволновая печь";
item3 = "Стиральная машина";
item4 = "Пылесос";
```

Создание списка таким образом может быть удобно если он содержит только несколько вещей как в данном примере, но что если список вещей, которые Вы хотите приобрести содержит более 100 наименований и Вам необходимо провести его сортировку по алфавиту или извлечь элемент имеющий определенное значение с помощью циклов?

Иными словами список составленный данным образом является набором ничем не связанных переменных, а массив является конструкцией специально предназначенной для хранения таких списков и имеет набор встроенных методов и свойств для проведения различных операций над ними.

Создание массивов

Вы можете создать массивы тремя разными способами:

Первый способ:

```
// Создаем список желаемых вещей из предыдущего примера с помощью массива
item = new Array();
item[0] = "Автомобиль";
item[1] = "Микроволновая печь";
item[2] = "Стиральная машина";
item[3] = "Пылесос";
```

Второй способ:

```
var item = new Array("Автомобиль", "Микроволновая печь", "Стиральная машина",
"Пылесос");
```

Третий способ:

```
var item = ["Автомобиль", "Микроволновая печь", "Стиральная машина", "Пылесос"];
```

Доступ к переменным в массиве

Для того, чтобы обратиться к элементу сохраненному в массиве необходимо указать имя массива и индекс желаемого элемента в квадратных скобках (массив[индекс]).

Обратите внимание: нумерация индексов в массивах начинается **не с 1, а с 0**.

Значения элементов в массивах могут быть изменены.

Свойства объекта Array

С помощью свойства **length** Вы можете узнать количество элементов в массиве.

Методы объекта Array

С помощью метода **concat()** Вы можете объединить два и более массива в один.

С помощью метода **sort()** Вы можете отсортировать значения массива.

С помощью метода **pop()** Вы можете удалить последний элемент массива.

Метод **slice(начало, конец)** позволяет извлечь (вырезать) элементы из массива и создать из извлеченных элементов новый массив.
начало указывает индекс элемента, с которого начнется извлечение.
конец указывает индекс элемента массива, на котором закончится извлечение. Если данный параметр не будет задан извлечение будет проведено до конца строки.

Объект String

Объект String (строковый объект) используется для хранения и обработки текстовой информации.

Синтаксис:

```
// Создание объекта String (первый вариант)
x=new String("произвольный текст");
// Создание объекта String (второй вариант)
x="произвольный текст";
// или
x='произвольный текст';
```

Свойства объекта String

С помощью свойства **length** Вы можете узнать длину строки.

Пример

```
//Создадим строку
var x=new String('Это строка произвольного текста.');
// Узнаем длину строки (количество символов учитывая пробелы) и выведем ее на страниц
document.write(x.length);
```

Методы объекта String

С помощью метода **toUpperCase()** Вы можете перевести все символы текста в верхний регистр, а с помощью **toLowerCase()** - в нижний.

Пример

```
//Создадим строку
x='ЭтО СтрОка';
//Выведем строку без изменений
document.write(x + '<br />');
//Переведем все буквы строки x в верхний регистр и выведем результат на страницу
document.write(x.toUpperCase()+'<br />');
//Переведем все буквы строки x в нижний регистр и выведем результат на страницу
document.write(x.toLowerCase()+'<br />')
```

ЭтО СтрОка
ЭТО СТРОКА
это строка

Обратите внимание: помните о важности регистра букв в JavaScript при обращении к методам, а также не забывайте о круглых скобках (), которые должны добавляться после их названия.

С помощью метода **concat()** Вы можете объединить две и более строки и вывести результат на страницу.

С помощью метода **replace()** Вы можете заменить одно произвольное слово в строке на другое.

Пример

```
//Создаем строку
x='<b>HTML</b> - это скриптовый объектно ориентированный язык программирования. ';
//Заменим слово 'HTML' на 'JavaScript' в строке x и выведем результат на страницу
document.write(x.replace('HTML','JavaScript'));
```

JavaScript - это скриптовый объектно ориентированный язык программирования.

Объект Date

Объект Date позволяет производить различные операции с датой и временем.

Синтаксис:

```
//Определим текущую дату и запишем ее в переменную x
x=new Date();
//При выводе на страницу текущая дата (24 Декабря 2010) будет выглядеть
//следующим образом:
Fri Dec 24 2010 22:15:31 GMT+0600

/* Fri = Friday (Пятница) - обозначает текущий день недели
   Dec = December (Декабрь) - обозначает текущий месяц
   24 - обозначает день месяца
   2010 - обозначает год
   22:15:31 - текущее время
   GMT+0600 - смещение времени от Гринвича +6 часов
*/
```

Методы объекта Date

Метод **getDate()** позволяет извлечь из объекта день месяца.

Пример

```
//Определим текущую дату и запишем результат в x
x=new Date();
//Извлечем день месяца из объекта x и выведем результат на страницу
document.write(x.getDate());
```

Метод **getFullYear()** позволяет извлечь из объекта заданный год.

Метод **setFullYear(год, месяц, число_месяца)** позволяет изменить дату, заданную в объекте, на желаемую.

Объект Math

Используя свойства и методы **объекта Math** Вы можете выполнять в JavaScript различные математические операции.

Обратите внимание: для того, чтобы обращаться к свойствам и методам математического объекта его не нужно (в отличие от остальных встроенных объектов JavaScript) предварительно создавать.

Свойства объекта Math

Свойства данного объекта содержат значения часто используемых математических констант:

Пример

```
//Выведем значение числа Пи на страницу
document.write(Math.PI+'<br />');
//Выведем значение экспоненты
document.write(Math.E+'<br />');
//Выведем значение натурального логарифма 10
document.write(Math.LN10+'<br />');
//Выведем значение квадратного корня 2
document.write(Math.SQRT2);
```

```
3.141592653589793
2.718281828459045
2.302585092994046
1.4142135623730951
```

Методы объекта Math

С помощью методов объекта Вы можете производить над числами различные математические операции.

Метод **round()** позволяет округлять числа до ближайшего целого.

Пример

```
//Округлим число 25.34 до ближайшего целого и выведем значение на страницу
document.write(Math.round(25.34)+'<br />');
//Округлим число 25.88 до ближайшего целого и выведем значение на страницу
document.write(Math.round(25.88));
```

Метода **random()** позволяет генерировать случайные числа между 0 и 1.

Пример

```
//Сгенерируем случайное число между 0 и 1 и выведем его на страницу
document.write(Math.random()+'<br />');
/* Вы также можете генерировать случайные числа в произвольном промежутке. Например д
того, чтобы сгенерировать случайное число в промежутке от 0 до 100 нужно домножить
число, полученное с помощью метода random(), на 100 и затем округлить его до
ближайшего целого */
document.write(Math.round(Math.random()*100));
```

Метод **pow(число,степень)** позволяет возводить числа в степень.

Пример

```
//Возведем 3 в 3 степень
document.write(Math.pow(3,3) + '<br />');
//Возведем 4 в 0.5 степень (найдем корень из 4)
document.write(Math.pow(4,0.5) + '<br />');
//Возведем 5 в -1 степень
document.write(Math.pow(5,-1));
```

```
27
2
0.2
```

Методы **max** (максимум) и **min** (минимум) выбирают максимальные и минимальные числа из предложенных.

Пример

```
//Выберем из чисел 10 68 35 12 44 максимальное и выведем его на страницу
document.write(Math.max(10,68,35,12,44) + '<br />');
//Выберем из чисел 10 68 35 12 44 минимальное и выведем его на страницу
document.write(Math.min(10,68,35,12,44));
```

Регулярные выражения

Регулярные выражения позволяют производить гибкий поиск слов и выражений в текстах с целью их удаления, извлечения или замены.

Синтаксис:

```
//Первый вариант создания регулярного выражения
var regexp=new RegExp(шаблон,модификаторы);
//Второй вариант создания регулярного выражения
var regexp=/шаблон/модификаторы;
```

шаблон позволяет задать шаблон символов для поиска.
модификаторы позволяют настроить поведение поиска:
i - поиск без учета регистра букв;
g - глобальный поиск (будут найдены все совпадения в документе, а не только первое);
m - многострочный поиск.

Поиск слов и выражений

Самым простым применением регулярных выражений является поиск слов и выражений в различных текстах.

Специальные символы

Помимо обычных символов в шаблонах регулярных выражений могут использоваться **специальные символы** (метасимволы). Специальные символы с описаниями приведены в таблице ниже:

Специальный символ	Описание
.	Совпадает с любым символом, кроме символа конца строки.
\w	Совпадает с любым буквенным символом.
\W	Совпадает с любым не буквенным символом.
\d	Совпадает с символами, которые являются цифрами.
\D	Совпадает с символами, которые не являются цифрами.
\s	Совпадает с пробельными символами.
\S	Совпадает с не пробельными символами.
\b	Совпадения будут искаться только на границах слов (в начале или конце).
\B	Совпадения будут искаться только не на границах слов.
\n	Совпадает с символом перевода строки.

Символы в квадратных скобках

Используя квадратные скобки **[кейу]** Вы можете указать группу символов, поиск которых нужно произвести.

Символ ^ перед группой символов в квадратных скобках **[^квг]** говорит о том, что нужно произвести поиск всех символов алфавита кроме заданных.

Используя тире (-) между символами в квадратных скобках **[а-з]** Вы можете задать диапазон символов, поиск которых нужно произвести.

С помощью квадратных скобок **[012]** Вы можете также искать числа.

Квантификаторы

Квантификатор - это конструкция позволяющая задать сколько раз предшествующий ей символ или группа символов должна встречаться в совпадение.

Использование круглых скобок

Заключая часть шаблона регулярного выражения в круглые скобки Вы указываете выражению запомнить совпадение, найденное этой частью шаблона. Сохраненное совпадение может использоваться позднее в Вашем коде.

К примеру регулярное выражение /(Дмитрий)\sВасильев/ найдет строку 'Дмитрий Васильев' и запомнит подстроку 'Дмитрий'.

Методы

С помощью метода **exec()** Вы можете производить поиск желаемого значения. Метод возвращает найденное значение.

С помощью метода **test()** Вы можете проверить содержится ли в тексте выражение заданное в регулярном выражении. Если выражение содержится в тексте метод возвращает true, а если нет false.

Создание объектов

Что такое объект?

Объект - это конструкция, которая может иметь свойства и методы. Объекты могут быть созданы двумя способами:

Синтаксис (первый способ):

Пример

```
//Создадим объект obj
var obj=new Object();
//Добавим объекту obj свойство name со значением 'Дмитрий'
obj.name='Дмитрий';
//Добавим объекту obj свойство age со значением 26
obj['age']=26;
//Обратимся к свойствам объекта для вывода их значений
document.write(obj.name+'<br />');
document.write(obj.age);
```

Дмитрий
26

Синтаксис (второй способ):

Пример

```
//Создадим объект obj со свойствами name и age, которые содержат
//значения 'Дмитрий' и 26
var obj={
   name:'Дмитрий',
   age:26
}
//Обратимся к свойствам объекта для вывода их значений
document.write(obj.name+'<br />');
document.write(obj.age);
```

В JavaScript Вы можете обращаться к свойствам объектов двумя способами:

1. Используя точку ('.') после имени объекта:

```
//Присваиваем свойству объекта произвольное значение
имя_объекта.свойство=значение
//Обращаемся к значению свойства объекта
имя_объекта.свойство
```

2. Заключая название свойства в квадратные скобки ([]) после имени объекта:

```
//Присваиваем свойству произвольное значение
имя_объекта['свойство']=значение
//Обращаемся к значению свойства
имя_объекта['свойство']
```

Свойства объектов

Свойства - это значения, связанные с объектами.

Например, объект человек может иметь следующие свойства: имя, возраст, профессия.

Пример

```
//Создадим объект human со свойствами name, age и job
var human={
   name:'Дмитрий',
   job:'Дизайнер',
   age:26
}
//Выведем значения свойств на страницу
document.write(human.name+'<br />');
document.write(human.job+'<br />');
document.write(human.age);
```

Дмитрий
Дизайнер
26

Свойства объектов

Свойства - это значения, связанные с объектами.

Например, объект человек может иметь следующие свойства: имя, возраст, профессия.

Пример

```
//Создадим объект human со свойствами name, age и job
var human={
   name:'Дмитрий',
   job:'Дизайнер',
   age:26
}
//Выведем значения свойств на страницу
document.write(human.name+'<br />');
document.write(human.job+'<br />');
document.write(human.age);
```

Дмитрий
Дизайнер
26

Методы объектов

Методы - это функции, связанные с объектами.

Например, объект человек может иметь следующие методы: идти(), сидеть(), думать().

Конструкторы объектов

В предыдущих примерах мы создавали *прямые экземпляры объектов*. В JavaScript существует еще один способ создания объектов - с помощью **конструктора объектов**.

Конструктор объектов - это шаблон, на основе которого создаются объекты. Написанный один раз конструктор, позволяет создать неограниченное количество объектов любой сложности написав лишь одну строчку кода.

Преимущества данного способа заключается в том, что он позволяет разделить логику создания объекта от его использования. Например, один человек может специализироваться на создании сложных конструкторов объектов, а другой на написании на основе этих объектов конечных программ под заказ.

Свойство prototype

С помощью свойства **prototype** Вы можете добавлять новые свойства и методы к конструкторам объектов.

Добавленные к конструктору свойства и методы будут также добавлены ко всем объектам, которые были созданы данным конструктором.

Синтаксис:

```
конструктор.prototype.свойство(метод)=значение(функция);
```

Наследование

Наследование - является важным механизмом объектно-ориентированного программирования. Наследование позволяет ускорить и упростить разработку конструкторов объектов.

С помощью механизма наследования Вы можете перенимать свойства и методы с уже имеющихся конструкторов вместо того, чтобы задавать их еще раз в ручную.

Обратите внимание: с помощью конструкции *свойство || "Стандартное значение"* мы можем присвоить свойству объекта стандартное значение в случае, если значение не было передано в конструктор при создании объекта.

Структура JavaScript

В окружении браузера JavaScript структурно состоит из трех частей:

1. Ядро (также известно как ECMAScript) является основой для функционирования остальных частей. В ядре реализуется синтаксис языка т. е. определяются ключевые и зарезервированные слова, условные конструкции, циклы, объекты и т.д. Ядро само по себе не имеет средств для вывода информации.

2. Объектная модель браузера (Browser Object Model или сокращенно BOM). С помощью BOM Вы можете управлять поведением браузера из JavaScript, считывать информацию о браузере, выполнять код по расписанию и т.д.

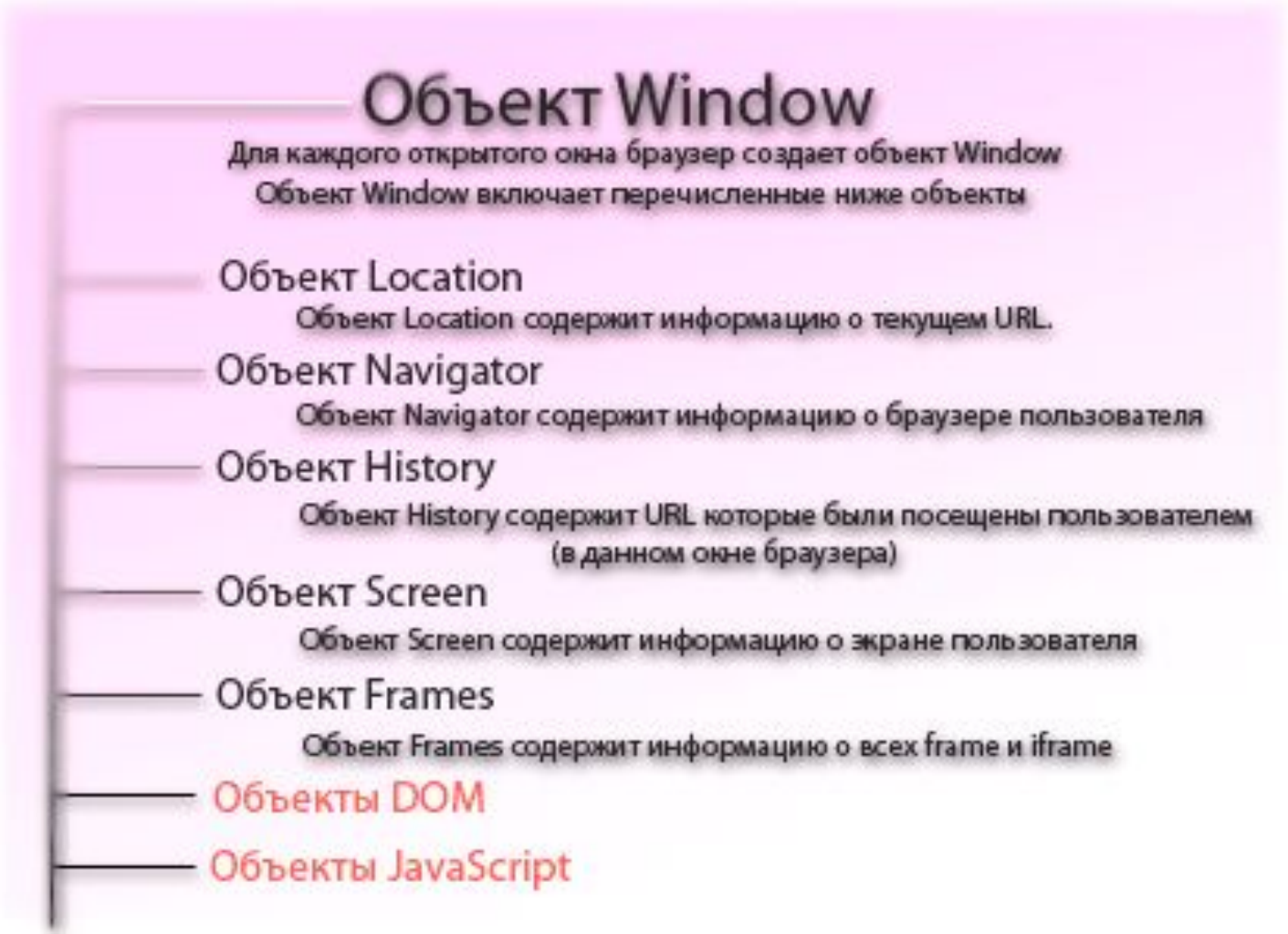
3. Объектная модель документа (Document Object Model или сокращенно DOM) это инструмент, с помощью которого языки программирования могут получать доступ к HTML и XML документам. Таким образом с помощью DOM Вы можете динамически изменять содержимое HTML и XML страниц.

BOM объекты

С помощью **объектной модели браузера (BOM)** Вы можете управлять поведением браузера из JavaScript.

BOM включает в себя несколько объектов.

BOM объекты:



ЛЕКЦИЯ 8. Язык программирования PHP

Установка сервера php. Настройка php сервера

В данном курсе мы рассмотрим такие вопросы, как:
- создавать переменные на языке программирования php,
- подключаться к базе данных,
- создавать функции и вызывать их,
- условия if-else в php,
- циклы for, while,
- подключение блоков (include),
- создание формы и так далее

Введение в язык php

Язык ПХП - это скриптовый язык программирования, который имеет в свою очередь:

- Открывающиеся теги и закрывающиеся.
- Переменные.
- Функции, которые могут принимать значения и отдавать полученные значения.
- Условия, циклы.

PHP — это серверный язык создания сценариев (или стороны сервера), разработанный специально для Web. В HTML-страницу можно внедрить код PHP, который будет выполняться при каждом ее посещении. Код PHP интерпретируется Web-сервером и генерирует HTML или иной вывод, наблюдаемый посетителем страницы.

Разработка PHP была начата в 1994 г. и вначале выполнялась одним человеком, Расмусом Лердорфом (Rasmus Lerdorf). Этот язык был принят рядом талантливых людей и претерпел три основных редакции, пока не стал широко используемым и зрелым продуктом, с которым мы имеем дело сегодня. К январю 2001 г. он использовался почти в пяти миллионах доменов во всем мире и их число продолжает быстро расти.

Первоначально PHP являлось сокращением от Personal Home Page (Персональная начальная страница), но затем это название было изменено в соответствии с рекурсивным соглашением по наименованию GNU (GNU = Gnu's Not Unix) и теперь означает PHP Hypertext Preprocessor (Препроцессор гипертекста PHP).

Что нового в PHP 4?

Имеется ряд важных усовершенствований 4 версии:

- PHP 4 работает значительно быстрее предшествующих версий, поскольку в нем используется новый механизм Zend Engine. Если требуется еще более высокая производительность, по адресу http://www.zend.com можно получить модули Zend Optimizer, Zend Cache или Zend Compiler.
- PHP всегда можно было использовать как эффективный модуль сервера Apache. С появлением новой версии PHP можно устанавливать и в виде модуля ISAPI для Internet Information Server компании Microsoft.
- Теперь поддержка сеансов является встроенной. В предшествующих версиях для управления сеансом или создания собственного сеанса требовалось устанавливать дополнительный модуль PHPLib.

Некоторые преимущества PHP. К числу конкурентов PHP относятся Perl, Active Server Pages (ASP) от Microsoft, Java Server Pages (JSP) и Allaire Cold Fusion.

PHP обладает множеством преимуществ по сравнению с этими продуктами:

- Высокая производительность
- Наличие интерфейсов ко многим различным системам баз данных
- Встроенные библиотеки для выполнения многих общих задач, связанных с Web
- Низкая стоимость
- Простота изучения и использования
- Переместимость
- Доступность исходного кода

Синтаксис и грамматика. Синтаксис PHP заимствован непосредственно из C. Java и Perl также повлияли на синтаксис данного языка.

Переход из HTML. Есть три способа выхода из HTML и перехода в "режим PHP кода":

1. <? echo("простейший способ, инструкция обработки SGML\n"); ?>
2. <?php echo("при работе с XML документами делайте так\n"); ?>
3. <script language="php">

   echo ("некоторые редакторы (подобные FrontPage) не любят обрабатывающие инструкции");

   </script>;

Типы переменных. PHP поддерживает переменные следующих типов:
integer - целое
double - число с дробной частью
string - строковая переменная
array - массив
object - объектная переменная
pdfdoc - PDF-документ (только при наличии поддержки PDF)
pdfinfo - PDF-инфо (только при наличии поддержки PDF)

Тип переменной обычно не устанавливается программистом; вместо этого, он определяется PHP во время выполнения программы, в зависимости от контекста, в котором она используется. Если вам нравится указывать тип переменной непосредственно, используйте для этого инструкцию cast либо функцию settype(), но учтите, что переменная может вести себя по-разному, в зависимости от того, какой тип определен для нее в данное время.

Все страницы с расширением php или html. Имеют строгую структуру:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
5 <title>Новый Документ</title>
6 </head>
7
8 <body>
9 </body>
10 </html>
```

Первая и четвертая строка - это кодировка.
Вторая - это открытие тега html.
Третья - это открытие тега head
Пятая - это название заголовка страницы (По умолчанию - это **Новый Документ**).
Шестая - закрытие тега head
Восьмая - открытие тега body. Вот здесь как раз мы и будем писать то, что нам нужно.
Девятая - закрытие тега body.
Десятая - закрытие тега html.

Чтобы нам в заголовке написать "**основы php 5 и mysql программирования**", на пхп коде, необходимо: в строке 5 написать следующее:

<title><php echo "**основы php 5 и mysql программирования**"; ?></title>

<?php ?> - это необходимо для обозначения области программирования php.

То есть **<?php** - это открытие пхп кода, а **?>** - это закрытие кода.

Между <body> и </body> вам нужно писать текст, который выводится в браузере.

Структура синтаксиса

$db - это переменная, необходимая для запроса. Знак **$** - применяется ТОЛЬКО для переменных.
mysql_connect - это функция подключения к базе данных. Она принимает три параметра. Путь к соединению, имя пользователя и пароль.
mysql_select_db - применяется для выборки с базы данных с определенной базы, потому что у вас может быть не одна база данных. Здесь вы выбираете конкретную базу данных.
mysql_query - происходит выборка из базы данных. На русском языке это будет выглядеть так: "**ВЫБРАТЬ заглавие, текст ИЗ таблицы, ГДЕ первая строка**".
Не забывайте ставить точку с запятой. Потому что это частая ошибка программистов.
Далее следует условие if если выборка произошла, то мы перекидываем выборку в массив для того, чтобы иметь доступ к информации и перекидываем это в переменную $myrow. Если в массиве ничего нету, мы печатаем сообщение, посредством функции echo.
Теперь переменная $myrow - это массив. И для доступа к полю title необходимо записать $myrow["title"], то есть в массиве $myrow выбираем поле title.

```php
<?php

$db = mysql_connect("localhost","name","4ad2fg6g"); //проверка имени и пароля и соединение с базой данных
mysql_select_db("baza",$db); //выборка определенной базы данных


$result = mysql_query("SELECT title,text FROM table WHERE id = '1'",$db); // выборка с базы данных из таблицы table
if(!$result) // если нет результата выбовки
{
echo "<p>Запрос выборки из базы данных не прошел. Напишите, об этом администрацию: mail@life-prog.ru. <br>Код ошибки:</p>";
exit(mysql_error()); //печатание кода ошибки
}
if (mysql_num_rows($result) > 0) //если выборка произошла
{
$myrow = mysql_fetch_array($result); //перекидываем выборку в массив
}
else {echo "Информация не может быть извлечена. В таблице нет записей";
exit(); //остановка вывода в браузер
}


?>
```

Php if or and && | Условия в PHP

Условия в php имеют ключевое слово **if**. И логические операторы **OR** или **||**, **AND** или **&&**.
Конструкция условия if следующая: if (условие) тогда действие;

Рассмотрим пример использования **or and** в операторе **if**

```
<?php
$x = 1;
if (isset($x)) echo "переменная x существует";
//////////////////////////////////////////////////
if (isset($x) && $x == 1 ) echo "<br>x = 1";
//////////////////////////////////////////////////
$x = $x+1;
if (isset($x) and $x == 2 ) echo "<br>x = 2";
//////////////////////////////////////////////////
$x++;
if (isset($x) and $x == 3 && $x != 0) echo "<br>x = 3 and x != 0";
?>
```

Итак, мы видим, что в первом случае, есть только одно условие if (isset($x)), другими словами: если переменная $x существует, тогда выводится сообщение echo "переменная x существует";

Второе условие более сложнее: if (isset($x) && $x == 1) - если переменная существует и равна 1, тогда выводим echo "
x = 1";
 - это перевод строки.

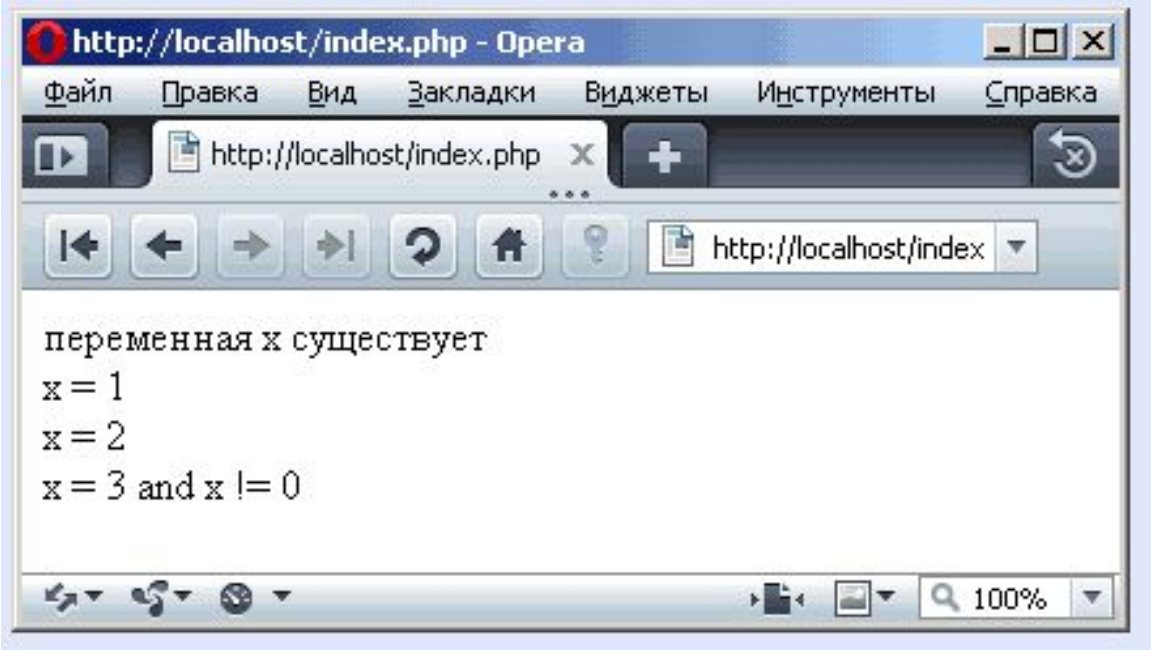
Логический **оператор AND** выполняет условие только в том случае, если все условия истинные. Если хотя бы одно условие ложно, то все условие ложно. Вот почему оператор AND иногда сравнивается с математическим умножением. 451*6845*655*0 = 0. Какие бы не были числа, но если в выражении есть ноль, то все выражение нулевое.

Третье условие аналогично второму, только вместо логического оператора && используем оператор and. Этим показывает, что логические операторы **&&** и **and** имеют тот же смысл.

Четвертое условие показывает сложные условия, которые вы можете использовать в своих программах.

```
$x = $x+1;

$x++;
```

Две строки выше имеют один и тот же смысл. Они увеличивают переменную на единицу.
Результат работы данного кода будет следующим:



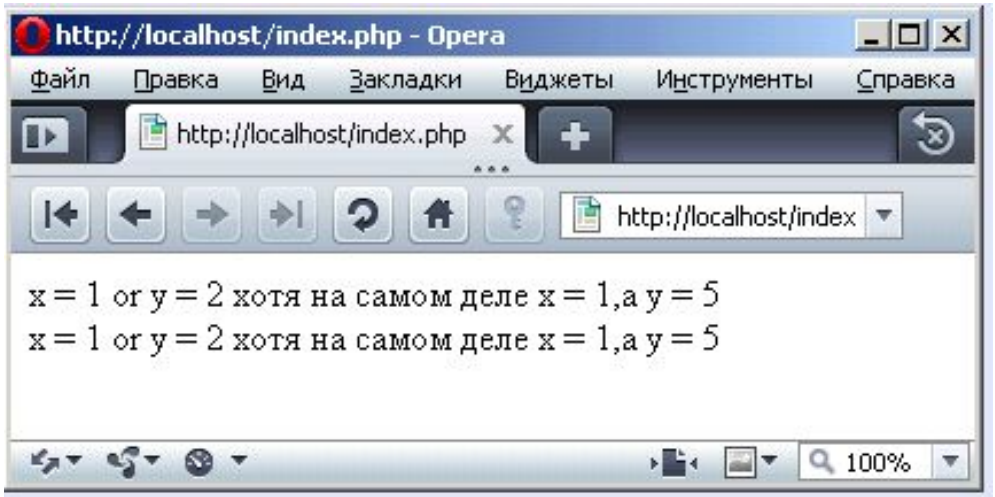
Логический оператор OR отличается от оператора AND тем, что если в условии есть хоть одно истинное условие, то все условие истинно. Вот почему оператор OR иногда сравнивается с математическим сложением. 0+0+0+1 = 1. Если в выражении много нулей, но есть хотя бы одна единица, то все выражение ненулевое.

Пример:

```
<?php
$x = 1;
$y = 5;
if ($x == 1 or $y ==2) echo "x = 1 or y = 2 хотя на самом деле x = $x,a y = $y";
if ($x == 1 || $y ==2) echo "<br>x = 1 or y = 2 хотя на самом деле x = $x,a y = $y";
?>
```

Мы видим, что на самом деле выполняется только одно условие, второе условие ложно, но благодаря оператору OR все выражение истинно. Также мы видим, что операторы **OR** и **||** идентичны.

Результат работы данного кода будет следующим:



Php if else else if | Условия

Сложные условия в php имеют ключевое слово **if else**. И логические операторы **OR** или **||**, **AND** или **&&**.
Конструкция сложного условия if следующая:
if (условие) тогда действие;
else if (условие) тогда действие;
...
else тогда действие;

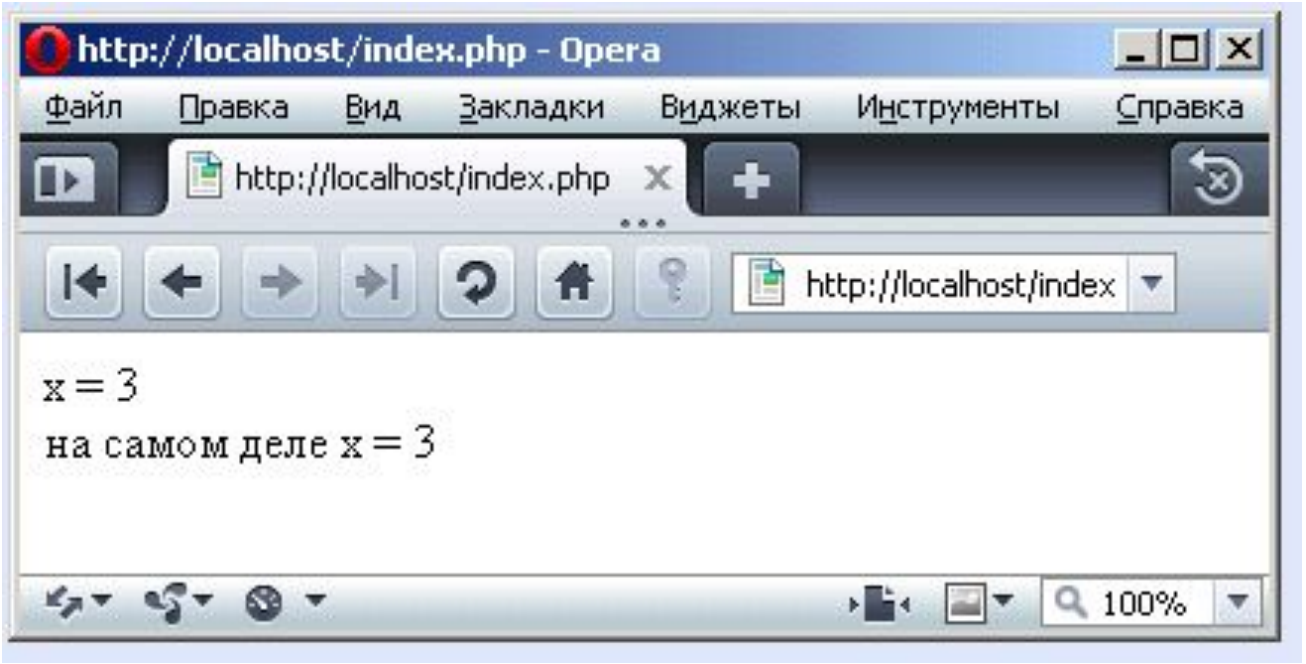
Рассмотрим пример использования сложного операторе **if**:

```
<?php
$x = rand(1,3);

if ($x == 1) echo "x = 1";
else if ($x ==2) echo "x = 2";
else echo "x = 3";

echo "<br>на самом деле x = $x";
?>
```

Итак, мы видим, что есть генерация случайных чисел от 1 до 3 и условия, если x == 1, тогда вывод echo "x = 1"; иначе, если x == 2, тогда echo "x = 2";, иначе без сомнения x == 3.

Ниже мы выводим настоящее число x и видим, что все работает отлично.

Результат работы данного кода будет следующим:



Php for цикл

Цикл for в php обычно используется для одномерных или двумерных массивов или циклических выражений.

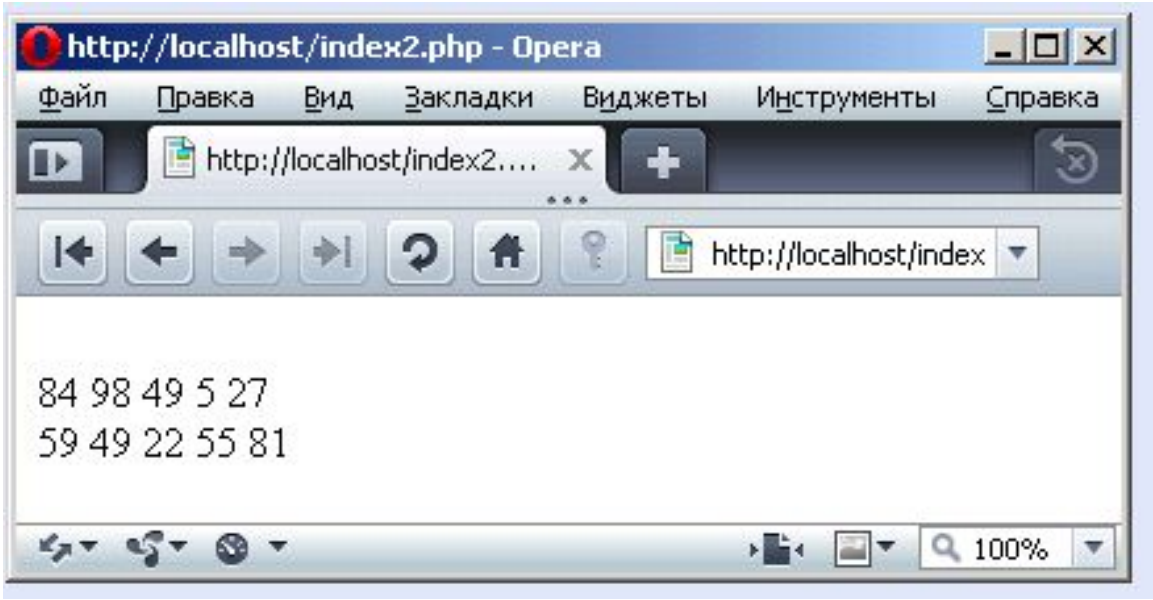
Структура цикла **for** следующая:

for (начальное значение ; условие ; инкремент) действия

Приведем пример программного кода заполнение массива и вывода результата:

```
<?php
$i = 0;
$mas[10] = 0;
////////////////////////////////////
for ($i = 0;$i<10;$i++){
$mas[$i] = rand(0,100); //заполнение массива случайными числами
}
////////////////////////////////////
for ($i = 0;$i<10;$i++){
if ($i % 5 == 0) echo "<br>";
echo "$mas[$i] "; //вывод массива
}
?>
```

Цикл for в php похож на синтаксис языка Си. rand(0,100); - это функция случайных величин, которая принимает случайные значения от 0 до 100.



Php while do

Цикл do while применяется там, где должно выполниться хотя бы одно действие.

Php while do имеет следующую структуру:

do
{
действие
}
while(условие);

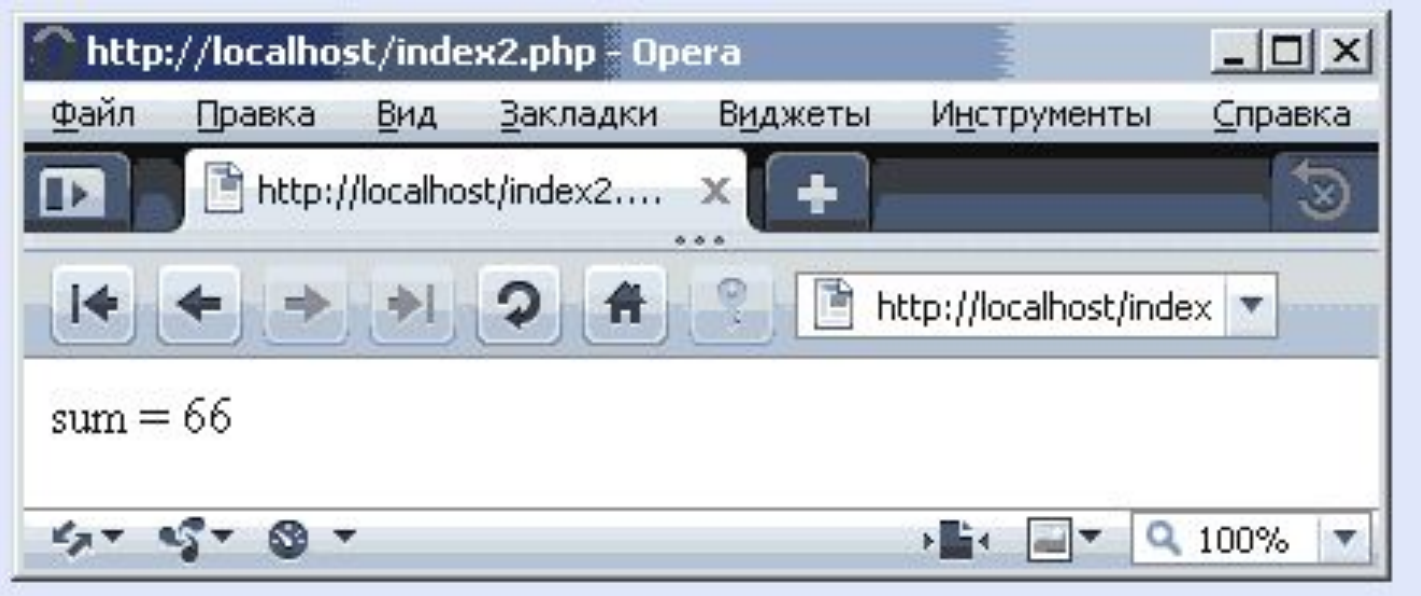
То есть сначала идет действие, после этого проверяется условие, по которому либо выйти из цикла, либо продолжать действия.

Напишем подсчет арифметической прогрессии.

```
<?php
$sum = 0;
$i = 0;
////////////////////////////////
do
{
$sum = $sum + $i; //арифметическая прогрессия
}
while($i++ <= 10);
echo "sum = $sum";
?>
```

Как видим сумма принимает значение суммы плюс переменная $i, которая в условии инкрементируется и проверяется условие ($i++ <= 10).

Результат работы:



Применение циклов в Php

Циклы в php имеют много применения. Давайте сделаем нахождение арифметической прогрессии, заполнение массива случайными числами, вывод массива и сортировку массива по убыванию.

```php
<?php
$sum = 0;
$i = 0;
$mas[10] = 0;

////////////////////////////////////
do
{
$sum = $sum + $i; //арифметическая прогрессия
}
while($i++ <= 10);
echo "sum = $sum<br>";

////////////////////////////////////
for ($i = 0;$i<10;$i++){
$mas[$i] = rand(0,100); //заполнение массива случайными числами
}

////////////////////////////////////
for ($i = 0;$i<10;$i++){
if ($i % 5 == 0) echo "<br>";
echo "$mas[$i] "; //вывод массива
}
```

В первой цикле мы находим арифметическую прогрессию и результат выводим на экран.

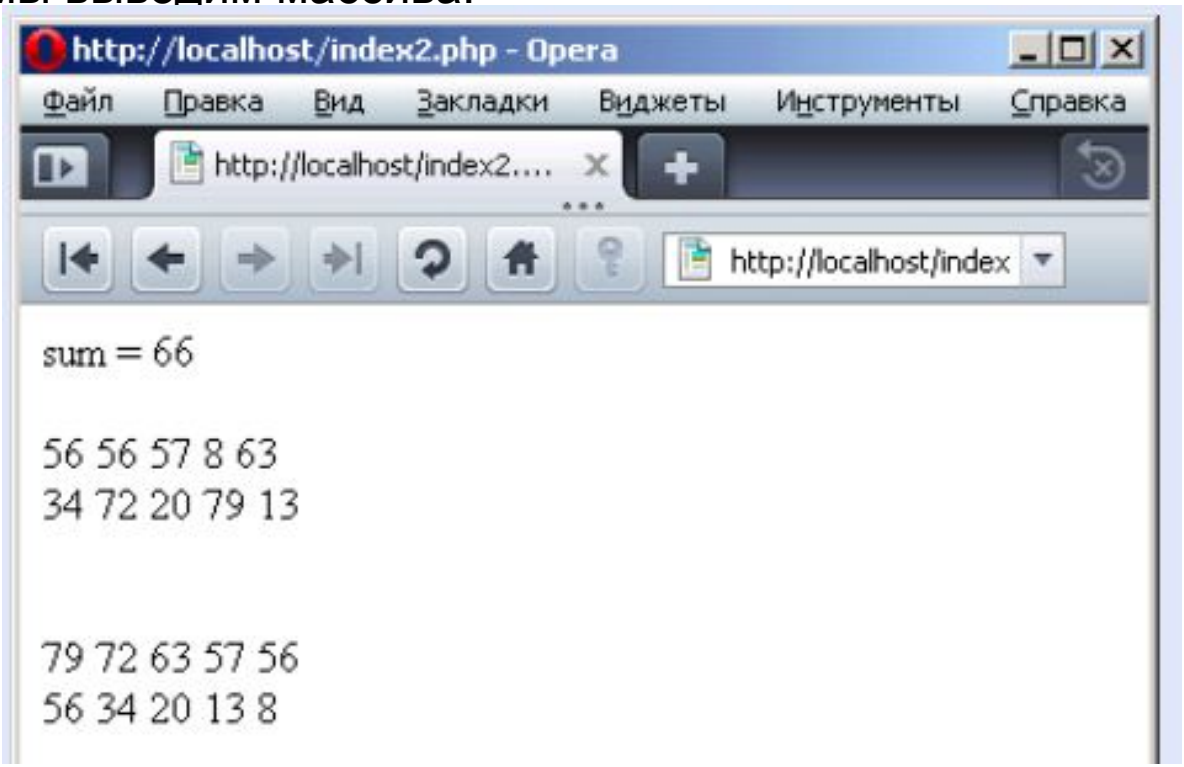
Во втором цикле мы заполняем массив случайными числами от 0 до 100.

В третьем цикле мы выводим массива

В четвертом цикле происходит сортировка массива по убыванию. Идентификатор $ident показывает сортировка закончена или нет, ведь если она не закончена, то $ident = 1, иначе $ident = 0, а значит с цикла нужно выходить.

В пятом цикле мы выводим массива.

Результат работы:



Генератор случайных чисел и слов в PHP

Для чего можно использовать генератор случайных чисел и слов?

Например, случайный пароль. Рассмотренный ниже скрипт (скорее php код) очень хорошо подходит для таких функций как верификация, создание инвайтов, создание автоматического сложного пароля и последующего его отправки на почту и многого др.

Также генератор можно использовать как защитный код (или капча). Такой код может выводиться, например, в виде картинки. В общем, приведенный ниже скрипт генерации, очень полезен во многом.

Сначала необходимо задать область случайных символов. Делать это будим с помощью двух функций: range и array_merge()
range – эта функция создает диапазон значений, например:
range('A', 'Z') – создаст диапазон от A до Z включительно.
range('0', '9') – создаст диапазон от 0 до 9 соответственно.
array_merge() – слияние массивов.
Создаем переменную $code

```
$code= array_merge( range('A', 'Z'), range('0', '9'));
```

она сливает массивы, перечисленные в ее аргументах, в один большой массив и возвращает результат.
Создаем сеансовую переменную $_SESSION и присваеваем ей пустоту

```
$_SESSION['code'] = ' ';
```

Запускаем цикл пока не достигнет наш случайный код 5-и знакам.

```
for ($i = 0; $i < 5; $i++)
```

И наконец, формируем код:

```
$_SESSION['code'] .= $code[array_rand($code)];
```

Ну и проверяем работоспособность нашего кода

```
echo $_SESSION['code'];
```

Вот как будет выглядеть наш PHP код:

```
<?PHP
$code= array_merge( range('A', 'Z'), range('0', '9'));
$_SESSION['code'] ='';
for ($i = 0; $i < 5; $i++)
$_SESSION['code'] .= $code[array_rand($code)];
?>
```

Еще один очень интересный момент:
Часто из защитного кода, для его лучшего распознания, выкидывают плохо читабельные символы, такие как 0 O 1 J I ну и далее на ваше усмотрение. Делается это так.
$code= array_merge(range('A', 'H'), array('K', 'M', 'N', 'P'),range('R', 'Z'), range('1', '9'));
В данном случае выкинуты I, J, L, O,0 и Q. array – это ряд конкретных значений.

Создание картинки в Php

В начале создаем php файл, например img.php

Создаем переменную, допустим, $text и присваиваем ей какое либо значение:

```
$text='Privet!';
```

Создаем пустое изображение 60x20 пикселей

```
$im = imagecreate(60, 20);
```

Задаем цвет картинки $im

```
imagecolorallocate($im, 255, 255, 255);
```

Эта функция возвращает идентификатор цвета для изображения $im. В данном случае – это белый цвет.

Задаем цвет нашего текста в картинке $im

```
$color = imagecolorallocate($im, 0, 125, 0);
```

Горизонтальное рисование строки $text на изображении $im

```
imagestring($im, 4, 3, 2, $text, $color);
```

Здесь:
$im – наш рисунок;
4 – значение встроенного TrueType шрифта, причем значения могут быть от 1 до 5. Очень ограниченные возможности шрифта;
3 – отступ текста от левого края в пикселях;
2 – отступ текста от верхнего края в пикселях;
$text – собственно наш текст;
$color – цвет текста, который мы задали.

При выводе изображения непосредственно в браузер необходимо передать браузеру mime-тип выводимых данных. Это следует сделать с помощью функции header ()

```
header('Content-type: image/png');
```

Выводим изображение в браузер в формате PNG

```
imagepng($im);
```

Вот как будет выглядеть php код созданной картинки:

```php
<?php
$text='Privet!';
$im = imagecreate(60, 20);
imagecolorallocate($im, 255, 255, 255);
$color = imagecolorallocate($im, 0, 125, 0);
imagestring($im, 4, 4, 2, $text, $color);
header('Content-type: image/png');
imagepng($im);
?>
```

PHP Скрипт – простейший счетчик посещений

Рассмотрим **простейший PHP счетчик,** который имеет защиту от обновления страницы. Если посетитель обновит страницу, то этот счетчик все равно покажет одно посещение. Это значит что скрипт будет защищать от такой коварной клавиши как F5 при зажатии которой можно получить за минуту более 1000 посетителей.
Этот скрипт содержит 1 PHP файл и 1 mySQL таблицу.

База данных, допустим "my_site" Создаем таблицу "counter" с одним полем visitor(Int, 11). И вставляем первую строку с нулем.

```
<? php
   // Для использования session переменных поместите этот код в самое начало страницы
session_start ();

   // Соединение с датабазой
mysql_connect ("localhost","","");
mysql_select_db ("my_site");

   /* Проверка session переменной "visitor". Если она не существует, обновляем базу данных и создаем эту сессию через  Session ID.*/
if (!session_is_registered("visitor"))
{
$visitor=session_id();
session_register("visitor");

   // Обновляем значение в столбце "all_visitor" добавляя +1.
mysql_query ("update counter set all_visitor=all_visitor+'1'");
}
?>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<title>Счетчик</title>
</head>
<body>
Количество просмотров:
<? php
   // Вытягиваем данные из counter и вкладываем результат в переменную $result.
$result=mysql_query("select * from counter");
$row=mysql_fetch_assoc($result);

   // Выводим all_visitor запись.
echo $row['all_visitor'];

   // Закрываем связь с базой данных.
mysql_close();
?>
</body>
</html>
```

Форма обратной связи php | Скрипт формы на php

Наша форма заказа на php будет иметь название zakaz.php. Форма обратной связи делается при помощи ключевого слова form. Поэтому мы пишем:

<form action="zakaz.php" method="post" name="form_zakaz"> - здесь мы указываем путь при нажатии на кнопку "Заказать", и выбираем метод пост. Имя нашей формы - это form_zakaz. После чего не нужно забывать закрыть эту форму: </FORM>
Функция mail - это функция отправки сообщения на почту. К примеру, мы отправляем письмо к адресату: mail@mail.ru, тема сообщения "Заказ продукции", а само сообщение идет дальше, то есть код следующий: mail("mail@mail.ru", "Заказ продукции", "Здесь само сообщение к этому адресату.");
В этом коде мы подключаем стили: <link href="style.css" rel="stylesheet" type="text/css">
Поэтому вам придется создать файл style.css, в котором будет следующий код:

```
body { background:#446622;
font-family:Verdana;
font-size:14px;
}

p {font-family:Verdana, Arial, Helvetica, sans-serif; font-size:13px;
color:#5500FF; margin:15px;}
a {color:#006600;}
td {background:#C6C6C6;}

.head { color:#443399; font-family:Verdana; font-size:16px; text-align:center;
font-weight:bold;}
.red { color:#FF0000;}
.form {font-size:11; color:#000000; font-family: verdana, serif;}
.form2 {border: 1px black solid; font-size: 11px; width:300; color:black;}
.zakaz {border: 1px black solid; font-size: 11px; width:152; background-
Color:#ffffff; color:black;}
```

Php скрипт формы заказа или формы обратной связи с отправкой на почту будет иметь следующий вид:

```
<?

//сохраняем данные из пост в простые переменные, которые мы выбираем сами
if (isset($_POST['FIO'])) {$FIO = $_POST['FIO'];}
if (isset($_POST['money'])) {$money = $_POST['money'];}
if (isset($_POST['code'])) {$code = $_POST['code'];}
if (isset($_POST['telefon'])) {$telefon = $_POST['telefon'];}
if (isset($_POST['mail'])) {$mail = $_POST['mail'];}

if (isset($_POST['submit'])) { //проверяем была ли нажата кнопка "Заказать", если да, то идем дальше...
if (empty($FIO) or empty($telefon) or $money=="---") $ident1 = "* Введены не все поля со звездочкой (*)<br>";
// после сравнения проверяем, пускать ли пользователя дальше или, он сделал ошибку, и остановить скрипт
if ($code != 8) {$ident2 = "* Вы ввели неправильно защитную информацию<br>"; }
if (!isset($ident1) and !isset($ident2)) {$ident3 = "Заказ осуществлен. Благодарим за доверие к нам."; }
}

?>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org
/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Заказ продукции</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>

<body><table width="1000" border="0" align="center" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF" class="main_border">
<tr>
<td valign="top"><table width="100%" border="0" cellspacing="0" cellpadding="0">
```

```
<?

echo '
<H1 class="head">Заказ продукции</H1>
<p>У нас вы можете заказать яблоки, груши, ананасы, бананы, абрикосы. Для заказа заполните форму заказа.</p>
';
echo "<br>";
if (isset($ident1) or isset($ident2) ) { echo "<p class ='red'>Ошибка!!!<br>$ident1 $ident2"; } //если есть ошибка, то выводим ее
if (isset($ident3)) { //если ошибки нету, то отправляем заказ на почту mail@mail.ru.

if (!empty($mail)) $mail = "\nE-mail: $mail \n";
mail("mail@mail.ru", "Заказ продукции", "Здравствуй, Админ сайта! \nК вам поступил заказ от $FIO \nОплата производится через:  $money \nМобильный телефон: $telefon \n$mail");
echo "<p class='red'><strong>$ident3</strong></p>"; return; //выводим сообщение о завершении заказа и останавливает выполнение кода на этой странице
}

?>
```

```
<form action="zakaz.php" method="post" name="form_zakaz">
<table>
<tr>
<td>
<p class="form">Ф.И.О.*</p>
<td><input type=text name="FIO" size=50 value='<?= $FIO?>' maxlength=50
class='form2'>

<tr>
<td>
<p class="form">Оплата через*</p>
<td><select name="money" size="1">
<option value="---">---</option>
<option value="Веб-мани">Веб-мани</option>
<option value="Наличными">Наличными</option>
<option value="Виза">Виза</option>

</td></tr>

<TR>
<td>
<p class="form">Телефон мобильный*</p>
<td><input type=text name="telefon" size=50 value='<?= $telefon?>' maxlength=50
class='form2'>
```

```
<TR>
<td>
<p class="form">E-mail</p>
<td>
<input type=text name="mail" size=50 value='<?= $mail ?>' maxlength=50
class='form2'>
<TR>
<td>
<p class="form">Защитная информация: 5+3 = *:</p>
<td><input type=text name="code" size=50 value='' maxlength=50 class='form2'>

</td>
</tr>

</table>

<p><input type="submit" name="submit" value="Заказать" class="zakaz">
</FORM>
<p class="red">Звездочками (*) обозначены поля, обязательные для заполнения.</p>
<br>
</tr>
</table>

</td>
</tr>
</table>

</body>
</html>
```

Эту форму можно использовать как форму обратной связи, все что вам нужно - это изменить слово "Заказать" на "Оправить" и изменить содержимое полей.

Общее пояснение к коду формы: сначала происходит проверка формы кодом php. Мы проверяем была ли нажата кнопка "Заказать", если она была нажата, то мы проверяем все ли поля были заполнены или нет и правильно ли введена защита. Если все это проходит, то мы выводим сообщение, что заказ успешно закончен, после чего мы отправляем все данные на почту mail@mail.ru.

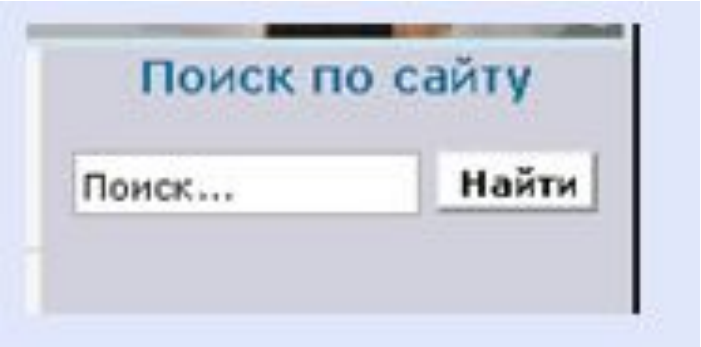
PHP поиск по базе данных

Рассмотрим поиск, который ищет в Базе Данных информацию и делает ссылки на них.

Вот скрипт самого поиска:

```
<!-- Поиск -->
<form action="search.php"
method="POST" onsubmit="javascript: if
((keyword.value==")||(keyword.value=='По
иск...')) { return false; } else { return true;
}">
<input name="keyword" onfocus="if
(this.value=='Поиск...') this.value=""
value="Поиск..." style="font: 11px Verdana;
height:14px; width:102px;" />
<input type="submit" name="send" value="
Найти" style="font: bold 11px Verdana;
background-color:#FFFFFF; height:19px;
width:50px;">
</form>
<!--Конец Поиск -->
```

Вот такой вид имеет форма поиска:



Эта **форма поиска защищает** от того, когда пользователь ничего не ввел.

Теперь в коде, который выше, мы видим следующе:<form action="**search.php**"

Здесь как раз search - это файл страницы куда будет передаваться введенная в поиске информация.

Поэтому вы можете там изменить на название вашей страницы, а если нет, то назовите ваш файл search.php и в него введите следующий код:

Скрипт php поиска по базе данных имеет:

- Защиту от взлома.
- Защиту от ошибок.
- Прост в установке на ваш сайт.

```
<?
echo "<p>По вашему запросу:";

if (isset($_POST['keyword'])) {$keyword = $_POST['keyword'];}

$keyword = trim($keyword);    \\ убираются лишние пробелы из начала и конца
строки
$keyword = stripslashes($keyword);    \\удаляет экранирование символов
$keyword = htmlspecialchars($keyword);    \\ заменяет html теги на коды
echo "<b>$keyword</b>";

$search_query = "SELECT id, title, view FROM table WHERE title LIKE
'%".strtoupper($keyword)."%' OR text LIKE '%".strtoupper($keyword)."%' LIMIT 100";
  \\ table - это ваша таблица, где надо искать
$query = mysql_query($search_query);    \\ Здесь непосредственно происходит поиск

if(!$query)
{
echo "<p class='text'>Поиск не осуществлен. Код ошибки:</p>";
echo exit(mysql_error());
}
if (mysql_num_rows($query) > 0)
{
$myrow = mysql_fetch_array($query);

do
{
printf("<p><a class ='post_link' href='articles.php?id=%s'>%s</a><span
class='post_view'> --> Просмотров: %s</span>
</p>",$myrow["id"],$myrow["title"],$myrow["view"]);    \\ здесь делаем ссылку где
находится данная информация.

}while ($myrow = mysql_fetch_array($query));
} else echo "<p>Ничего не найдено.";

?>
```

PHP загрузка картинок | Загрузка изображений php

Рассмотрим как сделать скрипт загрузки изображений на ваш сайт, при чем он еще будет иметь защиту от больших размеров и ограничение по количеству пикселей.

Для начала вам нужно вставить следующий код:

```
<p>Загрузаемый файл должен иметь ограничения: размер не превышает 1 Мб, пиксели по ширине не более 600, по высоте не более 5000.
<form name="upload" action="add_img.php" method="POST" ENCTYPE="multipart/form-data">
Выберите файл для загрузки: <input type="file" name="userfile">
<input type="submit" name="upload" value="Загрузить">
</form>
```

Это обычная форма загрузки изображения. Как видим то при нажатии на "Загрузить" мы перейден на страницу add_img.php, поэтому в этой странице вы вставим следующий текст:

```php
<?php
$uploaddir = 'imgarticles/';    // это папка, в которую будет загружаться картинка
$apend=date('YmdHis').rand(100,1000).'.jpg';    \\ это имя, которое будет присвоенно изображению
$uploadfile = "$uploaddir$apend";    \\ в переменную $uploadfile будет входить папка и имя изображения

if($_FILES['userfile']['size'] != 0 and $_FILES['userfile']['size']<=1024000) {    \\ Здесь мы проверяем размер если он более 1 МБ
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {    \\ Здесь идет процесс загрузки изображения
$size = getimagesize($uploadfile);    \\ с помощью этой функции мы можем получить размер пикселей изображения
if ($size[0] < 601 && $size[1]<5001) {    \\ если размер изображения не более 600 пикселей по ширине и не более 5000 по высоте
echo "Файл загружен. Путь к файлу: <br><b>http://ВашСайт.РУ/$uploadfile</b>";
}else {echo "Размер пикселей превышает допустимые нормы (ширина не более - 600 пикселей, высота не более 5000)";
unlink($uploadfile);    \\ удаление файла
}
} else {echo "Файл не загружен, верьнитель и попробуйте еще раз";}
}else { echo "Размер файла не должен превышать 1000Кб";}
?>
```

PHP редирект

Редирект - это очень полезная вещь для тех, кто не хочет ссылаться на другие сайты прямой ссылкой. Ведь не секрет, что когда человек ссылается прямой ссылкой, то его рейтинг понижается.

К вашему вниманию предлагаю скрипт редиректа на php:

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
Перенаправление
<?php
$url = isset($_REQUEST['url']) ? $_REQUEST['url'] : '';
if(preg_match('#(http?|ftp)://\S+[^\s.,>)\];\'\"!?]#i',$url)){
sleep(0);
echo "<html><head><meta http-equiv=\"refresh\"
content=\"0;url=$url\"></head></html>";
exit();
}
?>
```

Вы этот код сохраняете и называете go.php. Следовательно теперь ссылка на другой сайт будет выглядеть следующим образом:

```
<a href="go.php?url=http://zdos.ru/">Заработок в Интернете на добавлении статей</a>
```

Таким образом вы ссылаетесь на сайт ЗДОС, но через ссылку редирект.

База данных и функция date. Как лучше хранить формат дату

В MySql есть тип поля date, что лучше его использовать или функцию date в php.

Вопрос: У них формат разный в базе данных. Например: в базе данных 2000-10-7, а в php наоборот 7-10-2000. Или лучше хранить в бд количество секунд с 1970 года? Имеется в виду запись в БД формат даты?

Ответ: Язык php тесно связан с Базами Данных. Поэтому если мы в базе данных в таблице устанавливаем в поле date тип DATE, то при вызове функции:

```
$date = date("Y-m-d");
```

Данные так и запишутся: Год-месяц-день (2011-12-17).

```
$date = date("Y.m.d");
```

Данные так и запишутся: Год.месяц.день (2011.12.17).

```
$date = date("Y.m.d G:i");
```

В переменной будет: Год.месяц.день часы:минуты (2011.12.17 21:16).

Теперь если вы запишите эти данные в вашу таблицу Базы Данных, то именно этот формат и запишется:

```
$res = mysql_query("INSERT INTO table (date) VALUES ('$date')",$db);
```

Если же вам нужно выводить дату таким форматом: (13.12.2011 21:24). То нужно написать так:

```
$date = date("d.m.Y G:i");
```

Но учтите, что в таком формате данные в Базу Данных не запишутся. Они будут вставлены, то будет ошибка, и будет вставлена Дата по умолчанию.

Итак, мы видим, что тип date в базе данных может иметь разный формат. Но строгая последовательноть должна присутствовать, то есть: Год, Месяц, День, Час, Минуты ...

Если же вам хочется выводить результат даты в другом виде, то вообще лучше всего нужно сделать так: в таблице вы тип ставите не DATE, а CHAR и количество символов - 100. Тогда оно будет сохраняться как вы хотите. А извлечение данных происходит также.

Как определить количество элементов в массиве на php

Бывают моменты, когда важно знать сколько элементов в массиве.

Для этого существует функция count. Если переменная masiv имеет определенное число элементов, то узнать можно вот таким кодом:

count($masivl);

К примеру, вы используете резулярные выражения и не знаете сколько будет слов в тексте или вам надо поотсылать сообщения (сделать рассылку) и в форме присутствуют почтовые ящики.

В таких случаях нужно знать, сколько было введено почтовых ящиков и в цикле пройти по каждому почтовому ящику.

К примеру:

В переменной: $mail_to = "a@mail.ru, b@mail.ru, c@mail.ru";

И нужно на эти почты поотсылать сообщения... то:

```
$mail = split (",", $mail_to); // регулярное выражение
for($i=1; $i<=count($mail); $i++){
    echo "$i) ".$mail[$i]."<br>";
    if
(!preg_match("/^(?:[a-z0-9]+(?:[-_]?[a-z0-9]+)?@[a-z0-9]+(?:\.?[a-z0-9]+)?\.[a-z]{2
,5})$/i",trim($mail[$i]))) exit("Введите адрес в виде somebody@server.com");
    mail($mail[$i], $thm, $msg);
    echo "- Письмо отправлено!!!";
}
```

В переменно $thm указывается тема, а в переменно $msg указывается сообщение (текст письма)...

База Данный (БД) подключение

База данных — это представленная в объективной форме совокупность разных материалов (к примеру: статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных так, чтобы эти материалы могли быть найдены и обработаны с помощью ЭВМ.

База данных в веб программировании – это место хранения данных в таблицах.

Для того, чтобы подключиться к БД, необходимо знать название БД, пользователя БД и пароль к БД.

Ниже приведен код запроса подключение к базе данных.

```
$db = mysql_connect("localhost","name","4ad2fg6g");
//проверка имени и пароля и соединение с базой данных
mysql_select_db("baza",$db);
//выборка определенной базы данных
```

$db - это переменная, необходимая для запроса. Знак $ - применяется ТОЛЬКО для переменных.

mysql_connect - это функция подключения к базе данных. Она принимает три параметра. Путь к соединению, имя пользователя и пароль.

mysql_select_db - применяется для выборки с базы данных с определенной базы, потому что у вас может быть не одна база данных. Здесь вы выбираете конкретную базу данных.

http://myrusakov.ru/php-osnovy.html PHP
http://ab-w.net/PHP/begind.php
http://lezhenkin.ru/examples/php/easy-documnet/
http://htmlweb.ru/php/example/

ЛЕКЦИЯ 9. Применение библиотек для ускорения работы с AJAX-запрсами. Библиотека JQuery

Что такое jQuery?

jQuery - это библиотека, которая значительно упрощает и ускоряет написание JavaScript кода.
Девиз jQuery *"write less, do more"* (пиши меньше, делай больше) отражает ее главное предназначение.

jQuery позволяет создавать анимацию, обработчики событий, значительно облегчает выбор элементов в DOM и создание AJAX запросов.

Данная библиотека работает со всеми браузерами.

Для jQuery написано огромное количество плагинов, которые позволяют расширить ее возможности еще больше.

Добавление jQuery на страницы

Для того, чтобы начать использовать jQuery необходимо:

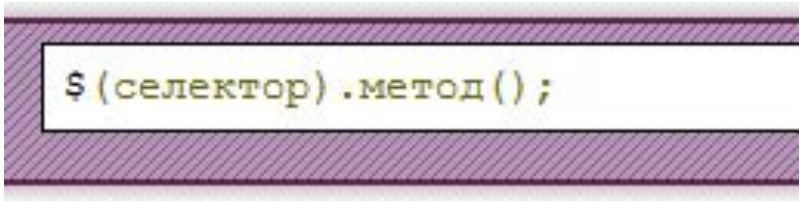
Скачать ее с официального сайта. Существуют две версии jQuery: для использования в готовых приложениях (production) и для разработки (development). Версия для разработки содержит комментарии и структурированный код. В сокращенной версии нет комментариев и код в ней не структурирован зато она занимает меньше места и поэтому страницы с ней будут загружаться быстрее. После того, как Вы выберите подходящую версию нажмите на кнопку "Download (jQuery)"

Добавить ее на страницу. Для этого следующий код должен быть добавлен на страницу в секцию head:

Команды jQuery

Код jQuery как и код JavaScript состоит из последовательно идущих **команд**. Команды являются основной структурной единицей jQuery.

Стандартный синтаксис jQuery команд:



Знак $ сообщает, что символы идущие после него являются jQuery кодом;
Селектор позволяет выбрать элемент на странице;
Метод задает действие, которое необходимо совершить над выбранным элементом. Методы в jQuery разделяются на следующие группы:

- Методы для манипулирования DOM;
- Методы для оформления элементов;
- Методы для создания AJAX запросов;
- Методы для создания эффектов;
- Методы для привязки обработчиков событий.

Обработчики событий jQuery

Обработчики событий - это функции, код которых исполняется только после совершения определенных действий.

Обработчики событий присутствовали и в JavaScript, но jQuery облегчает их использование и расширяет их функциональность.

Примеры действий, после которых выполняются обработчики:

- Курсор мыши наведен на элемент;
- Веб-страница или картинка полностью загружена;
- Изменено содержимое поля формы;
- HTML форма отправлена;
- Нажата клавиша на клавиатуре;

Общий вид определения обработчиков jQuery:

```
$(селектор).обработчик_события(function(){код_обработчика_события});
```

Эффекты jQuery

С помощью jQuery методов **fadeOut()**, **fadeIn()** и **fadeTo()** Вы можете постепенно скрывать и отображать элементы анимировано.

С помощью метода **animate()** Вы можете создавать на Ваших страницах полноценную анимацию.

Управление стилями в jQuery

jQuery имеет группу различных методов значительно упрощающих оформление элементов.

Одним из самых важных методов в этой группе является метод **css()**.

С помощью метода css Вы можете узнавать текущие или устанавливать новые значения свойств оформления элементов.

Создание AJAX запросов в jQuery

С помощью JavaScript Вы можете создавать асинхронные запросы и отправлять их на сервер.

Использование асинхронных запросов позволяет значительно ускорить загрузку страниц, так как в этом случае обновляться будет только та часть страницы, которая содержит новые данные, а не страница целиком.

Техника использования асинхронных запросов называется **AJAX** - **A**synchronous **J**avaScript **A**nd **X**ML (Асинхронный JavaScript и XML).

Создание AJAX запросов на "чистом" JavaScript имеет несколько недостатков:

Код даже самого простого AJAX запроса получается достаточно громоздким и сложным для понимания без специального ознакомления.

Необходимо добавлять дополнительный код для поддержки старых версий браузеров.

AJAX запрос созданный с помощью jQuery занимает всего одну строчку кода, и уже оптимизирован для использования и с новыми и старыми версиями браузеров.

Синтаксис:

```
$("селектор").load(url,данные,функция)
```

Плагины jQuery

Реализация одних и тех же функций в различных приложениях побуждает разработчиков заново писать один и тот же код несколько раз лишь незначительно изменяя его под конкретное приложение.

Плагины jQuery позволяют забыть разработчикам о данной проблеме. Разработчик может один раз написать плагин, который позволяет реализовать определенную функцию и затем использовать его в необходимых приложениях написав только одну строчку кода.

В интернете можно найти огромное количество бесплатных jQuery плагинов написанных другими разработчиками и использовать их для создания своих приложений. Можете начать поиски интересных jQuery плагинов на сайте PluginDetector.com.

Создание плагина

Для того, чтобы создать плагин необходимо добавить к объекту jQuery.fn свойство, имя которого будет является именем плагина:

Синтаксис:

```
//Определяем код плагина
(function($){
   $.fn.имяПлагина = function() {
      // Код плагина
   };
})(jQuery);
//Вызываем плагин
$(селектор).имяПлагина();
```

Что такое AJAX?

AJAX расшифровывается **A**synchronous **J**avaScript **A**nd **X**ML (Асинхронный JavaScript и XML).

AJAX - это не новый язык программирования или разметки. AJAX - это эффективный способ совместного использования HTML, CSS, JavaScript и DOM.

С помощью использования AJAX Вы можете заметно увеличить скорость реакции интерфейса и значительно уменьшить нагрузку на сервер. Это становится возможным благодаря асинхронному обмену информацией и способностью перезагружать только "обновленную" часть страницы без необходимости перезагрузки страницы целиком.

AJAX используется многими известными веб-приложениями такими как: Facebook, Flickr, Gmail, Google Maps и Youtube.

Какие технологии включает AJAX?

AJAX - это набор технологий, которые поддерживаются веб-браузерами. AJAX использует:

- **HTML** в качестве "каркаса";
- **CSS** для оформления;
- **DOM** для извлечения или изменения информации на странице;
- **Объект XMLHttpRequest** для асинхронного обмена данными с сервером;
- **JavaScript** для связи перечисленных выше технологий между собой.

Этапы выполнения AJAX запроса

Система управления содержимым

Систе́ма управле́ния содержи́мым (конте́нтом) (англ. *Content management system*, **CMS**) — информационная система) — информационная система или компьютерная программа) — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления контентом (то есть содержимым).

Основные функции CMS:

- Предоставление инструментов для создания содержимого, организация совместной работы над содержимым,
- Управление содержимым: хранение, контроль версий, соблюдение режима доступа, управление потоком документов и т. п.,
- Публикация содержимого,
- Представление информации в виде, удобном для навигации, поиска.

В системе управления содержимым могут находиться самые различные данные: документы В системе управления содержимым могут находиться самые различные данные: документы, фильмы В системе управления содержимым могут находиться самые различные данные: документы, фильмы, фотографии В системе управления содержимым могут находиться самые различные данные: документы, фильмы, фотографии, номера телефонов, научные данные и так далее. Такая система часто используется для хранения, управления, пересмотра и публикации

Классификация CMS

За несколько лет системы управления содержимым веб-ресурсов значительно усовершенствовались. Классифицируем CMS по областям применения:

- Порталы. Используются для информационных ресурсов, основной целью ставят максимальное упрощение публикации статей и новостей.

-Движки без SQL. Это ответвление в разработке CMS развито относительно слабо, так как использование в качестве хранилища информации файлов вместо таблиц базы данных сопряжено с множеством труднорешаемых проблем. Достоинство этих CMS – в доступности для модификации контента и возможности размещения на бесплатных хостингах.

-Блог. Происходит от англ. weblog. Русский термин – «сетевой дневник» – это сайт, на котором находятся личные заметки автора. В основном заметками являются ссылки на сайты, которые кажутся владельцу ресурса наиболее интересными, и комментарии к ним. Блог может содержать не только ссылки, но и просто электронный дневник пользователя.

- Форумы – это инструмент для общения на сайте. Сообщения в форуме в чем-то похожи на почтовые – каждое из них имеет автора, тему и содержание. Но для того, чтобы отправить сообщение в форум, не нужна никакая дополнительная программа – нужно просто заполнить соответствующую форму на сайте.

- Магазины. К магазинам отнесем любой сайт, с которого можно заказать какой-либо товар. В данном случае в определение «товара» может входить абсолютно все, включая время доступа в Интернет, минуты сотовой связи.

- Групповая работа (Groupware) – комплекс программного обеспечения, позволяющий организовать работу предприятия, отношения с клиентами и заказчиками в Интернете. Обычно представляет собой полностью или частично закрытую часть сайта с возможностью отслеживать сроки выполнения поставленных задач, распределение ролей и временных нормативов. Иногда можно выносить вопросы на обсуждения и решения вышестоящего руководства.

- Обучение (e-Learning) – дистанционная форма обучения с использованием Интернета. Онлайновая форма обучения уже не один год является «маяком», на который ориентируются образовательные системы разных стран мира. Главным стратегическим направлением является быстрое обновление знаний и эффективное использование информации.

- Базы знаний (KnowledgeBase) позволяют накапливать опыт множества разработчиков. Каждая такая база знаний имеет свою специфичную структуру, поэтому никаких общих решений на данный момент не предложено.

- Биллинг (Billing). Программное обеспечение, позволяющее провайдерам и реселлерам работать со счетами клиентов. Такие CMS являются неотъемлемой частью крупной системы учета потребления услуг пользователями. Задача же CMS данной категории – в отображении информации о предоставленных услугах, подключении новых услуг, изменении текущих параметров, приеме платежей и т.п. Часто такие системы пишутся своими силами.

Система управления содержимым

Достаточно часто планируя создание веб-ресурса, веб-мастера теряются в догадках, какую систему управлением содержимым сайта выбрать? В сети Интернет существует множество предложений и мнений. Есть ярые сторонники одних CMS и не менее ярые противники их. Мнения разнятся, и такое положение дел только осложняет задачи для веб-мастера. Зная некоторые нюансы, каждый может быстро понять, какую систему ему использовать и почему он склонился именно к ней. Для этого необходимо иметь четкое представление о создаваемом веб-ресурсе и о дальнейшем его развитии. Такой продуманный подход позволит сделать правильный выбор в целесообразности применения той или иной системы управления содержимым (CMS) для будущего сайта.

Некоторые CMS такие, как Joomla, Wordpress, Drupal и Dle достаточно распространены в сети Интернет. Каждая из них обладает своими достоинствами и недостатками. Выбор той или иной системы значительным образом зависит от будущей тематики сайта, например, новостные сайты, файловые обменники, сайты с мультимедийным содержимым, блоги, форумы и т.д. Целесообразность использования CMS складывается из того, что если планируется создание большого проекта с постоянно обновляющейся информацией или контентом в целом, то CMS для такого сайта необходима. Использование системы управления содержимым на сайтах - "Визитках", где в наличие имеется всего 5-10 страниц с информацией, не оправдано. Ведь система управления содержимым - это не один маленький скрипт, а десятки дополнительных страниц содержащих PHP, Java script скрипты. Они создают нагрузку на сервер, что не очень хорошо и оправдано для крошечного веб-ресурса. В этом случае, лучше отказаться от использования системы управления.

Wordpress (CMS). Это достаточно неплохой движок сайта. Универсальность и бесплатность этой системы управления содержимым, делают ее популярной и востребованной среди веб-мастеров. Изюминка этой системы заключается в том, что существует большое количество плагинов расширения, русскоязычных тем оформления и все это абсолютно бесплатно. Простота в установке и работе с этой CMS удивляет. Даже неопытный пользователь, имеющий только общие представления о сайтостроении, сможет без труда работать и настраивать эту систему. Установка занимает не более 5 минут, а дальше все зависит от широты фантазии веб-мастера. Wordpress - это постоянно развивающийся программный продукт. В отличие от большинства, когда-либо существующих CMS, Wordpress продолжает свою жизнь и развитие. Постоянно выходят новые версии этого движка. Целесообразно использовать Wordpress для небольших проектов. В этом случае он покажет себя только с лучшей стороны. Создание блога под управлением Wordpress будет правильным и оправданным решением для любого веб-мастера. Создание обычного сайта, тоже не противоречит возможностям этой системы, но все же при всех радужных характеристиках, Wordpress является блоговым движком и лучше эту систему использовать по назначению. Хотя веб-мастер вправе принимать индивидуальное решение в области применения. Официальный сайт Wordpress.

Joomla (CMS). Очень знаменитая система управления содержимым (CMS). Пригодна для управления большими проектами с любым содержимым. Этот программный продукт постоянно обновляется и развивается. На официальном сайте этого движка можно найти тысячи различных плагинов - расширения. Шаблонов в сети Интернет огромная масса. Эта CMS не обделена функционалом. Сайт под управлением этой системы получится современным и продвинутым в плане различных наворотов. В установке на сервер - Joomla практически не отличается от установки Wordpress. Он бесплатный. Однако, с выходом новой версии движка, разработчики не позаботились даже о совместимости плагинов расширения с предыдущей версией, а значит, люди использующие старую версию остались без поддержки. Этот факт говорит о безалаберности разработчиков и о том, что их мало волнуют пользователи их продукта. При всей красочности и функциональности, например, "Административной панели", в ней присутствует огромное количество бесполезных и ненужных функций. Joomla, достаточно серьезно нагружает сервер, что зачастую приводит к недоступности ресурса для пользователей. Этот факт влечет за собой потерю посетителей, а если это Интернет-магазин, то к снижению продаж.

Drupal (CMS). Достаточно мощный инструмент управления содержимым сайта. Поддерживает работу, как с малыми, так и с большими проектами. Основной аудиторией использующей эту CMS, являются веб-разработчики имеющие опыт в программировании. Ведь, как бывает, люди имеющие желание создать свой сайт, не имеют никакого опыта в программирование. Drupal, потребует этих знаний. Значит - эта система в большей степени предназначена для профессионалов, чем для любителей. Конечно, если есть желание, то каждый может попробовать работу этой CMS.

DLE (DataLife Engine) (CMS). Универсальная система управления контентом. Изначально была представлена, как новостной движок, но благодаря гибким настройкам применима абсолютно для любого сайта. В сети Интернет почему-то ассоциируется с сайтами, распространяющими пиратские программы, в народе - "Варезники". Эта система имеет большое количество различных шаблонов и модулей-расширения. Отличается от других надежностью, простотой, функциональностью и выверенным набором настроек. В ней присутствует все, что необходимо для управления и ничего лишнего. Работает эта CMS быстро и не создает никакой нагрузки на сервер. На сегодняшний день является одной из самых лучший систем управления содержимым. Производитель поддерживает и развивает ее на высоком уровне, что позволяет этой CMS быть несомненным лидером среди ей подобных. DLE невозможно поставить в один ряд с Wordpress и Joomla, они отличаются, как небо и земля. DataLife Engine предназначена для солидных, профессиональных веб-ресурсов.