

## ОГЛАВЛЕНИЕ

Обзор Web-технологий.....	3
HTML (HyperText Markup Language).....	4
Dynamic HTML (DHTML — Dynamic HiperText Markup Language). ....	4
Языки сценариев (JavaScript и VBScript).....	4
Технологии Java и CGI .....	5
ASP и PHP .....	5
Dreamweaver .....	6
Модели организации сайта.....	7
Линейная организация.....	7
Решетка.....	8
Иерархия .....	9
Модель паутины .....	12
Основы разметки гипертекста HTML .....	12
Списки .....	14
Гиперссылки .....	15
Таблицы.....	16
Изображения.....	18
Фреймы.....	20
Формы .....	22
Введение в JavaScript .....	24
Язык сценариев JavaScript.....	24
Обработчик событий.....	25
Сценарии в HTML-документе .....	26
Функции и объекты пользователя .....	31
Введение в PHP .....	33
Управляющие конструкции PHP .....	35
Работа с MySQL .....	36
Практикум .....	39
Практическая работа № 1. Основы разметки гипертекста HTML .....	39
Практическая работа № 2. Разработка сценариев Web - страниц.....	40
Практическая работа №3. Динамическое изменение Web - страниц .....	44
Практическая работа №4. Работа с мышью и клавиатурой.....	49
Практическая работа №5. Дизайн сайта .....	51

Практическая работа №6. Основы программирования на языке PHP.....	51
Практическая работа №7. Работа с MySQL через PHP .....	52
Практическая работа № 8. Установка и настройка сервера IIS.....	52
Практическая работа № 9. Разработка простейшего приложения Web Forms ASP.NET.....	58
Практическая работа № 10. Исследование структуры приложения Web Forms ASP.NET.....	63
Практическая работа № 11. Разработка и применение HTTP-обработчика ASP.NET .....	68
Практическая работа № 12. Применение серверных HTML-элементов управления ASP.NET .....	72
Практическая работа № 13. Применение серверных базовых WEB-элементов управления ASP.NET .....	75
Практическая работа № 14. Применение серверных полнофункциональных элементов управления ASP.NET.....	78
Практическая работа № 15. Применение серверных элементов управления проверкой достоверности ASP.NET .....	82
Практическая работа № 16. Применение серверных элементов управления AJAX ASP.NET.....	85
Практическая работа № 17. Кэширование страниц ASP.NET.....	87
Лабораторный практикум.....	90
Лабораторная работа №1. Web-сервер Apache .....	90
Лабораторная работа №2. Статический html-документ .....	96
Лабораторная работа №3. Каскадные таблицы стилей CSS.....	108
Лабораторная работа №4. Динамический html-документ.....	113
Лабораторная работа №5. CGI-скрипт.....	116
Лабораторная работа №6. PHP-скрипт. ....	121
Лабораторная работа №7. Графическая библиотека PHP GD .....	124
Лабораторная работа №8. Технология AJAX.....	125
Задание на курсовое проектирование .....	128
Задание на самостоятельную работу студента.....	129
Темы для выполнения СРС и курсового проектирования .....	130
БИБЛИОГРАФИЧЕСКИЙ СПИСОК .....	132

## ОБЗОР WEB-ТЕХНОЛОГИЙ

Web-приложения начали свое развитие с Web-узлов и Web-систем. Первые Web-узлы, созданные Тимом Бернерсом-Ли для Европейской лаборатории физики частиц CERN, составляли распределенную систему гипермедиа, позволяющую исследователям получать прямой доступ со своих компьютеров к документам и информации других исследователей. Доступ к документам осуществлялся с помощью специальных программ - браузеров, работающих на клиентских компьютерах. С помощью такой программы пользователь может запрашивать документы Web с других компьютеров сети и отображать их на экране своего компьютера. Для просмотра документа необходимо запустить браузер, а затем ввести имя документа и имя узлового компьютера, на котором он находится. Браузер отправляет этому узлу запрос на документ, который обрабатывается программным приложением, получившим название Web-сервера. Web-сервер получает запрос, находит документ в своей файловой системе и отправляет его обратно браузеру.

Web-система - это система гипермедиа, поскольку ее ресурсы связаны между собой. Термин Web означает, что система рассматривается как набор узлов со ссылками друг на друга. Web-приложение строится на основе Web-системы или расширяет ее, добавляя к ней бизнес-логику и новую функциональность. Упрощенно можно считать, что Web-приложение - это Web-система, позволяющая пользователям реализовывать бизнес-логику через браузер.

Web-приложения обеспечивают возможность динамического изменения содержания Web-страниц и позволяют пользователям изменять бизнес-логику приложения на сервере. Различия между Web-узлом и Web-приложением весьма условны и сводятся к возможности пользователя влиять на бизнес-логику системы. Если на сервере никакая бизнес-логика не предусмотрена, то такую систему нельзя назвать Web-приложением. Если же на бизнес-логику приложения можно влиять через Web-браузер, то система относится к числу Web-приложений. Практически во всех Web-приложениях, за исключением самых наипростейших, пользователю приходится не просто просматривать информацию. Он вводит различные данные, представляющие простой текст, информацию об именах файлов или сведения о выбранных управляющих элементах.

Для разработки Web-приложений используются различные технологии, обеспечивающие механизм создания динамических Web-страниц, которые способны реагировать на введенную пользователем информацию. Существует несколько подходов к созданию Web-приложений. Самые первые из них сводятся к выполнению на Web-сервере отдельных модулей. Вместо запроса на Web-страницу в формате HTML браузер отправляет запрос, интерпретируемый Web-сервером как запрос на загрузку и выполнения некоторого модуля. Результатом выполнения модуля может

быть страница в формате HTML, изображение, аудио-, видеоинформация или другие данные.

### **HTML (HyperText Markup Language)**

HTML — язык разметки гипертекста — является приложением языка SGML (Standard Generalized Markup Language) — стандартный обобщенный язык разметки. Средствами HTML задаются синтаксис и размещение специальных встроенных указаний, в соответствии с которыми браузер отображает содержимое документа (текст, графика, мультимедиа, гиперссылки). Говоря другими словами, HTML - язык компоновки документов и спецификации гиперссылок, используемый для кодировки документов в WWW.

С течением времени HTML обогатился средствами динамической интерпретации, или

### **Dynamic HTML (DHTML — Dynamic HyperText Markup Language).**

#### **DHTML (Dynamic HyperText Markup Language)**

До недавнего времени информация в большинстве Web-документов была статической, что требовало реакции сервера на действия пользователя. С введением DHTML парадигма Web сместилась от взаимодействия с сервером в сторону создания интерактивных Web-узлов и Web-приложений. Основной отличительной особенностью DHTML от HTML является возможность взаимодействия DHTML-документов с пользователем на клиентском компьютере, что в значительной степени обогащает возможности создаваемых с их помощью Web-страниц и Web-приложений и в то же время сводит часть взаимодействия пользователя с сервером к взаимодействию пользователя с DHTML-документом. Таким образом, можно говорить о перенесении некоторой доли вычислений с серверной на клиентскую сторону, что, разумеется, сокращает объем передаваемой информации от клиента серверу и обратно и экономит время. Как следствие, страницы, разработанные с использованием модели DHTML, в отличие от HTML, работают значительно быстрее именно за счет снижения объема информации, передаваемой от клиента (браузера) серверу и обратно.

### **Языки сценариев (JavaScript и VBScript)**

Введя понятие DHTML, необходимо поговорить о языках создания сценариев событий HTML-документов. Для начала давайте определим понятие "сценарий". Итак, сценарий, в отличие от программы, имеет лишь одно направление выполнения — сверху вниз, именно поэтому, говоря о сценариях, встроенных в HTML-документы, не следует забывать о строгом порядке, в котором браузер формирует содержимое страницы.

Компания Netscape разработала язык создания сценариев JavaScript, а компания Microsoft — VBScript. Эти языки используются на стороне клиента, то есть генерируют объекты на основании HTML-страницы на стороне клиента в окне его браузера.

## **Технологии Java и CGI**

В 1994 году специалистами компании Sun была разработана технология создания динамических интерактивных Web-страниц – Java. Программы на языке Java называются апплетами (little applications). Апплеты пишутся на Java и посылаются по Web как HTML- файлы браузеру, где выполняются как HTML-документы. Существенным преимуществом Java является независимость программ от платформ, на которых программы выполняются. Хотя Java не обязательно выполняется в окне браузера, возможно создание независимых (stand-alone) Java-приложений, которые могут выполняться на компьютере независимо от Интернета.

Фактически программа на языке Java транслируется компилятором в специальный код, называемый байтовым (bytecode), а затем выполняется уже с помощью интерпретатора языка Java. Такое "разделение обязанностей" и позволяет обеспечивать полную независимость Java-кода от конечной платформы, на которой он будет выполняться. Разумеется, для каждой конкретной платформы имеется свой интерпретатор языка, называемый виртуальной машиной Java (Java Virtual Machine).

Схема исполнения апплетов коренным образом отличается от схемы выполнения CGI- скриптов. Последние, в частности, выполняются на стороне сервера, в отличие от Java- апплетов, которые выполняются, как правило, на стороне клиента.

Что же такое CGI (Common Gateway Interface), или интерфейс общего шлюза?

По сути CGI - способ взаимодействия Web-программ с браузером пользователя. Поэтому под CGI-программами понимают программы, написанные на любом языке программирования, способного выполняться на Web-сервере, включая C, C++, Visual Basic или даже командные языки операционных сред (например, C Shell). Но большинство

CGI-программ пишется на языке Perl. Perl (Practical Extraction and Report Language)

является одним из наиболее гибких языковых средств, служащих для программирования интерфейсов CGI. Изначально Perl предназначался для обработки больших объемов данных и генерации отчетов по обработке этих данных (как явствует из его названия). За последние несколько лет Perl превратился в полнофункциональный язык программирования. Изначально созданный исключительно для работы под управлением операционных систем семейства UNIX, Perl теперь совместим с такими ОС, как Amiga, MS-DOS, OS/2 Warp, VMS, Windows NT, Window 95 и Macintosh.

## **ASP и PHP**

В последнее время все большую популярность получают эти два средства создания интерактивных Web-страниц. Основным их достоинством является возможность формирования страниц на основании интерактива "клиент-сервер". Сами же программы, написанные на ASP (Active Server Pages – активные серверные страницы) и PHP (Personal Home Page),

настолько просты, что программирование с их помощью доступно даже неискушенным.

PHP часто еще называют препроцессором гипертекста (Hypertext Preprocessor). По сути PHP серверный (выполняющийся на стороне сервера) мультиплатформный язык описания сценариев, встраиваемый непосредственно в HTML-код. В настоящее время PHP интенсивно используют более полумиллиона доменов Всемирной компьютерной сети. Основу синтаксиса PHP составляют язык программирования C, Java и Perl. Целью создания языка является разработка динамически генерируемых страниц в кратчайшие сроки.

Например, если Вы создаете online-каталог, вам скорее всего понадобится разрабатывать не сами HTML-страницы, а их шаблоны, по которым PHP будет формировать HTML-страницы исходя из ваших потребностей. Традиционно этот колоссальный объем работы выполняется вручную. С помощью PHP гораздо проще организовать интерфейс к базе данных и динамическое формирование страниц.

Сравнивая PHP и ASP, которые решают, по сути, схожие задачи, следует отметить переносимость первого (PHP) в отличие от второго (ASP) и специальную "заточку" ASP под создание гибких и удобных интерфейсов к базам данных. Это включает использование ActiveX Data Objects (ADO). Колоссальная поддержка структурированного языка запросов к базам данных SQL является мощнейшим средством, используя которое разработчик может не переучиваясь, работать напрямую с базами данных привычным образом. ASP поддерживает работу со всеми базами данных, соответствующими стандарту ODBC.

Говоря простыми словами, Active Server Pages - это обычные страницы, которые содержат скрипты, выполняющиеся на сервере наряду с обычным HTML-кодом.

ASP становится совместимым со все большим числом операционных систем.

ASP нужен за тем, что активные серверные страницы (Active Server Pages) и HTML взаимодействуют с базами данных совершенно по-разному.

### **Dreamweaver**

В 1997 году появилась программа Dreamweaver 1.0 Macromedia, которая позволяла вручную не писать HTML-код. Этот визуальный редактор не является первым, но, несомненно, является лучшим благодаря своим функциям и возможностям, а именно понятному и гибкому интерфейсу. Этот редактор позволяет создавать Web-страницы как встроенными средствами, библиотеками, шаблонами, так и дает возможность работать непосредственно с кодом HTML-страниц. Несмотря на удобство и популярность Dreamweaver как средства создания Web-сайтов, невозможно создать хороший и практичный сайт, не зная основ, в частности HTML-кода, языков сценария JavaScript и VBScript. Поэтому в данных методических указаниях будут приведены основы разработки Web-сайтов, которые

необходимо знать и которые можно и нужно применять, даже работая в визуальном редакторе Drwemweaver, используя автоматизированные, встроенные средства редактора.

### **Модели организации сайта**

Существуют четыре основные логические организационные формы, используемые web-сайтами:

1. Линейная
2. Решетка
3. Иерархия
4. Паутина

Прежде, чем начать создавать структуру сайта, необходимо определить тип сайта и на него накладывать структуру. Существует логическая и физическая структура. Логическая структура описывает документы, которые связаны с другими документами и определяет связи между ними. Физическая структура описывает, где документ находится в действительности.

Причем надо помнить, что для пользователя важна логическая структура, а не физическая. Поэтому не раскрывайте физическую файловую структуру сайта, когда это возможно. Скрывая реальные пути, вы тем самым вольны изменять расположение файлов по своему усмотрению.

Логическая структура документа сайта не должна полностью соответствовать физической структуре.

Линейная форма является наиболее популярной из всех структур по причине того, что традиционные печатные информационные средства следуют этому стилю организации.

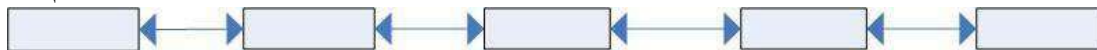
Линейную форму могут слегка модифицировать, но при слишком большом расширении она превращается в решетку, иерархию или паутину.

#### **Линейная организация**

Линейная организация делится на:

*Строго линейная организация.*

Она способствует упорядоченному продвижению по основной части информации

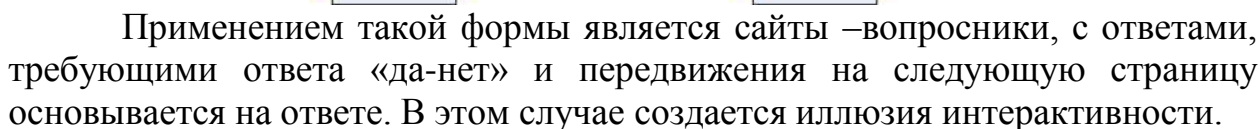


Такая форма хороша для презентаций. Для такой формы организации в силу предсказуемости событий возможно осуществить предварительную загрузку (preload) или предварительная выборка (prefetch) следующего блока информации, что поможет улучшить восприятие информации. Например, пока пользователь просматривает информацию одного экрана, изображение следующего экрана загружается в кэш браузера.

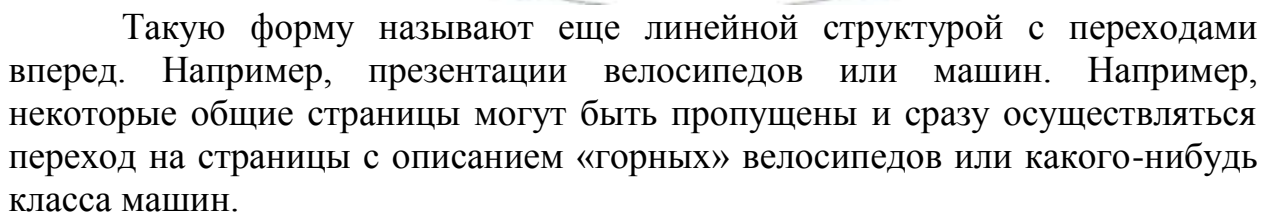
*Линейная форма с альтернативами*

Предыдущая форма дает мало выбора пользователю.

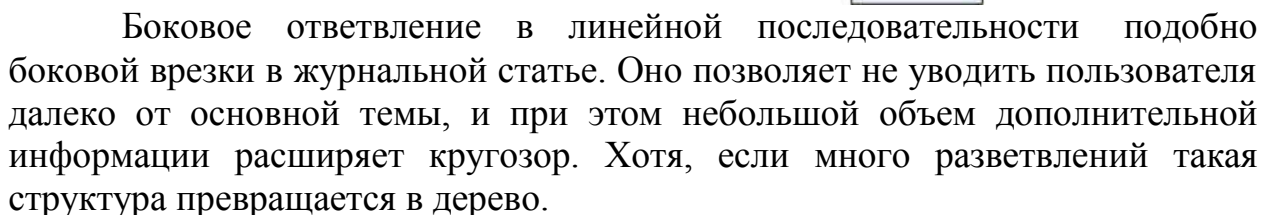
Линейная форма организации с альтернативами имитирует интерактивность, предоставляя два или более вариантов перехода со



Такая форма хорошо работает, когда необходимо сохранить общее направление, но при этом нужно добавить лечение вариации, такие как пропуск страниц.

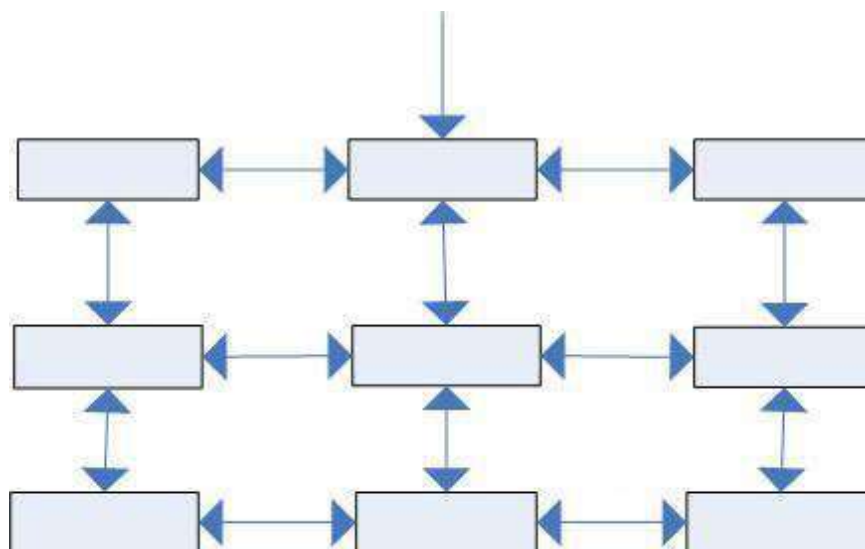


Такая форма позволяет контролировать отклонения от основного направления, но такая структура все же принуждает его вернуться к основному пути, сохраняя первоначальное движение.



Решетка – это двунаправленная структура, в которой присутствуют как горизонтальные, так и вертикальные связи между элементами.





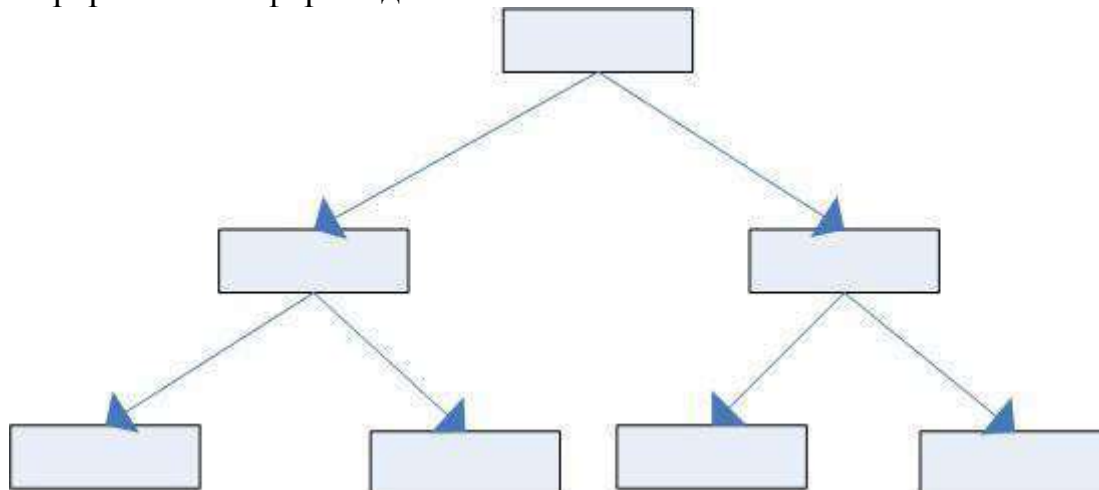
Правильно разработанная решетка имеет горизонтальные и вертикальные ориентиры, поэтому пользователь не чувствует себя заблудившимся внутри сайта. Например, предметы в каталоге могут быть собраны по категориям (рубашки, брюки, т.д.) Другой способ организации по ценам. Структура в виде решетки позволяет пользователю ориентироваться как по категориям, так и по ценам.

Каталоги – это наиболее частое использование решетки.

### **Иерархия**

Иерархия всегда начинается с корневой страницы или домашней страницы и эта страница отличается ото всех остальных.

Иерархические формы делятся на:

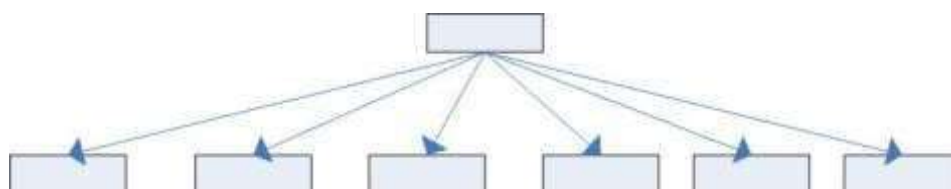


#### *Узкие деревья*

Использование узкой иерархии в качестве средства последовательного продвижения по сайту может помочь удерживать пользователя на правильном направлении.

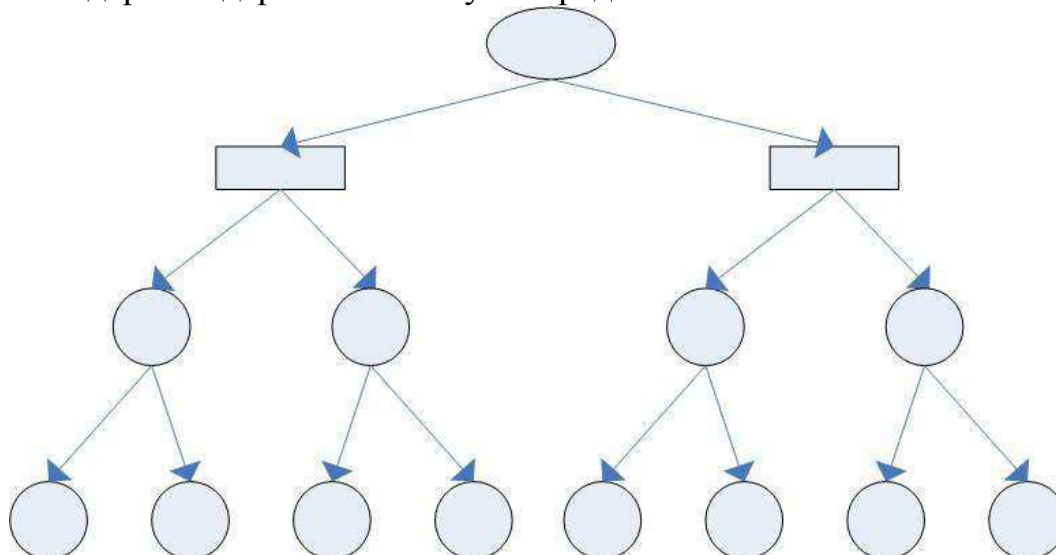
#### *Широкие деревья*

Основываются на большом количестве вариантов выбора. Его недостаток –слишком много вариантов в виде страниц.



*Запутанные деревья*

Стандартное дерево используется редко.



С домашней страницы перепрыгиваем в разделы, но в разделах существуют обратные и перекрёстные ссылки, которые усложняют структуру сайта. Страницы в таких случаях связаны перекрестными ссылками при помощи панели навигации или явных обратных ссылок, позволяющим быстро перемещаться по структуре сайта.

#### *Полное связывание*

В этом сайте каждая страница связана ссылкой с каждой страницей этого сайта. Количество ссылок = (количество страниц)\*(количество страниц – 1)

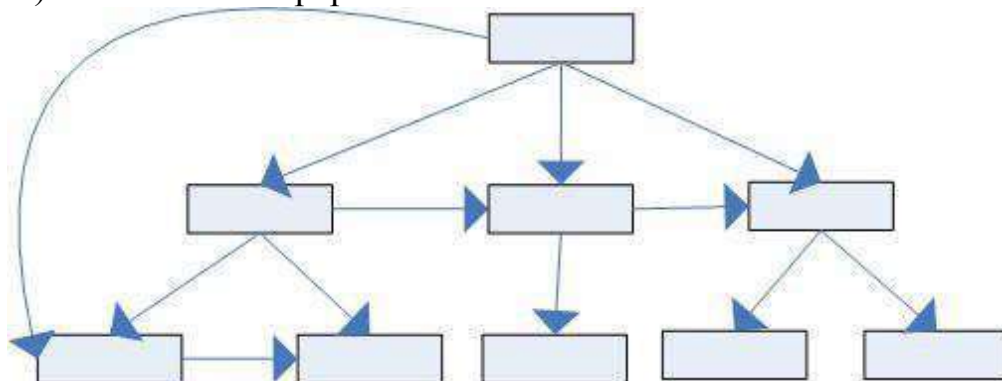
Такой сайт не является лучшим выбором. Большинство сайтов склонны использовать частичное связывание.



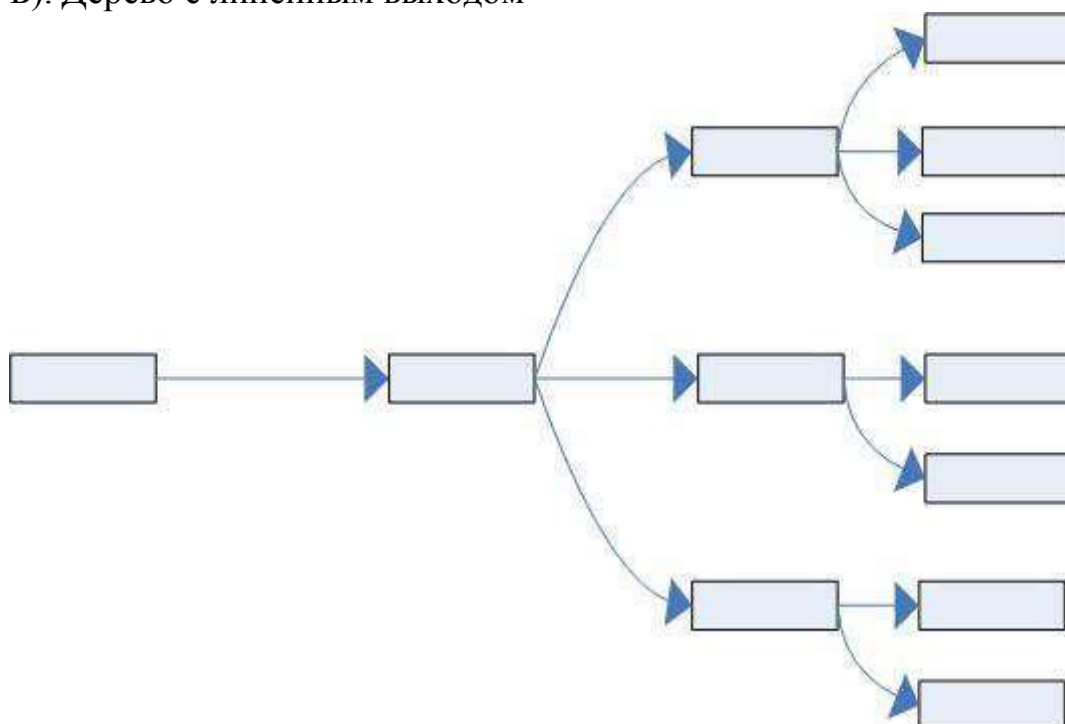
### *Смешанные формы или смешанная иерархия*

Это наиболее часто встречающаяся форма. Внутри формы содержится линейные участки, пропуски и даже решетки. Примеры таких форм:

#### А). Смешанная иерархия



#### Б). Дерево с линейным выходом



### *Модель "ступица и спицы"*

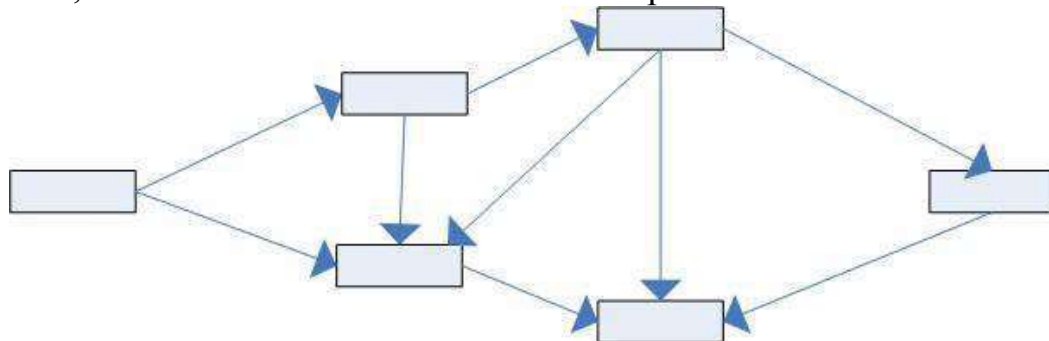
Многие сайты состоят из главных страниц, называемых ступицами и подчиненных страниц, доступ к которым осуществляется через спицы. Многие порталы используют этот стиль для поощрения повторных посещений страниц.



Одно из преимуществ центра и спицы состоит в том, что модель может обеспечивать простой способ осмысления сайта: центральные разделы (центр) со спицами родственного содержания, которые пользователь вкратце просматривает перед возвратом в центр.

### **Модель паутины**

Если совокупность документов выглядит так, будто имеет различные структуры, то она называется паутиной. Такая структура сложна для понимания, хотя она обеспечивает большие выразительные возможности.



## **Основы разметки гипертекста HTML**

Web-страницы пишутся с использованием языка HTML и обычно передаются провайдеру услуг Интернет ISP (Internet Service Provider) для размещения на Web-сервере. Для получения страниц с Web-сервера используется протокол HTTP (Hyper Text Transfer Protocol). Как известно, адрес Web-страницы называется универсальным локатором ресурсов URL (Universal Resource Locator), например, <http://www.microsoft.com>. Web-страницы загружаются в компьютер и просматриваются на экране с помощью специальных программ, называемых браузерами, таких, как Internet Explorer, Netscape Navigator и т.п. Каждый браузер имеет свой набор возможностей для просмотра Web-страниц. Сравнительный анализ показывает, что, например, Internet Explorer фирмы Microsoft обладает

большими возможностями, чем браузер Navigator фирмы Netscape. Internet Explorer поддерживает два языка реализации сценариев: VBScript и JavaScript, в то время как Netscape Navigator – только JavaScript.

При создании Web–страниц используют различные инструментальные средства. Обычно статическая часть Web–страницы создается текстовым и графическим редакторами, обеспечивающими сохранение данных в формате HTML, а динамическая часть программируется на языках VBScript и JavaScript или с использованием специальных средств языков программирования таких, как Delphi или Visual C++.

Язык разметки гипертекста HTML определяет структуру и динамику Web–страницы с помощью специальных символов, называемых тэгами. Тэг–это символ или ключевое слово, взятое в угловые скобки. Тэги, как правило, встречаются парами: начальный и конечный. Конечный тэг отличается от начального только наличием косой черты перед символом или словом. За символом или словом часто идет список параметров, в котором параметры отделяются друг от друга пробелами. Часто параметры имеют значения, задаваемые с помощью знака равенства.

Любая Web–страница должна начинаться тэгом `<html>` и заканчиваться тэгом `</html>`.

Программист может добавлять для себя некоторые примечания, но не показывать их пользователю. Достигается это следующим образом: `<!--` Текст комментария из одной или нескольких строк `-->` Комментарий можно включить до начала страницы, в любое место страницы или в конец страницы.

Web–страница обычно начинается с заголовка, который устанавливается тэгами `<head>` и `</head>`. Раньше заголовок включал формальную структуру с информацией об остальной части страницы. Теперь заголовок используется в сокращенном варианте и, как правило, содержит только заглавие Web–страницы выводимые тэгами `<title>` и `</title>`. Заглавие появляется в строке заголовка Web–браузера. Этот текст используется также поисковыми системами Интернет.

Основное содержание страницы называется телом, и вводится тэгами `<body>` и `</body>`. В тэге `<body>` можно использовать ряд ключевых строк, называемых атрибутами, для установки таких параметров, как цвет гиперссылок и т.д. Основными атрибутами тэга `<body>` являются: `align`, `alink`, `background`, `bgcolor`, `bgproperties`, `bottommargin`, `class`, `id`, `lang`, `language`, `leftmargin`, `link`, `rightmargin`, `scroll`, `style`, `text`, `onblur`, `onfocus`, `onload`, `onload`, `vlink`. Цвет устанавливается путем указания значения красной, зеленой и синей составляющих в шестнадцатеричном формате либо с помощью цветовой константы HTML. Например, для зеленого цвета можно использовать шестнадцатеричное значение `00FF00` или константу `green`. Цвет фона в тэге `<body>` можно задавать с помощью следующих констант: `ariseblue`, `aquamarine`, `bisque`, `blue`, `burlywood`, `chocolate`, `cornsilk`, `darkblue`, `darkgray`, `darkmagenta`, `darkorchid`, `darkseagreen`, `darkturquoise`, `deepskyblue`,

firebrick, fuchsia, gold, green, hotpink, ivory, lavenderblue, lightblue, lightgoldenrodyellow, lightpink, lightskyblue, lightyellow, limegreen, mediumaquamarine, mediumpurple, mediumspringgreen, midnightblue, moccasin, oldlace, orange, palegoldenrod, palevioletred, peru, powderblue, rosybrown, salmon, seashell, skyblue, snow, tan, tomato, wheat, yellow, antiquewhite, azure, black, blueviolet, cadetblue, coral, crimson, darkcyan, darkgreen, darkolivegreen, darkred, darkslateblue, darkviolet, dimgray, floralwhite, gainsboro, goldenrod, greenyellow, indianred, khaki, lawngreen, lightcoral, lightgreen, lightsalmon, lightslategray, lime, magenta, mediumblue, mediumseagreen, mediumturquoise, mintcream, navajowhite, olive, orangered, palegreen, papayawhip, pink, purple, royalblue, sandybrown, sienna, slateblue, springgreen, teal, turquoise, white, yellowgreen, aqua, beige, blanchedalmond, brown, chartreuse, cornflowerblue, cyan, darkgoldenrod, darkkhaki, darkorange, darksalmon, darkslategray, deeppink, dodgerblue, forestgreen, ghostwhite, gray, honeydew, indigo, lavender, lemonchiffon, lightcyan, lightgrey, lightseagreen, lightsteelblue, limegreen, maroon, mediumorchid, mediumslateblue, mediumvioletred, mistyrose, navy, olivedrab, orchid, paleturquoise, peachpuff, plum, red, saddlebrown, seagreen, silver, slategray, steelblue, thistle, violet, whitesmoke.

Для включения изображения в качестве фона необходимо добавить в тэг <body> атрибут background, задав в качестве его значения путь файла, содержащего требуемое изображение:

```
<body background="gif/back.gif">.
```

Таким образом, Web-страница имеет следующую структуру:

```
<!-- Примерная структура Web - страницы: -->
<html>
  <head> <title> Добро пожаловать в Web-страницу
</title> </head> <body align="center" alink="tomato"
link="FFFF00">
  Здесь содержимое страницы </body>
</html>
```

## Списки

В HTML-коде можно создавать различные списки. Существует несколько HTML-списков: списки определений, упорядоченные списки с цифровой и символьной нумерацией и неупорядоченные списки с маркерами возле каждого элемента списка.

Видимые заголовки различных уровней включаются тэгами <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, для которых необходимо задавать конечные тэги </h1>, </h2> и т.д.: <center> <h1> Добро пожаловать в Web-страницу </h1></center>

Размер шрифта заголовка более низкого уровня пропорционально уменьшается, т.е. заголовок <h6> будет выводиться самым мелким шрифтом.

Для формирования неупорядоченных списков предназначен тэг <ul>, который имеет следующие атрибуты: align, class, compact, id, lang, language, style, title, type, событие = "сценарий". Элементы списка вводятся тэгом <li>.

который не нуждается в закрывающем тэге. Весь список заканчивается тэгом `</ul>`:

```
<font color = "ffff00"> <ul>
<li> pascal <li> си <li> c++ </ul> </font>
```

Для формирования нумерованных списков используется тэг `<ol>`:

```
<ol>
<li> элемент списка </ol>
```

Для формирования списка с описанием используются следующие тэги:

```
<dl>
<dt> Элемент списка <dd> описание элемента </dl>
```

Возможно также использование вложенных списков:

```
<ul>
<li> элемент 1 списка <ol>
<li> элемент 2 списка </ol>
...
</ul>
```

### Гиперссылки

Гипертекстовый документ-это документ, содержащий ссылки на другие документы. Гипермедийный документ основан на гипертекстовом документе, но в дополнении к тексту содержит разнообразную графику, видео- и аудио объекты. В таких документах в качестве ссылок часто используются изображения. Существует очень много мультимедийных объектов, которые могут быть размещены на Web-странице.

Ссылка в гипертекстовом или гипермедийном документе состоит из двух частей: видимой части или указателя(anchor) и невидимой части или адресной части (URL- reference), дающей инструкцию браузеру о местоположении другого документа.

Указателем может быть слово, группа слов или изображение. Текстовые указатели обычно подчеркнуты одной линией. Их цвет регулируется специальными параметрами, а также установками браузера. Обычно в качестве указателя выбирается тот или иной фрагмент документа. Пример создания текстового указателя:

```
<A HREF= "JavaScript.html">JavaScript</A>
```

Второй частью ссылки является URL-адрес страницы, которая будет загружена, если щелкнуть на указателе ссылки кнопкой мыши. Указание адреса может быть относительным или абсолютным. Относительные указатели являются URL-адресами файлов, расположенных на том же компьютере, что и документ, в котором находится указатель этой ссылки. Таким образом , здесь используется только имя файла, а не весь длинный URL-адрес. Например, если браузер загрузил страницу, находящуюся по адресу `http://www.mysite.com/page` ,то относительный адрес `/picture` подразумевает адрес `http://www.mysite.com/page/picture`, т.е. подкаталог, расположенный на той же машине. Относительные указатели позволяют

размещать файлы в пределах данного сервера без больших изменений в межстраничной адресации. Абсолютные указатели – это URL- адреса, полностью определяющие компьютер, каталог или файл.

В качестве объектов, на которые ссылаются с помощью тэга <A>, могут быть не только Web-страницы, но и файлы различных типов, и даже встраиваемые объекты. Когда пользователь щелкает по ссылке на Web-страницу, производится ее поиск, а затем загрузка в окно Web-браузера. Если же ссылка определяет другой объект, то браузер находит по адресу этот объект и затем решает, что с ним делать. Если браузер знает сам, как с ним обращаться, то он очищает окно и загружает этот объект, например, изображение в формате GIF. Если он сам не умеет с ним обращаться, то он будет искать на машине пользователя соответствующую программу просмотра и запустит ее в дополнительном окне, например, MS Word.

Гипертекстовая и гипермедийная среда являются лишь частью сети Internet. Другие ресурсы начали свое существование задолго до рождения Web-сети. Рассмотренный способ связывания HTML-документов распространяется и на другие ресурсы.

Создание ссылки на электронную почту реализуется следующим образом:

```
<A HREF = "mailto:me@mycom.com">Посылка e-mail</A>
```

После того как пользователь активизирует эту гиперссылку, запустится программа электронной почты, и пользователю будет предоставлена возможность отправить сообщение по адресу, указанному в гиперссылке.

Связывание HTML-документа с группой новостей UseNet реализуется следующей ссылкой:

```
<A HREF = "news:news.newusers.questions">Пенза</A>
```

Браузер связывается с сервером UseNet, к которому данный пользователь имеет доступ, и запускает приложение для работы с телеконфигурациями.

Ссылки на сайты службы FTP осуществляется следующим образом:

```
<A HREF = "ftp://ftp.mysite.com/pub/FAQ">FTP</A>
```

Поскольку технологии WWW и FTP весьма похожи, то браузер сам обрабатывает ссылки. Если ссылка определяет папку, то щелчок по ней приводит к открытию этой папки; если же найден файл, то щелчок по нему приведет к копированию этого файла на компьютер.

### Таблицы

Таблицы предназначены для упорядоченного размещения информации на Web- страницы. В ячейки таблицы можно помещать любую информацию – текст, изображения, гиперссылки и т.п.

Основные тэги таблицы Таблица:

```
<TABLE> . . . </TABLE>
```



Это основные тэги, описывающие таблицу. Все элементы таблицы должны находиться внутри этих двух тэгов. По умолчанию таблица не имеет обрамления и разделителей. Обрамление добавляется атрибутом BORDER.

Строка таблицы:

```
<TR> . . . </TR>
```

Количество строк таблицы определяется количеством встречающихся пар тэгов <TR>..</TR>. Строки могут иметь атрибуты ALIGN и VALIGN, которые описывают визуальное положение содержимого строк в таблице.

Ячейка таблицы:

```
<TD> . . . </TD>
```

Описывает стандартную ячейку таблицы. Ячейка таблицы может быть описана только внутри строки таблицы. Каждая ячейка должна быть пронумерована номером колонки, для которой она описывается. Если в строке отсутствует одна или несколько ячеек для некоторых колонок, то браузер отображает пустую ячейку. Расположение данных в ячейке по умолчанию определяется атрибутами ALIGN="left" и VALIGN="middle". Данное расположение может быть исправлено как на уровне описания строки, так и на уровне описания ячейки.

Заголовок таблицы:

```
<TH> . . . </TH>
```

Ячейка заголовка таблицы имеет ширину всей таблицы; текст в данной ячейке имеет атрибут BOLD и ALIGN="center".

Подпись:

```
<CAPTION> . . . </CAPTION>
```

Данный тэг описывает название таблицы (подпись). Тэг <CAPTION> должен присутствовать внутри <TABLE>...</TABLE>, но снаружи описания какой-либо строки или ячейки. По умолчанию <CAPTION> имеет атрибут ALIGN="top", но может быть явно установлен в ALIGN="bottom". ALIGN определяет, где - сверху или снизу таблицы будет поставлена подпись. Подпись всегда центрирована в рамках ширины таблицы.

*Основные атрибуты таблицы*

**BORDER**

Данный атрибут используется в тэге TABLE. Если данный атрибут присутствует, граница таблицы прорисовывается для всех ячеек и для таблицы в целом. BORDER может принимать числовое значение, определяющее ширину границы, например, BORDER=3.

**ALIGN**

Если атрибут ALIGN присутствует внутри тэгов <CAPTION> и </CAPTION>, то он определяет положение подписи для таблицы (сверху или снизу). По умолчанию ALIGN=top. Если атрибут ALIGN встречается внутри <TR>, <TH> или <TD>, он управляет положением данных в ячейках по горизонтали. Может принимать значения left (слева), right (справа) или center (по центру).

### **VALIGN**

Данный атрибут встречается внутри тэгов <TR>, <TH> и <TD>. Он определяет вертикальное размещение данных в ячейках. Может принимать значения top (вверху), bottom (внизу), middle (по середине) и baseline (все ячейки строки прижаты кверху).

### **NOWRAP**

Данный атрибут говорит о том, что данные в ячейке не могут логически разбиваться на несколько строк и должны быть представлены одной строкой.

### **COLSPAN**

Указывает, какое количество ячеек будет объединено по горизонтали для указанной ячейки. По умолчанию = 1.

### **ROWSPAN**

Указывает, какое количество ячеек будет объединено по вертикали для указанной ячейки. По умолчанию = 1.

### **COLSPEC**

Данный параметр позволяет задавать фиксированную ширину колонок либо в символах, либо в процентах, например COLSPEC="20%".

Пример таблицы:

```
<table border="1"> <tr align = "center">
<th colspan = "3"> Это таблица </th> </tr>
<tr align = "center">
<td> Это </td> <td> ячейки </td> <td> для </td>
</tr>
<tr align = "center">
<td> данных </td> <td> в </td> <td> таблице </td>
</tr>
</table>
```

### **Изображения**

Одна из наиболее привлекательных черт Web - возможность включения ссылок на графические и иные типы данных в HTML-документ. Делается это при помощи тэга <IMG>. Использование данного тэга позволяет значительно улучшить внешний вид документов.

Синтаксис тэга:

```
<IMG SRC="URL" ALT="text" HEIGHT="n1" WIDTH="n2"
ALIGN="top"|"middle"|"bottom"|"texttop" ISMAP>
```

*Атрибуты тэга <img>*

### **URL**

Обязательный параметр, имеющий такой же синтаксис, как и стандартный URL. Данный URL указывает браузеру где находится рисунок. Рисунок должен храниться в графическом формате, поддерживаемом браузером. На сегодняшний день форматы GIF и JPG поддерживаются большинством браузеров.

*ALT="text"*

Данный необязательный элемент задает текст, который будет отображен браузером, не поддерживающим отображение графики или с отключенной подкачкой изображений. Обычно, это короткое описание изображения, которое пользователь мог бы или сможет увидеть на экране. Если данный параметр отсутствует, то на месте рисунка большинство браузеров выводит пиктограмму (иконку), активизировав которую, пользователь может увидеть изображение. Тэг ALT рекомендуется, если ваши пользователи используют браузер, не поддерживающий графический режим, например Lynx.

*HEIGHT="n1"*

Данный необязательный параметр используется для указания высоты рисунка в пикселах. Если данный параметр не указан, то используется оригинальная высота рисунка. Это параметр позволяет сжимать или растягивать изображения по вертикали, что позволяет более четко определять внешний вид документа. Однако, некоторые браузеры не поддерживают данный параметр. С другой стороны, экранное разрешение у вашего клиента может отличаться от вашего, поэтому будьте внимательны при задании абсолютной величины графического объекта.

*WIDTH="n2"*

Параметр также необязателен, как и предыдущий. Позволяет задать абсолютную ширину рисунка в пикселах.

*ALIGN*

Данный параметр используется, чтобы сообщить браузеру, куда поместить следующий блок текста. Это позволяет более строго задать расположение элементов на экране. Если данный параметр не используется, то большинство браузеров располагает изображение в левой части экрана, а текст справа от него.

*ISMAP*

Этот параметр сообщает браузеру, что данное изображение позволяет пользователю выполнять какие-либо действия, щелкая мышью на определенном месте изображения.

С версии HTML 2.0 у тэга <IMG> появились дополнительные параметры:

*BORDER*

Данный параметр позволяет автору определить ширину рамки вокруг рисунка.

*VSPACE*

Позволяет установить размер в пикселах пустого пространства над и под рисунком, чтобы текст не наезжал на рисунок. Особенно это важно для динамически формируемых изображений, когда нельзя заранее увидеть документ.

## *HSPACE*

То же самое, что и VSPACE, но только по горизонтали. Пример использования тэга <img>:

```
<IMG  
SRC="http://www.softexpress.com/images/nektion.jpg"  
ALT="СофтСервис лого" ALIGN="top" />
```

## **Фреймы**

Фреймы позволяют разбить окно просмотра браузера на несколько прямоугольных подобластей, располагающихся рядом друг с другом. В каждую из подобластей можно загрузить отдельный HTML - документ, просмотр которого осуществляется независимо от других. Между фреймами можно организовать взаимодействие. Фреймы определяются в структуре, называемой <frameset>, которая используется для страниц, содержащих фреймы, вместо раздела body обычного документа. Контейнер из тэгов <frameset> и </frameset> обрамляет каждый блок определений фрейма. Тэг <frameset> имеет два параметра: rows(строки) и cols(столбцы). Тэг <frame> определяет одиночный фрейм и располагается внутри пары тэгов <frameset> и </frameset>.

Параметры тэга <frame>:

- src - определяет адрес загружаемого документа
- name - определяет имя фрейма
- marginwidth - устанавливает ширину полей рамки
- marginheight - устанавливает ширину полей рамки
- scrolling - управление отображением полос прокрутки
- noresize - не позволяет пользователю изменять размеры фрейма

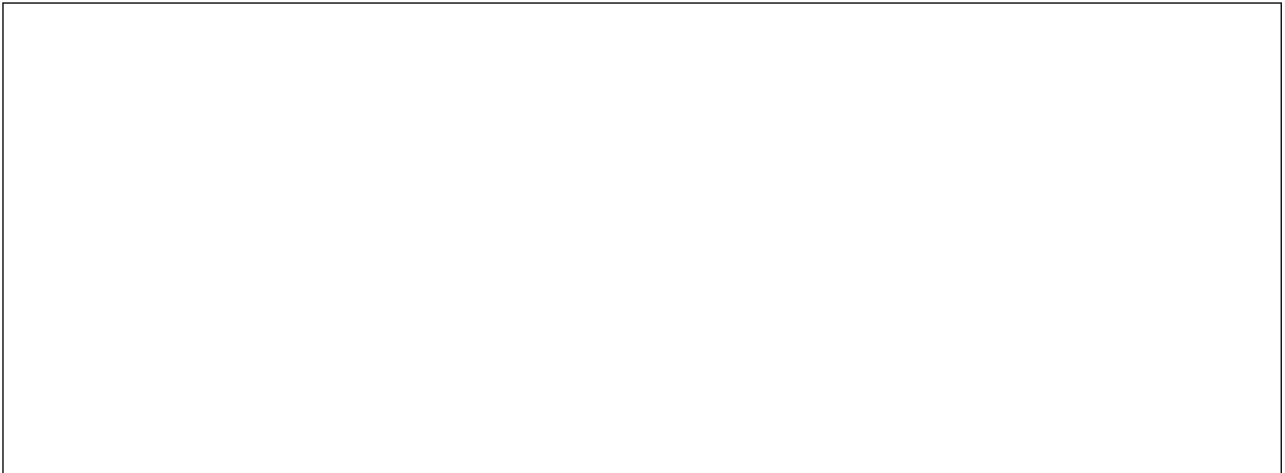
Пример:

Файл "index.htm"

Файл "header.htm"

```
<HTML>  
<BODY bgcolor="snow"> <CENTER>  
<H1>Данный фрейм содержит заголовок</H1> </CENTER>  
</BODY>  
</HTML>
```

Файл "menu.htm" <HTML>



Файл "info\_1.htm" <HTML>

```
<BODY bgcolor="lightgrey"> <H3>Правый фрейм</H3>
Файл "info_1.htm" <br/>
    Данный фрейм может содержать какую-либо информацию
по первому разделу </BODY>
</HTML>
```

Файл "info\_2.htm" <HTML>

```
<BODY bgcolor="blue"> <H3>Правый фрейм</H3> Файл
"info_2.htm" <br/>
    Данный фрейм может содержать какую-либо информацию
по ВТОРОМУ разделу </BODY>
</HTML>
```

Файл "footer.htm"

```
<HTML>
<BODY bgcolor="snow"> <CENTER>
<H3>Фрейм, представляющий нижнюю часть
страницы ("Подвал")</H3> </CENTER>
</BODY>
</HTML>
```

Вот такая страница должна получиться в результате



## Формы

Форма является средством интерактивного взаимодействия пользователей с сетью Интернет. Она представляет собой контейнер для размещения полей ввода и управляющих элементов. После того как пользователь отправит информацию, она обрабатывается программой (скриптом), размещенной на сервере. Скрипт – это короткая программа, специально созданная для обработки каждой формы. Существует также возможность создавать формы непосредственно в окне браузера читателя.

Каждая форма начинается тэгом `<FORM>`, в котором нужно определить два атрибута:

- ACTION – для задания URL, который примет и обработает данные формы; если атрибут не определен, данные отправляются по адресу страницы, на которой помещена форма;
- METHOD – для указания способа посылки информации для обработки скриптом:
  - POST – информация посылается отдельно от URL;
  - GET – информация посылается вместе с URL.

Например, если скрипт `comment_script` будет расположен в каталоге `cgi_bin` используемого сервера, то для метода POST форму следует определить таким образом:

```
<FORM METHOD = "POST" ACTION =
"/cgi_bin/comment_script">
```

На странице может быть расположено несколько форм, но их вложение не допускается.

Для создания в форме полей используются тэги: `<TEXTAREA>`, `<SELECT>` и `<INPUT>`.

Тэг `<TEXTAREA>` предназначен для построения поля ввода многострочной информации и имеет атрибуты:

- NAME – имя поля ввода (обязательный атрибут);
- ROWS – число текстовых строк;
- COLS – длина строки.

По умолчанию ROWS и COLS определяются используемым браузером.

Пример определения многострочного поля ввода с начальной информацией таков:

```
<TEXTAREA NAME = "RichEdit" ROWS=4 COLS=4>
Начальные данные </TEXTAREA>
```

Текст рассматривается как предварительно отформатированный.

Тэг <SELECT> используется для создания всплывающего меню или списка опций с полосой прокрутки и имеет атрибуты:

- NAME – название меню с опциями;
- SIZE – вертикальный размер окна; если он равен 1 или отсутствует, то выводится весь список, иначе- часть списка с полосой прокрутки; когда размер превышает числа опций, то список дополняется пустыми строками, при выборе которых возвращается пустое значение;
- MULTIPLE – возможность одновременного выбора нескольких опций.
- Опции включаются тэгом <OPTION> с атрибутами:
- VALUE – возвращаемое значение для скрипта;
- SELECTED – определяет опцию, выделенную по умолчанию.

Элементы меню записываются вслед за тэгами <OPTION>. Они определяют ширину окна.

Рассмотрим пример меню со списком опций:

```
<HTML><HEAD><TITLE>Пример меню
</TITLE></HEAD><BODY>
... ..
<FORM>
<SELECT NAME = "File">
  <OPTION SELECTED VALUE = "New"> создать <OPTION
VALUE = "Open"> открыть <OPTION VALUE = "Save">
сохранить </SELECT></FORM></BODY></HTML>
```

Тэг <INPUT> не является контейнером, т.е. не имеет закрывающего тэга </INPUT>, предназначен для сбора информации различными способами, включая текстовые поля, поля для ввода пароля, переключатели, флажки, кнопки для отправки данных (Submit) и для очистки формы (Reset, Clear и т.п.). Он имеет следующие атрибуты:

- NAME – имя элемента;
- SIZE – размер поля в символах;
- MAXLENGTH – максимальное число вводимых символов;
- VALUE – значение:
  - для текстового поля – текст по умолчанию;

- для флажков и переключателей – значение, возвращаемое программе обработки;
- для кнопок отправки и очистки формы – надпись на кнопке;
- **CHECKED** – включенное по умолчанию состояние флажка или переключателя;
- **TYPE** – тип элемента формы:
  - **TEXT** – поле ввода одной строки (по умолчанию предполагается);
  - **PASSWORD** – поле для ввода пароля;
  - **CHECKBOX** – флажок в виде квадрата;
  - **RADIO** – переключатель - радиокнопка;
  - **RESET** – кнопка для очистки формы;
  - **SUBMIT** – кнопка для отправки введенных данных на сервер для обработки программой – скриптом.

Рассмотрим пример использования тэга **<INPUT>**.

```

... ..
<FORM>
  Номер телефона: <INPUT TYPE = "text" NAME =
"Phone" size = "15" MAXLENGTH = "12">
  Введите пароль: < INPUT TYPE = "password" NAME =
"Key" Size = "30" MAXLENGTH = "30">
  <INPUT TYPE = "checkbox" NAME = "chbx1" VALUE =
"Val1"> Вкл<INPUT TYPE = "checkbox" NAME = "chbx2"
VALUE = "Val2" CHECKED> Питание </FORM>

```

## Введение в JavaScript

### Язык сценариев JavaScript

Клиентская часть среды проектирования Web-приложений содержит следующие основные компоненты:

- Браузер (или обозреватель), отображающий HTML-документ на экране монитора и являющейся пользовательским интерфейсом для Web-приложений;
- Язык HTML, с помощью которого создаются Web-страницы;
- Языки сценариев, в настоящее время в качестве стандарта принят язык JavaScript;
- Клиентские расширения, такие как элементы управления ActiveX, Java-апплеты, подключаемые модули.

Язык сценариев JavaScript предназначен для создания интерактивных HTML- документов. С помощью сценариев поддерживается диалог с пользователем, обеспечивается привлекательный вид Web-страниц, осуществляется навигация по странице сайта, поиск элементов на странице и многое другое. Основой языка является понятие объект. JavaScript может быть внедрен в HTML-документ, при этом он обеспечивает работу в среде, поддерживаемой браузерами.



Приложения, написанные на JavaScript, могут выполняться как на стороне клиента, так и на стороне сервера. При разработке приложений обоих типов используется так называемое ядро, в котором содержится определения стандартных объектов. Клиентские приложения выполняются браузером на машине пользователя.

Программа (сценарий) на языке JavaScript обрабатывается встроенным в браузер интерпретатором.

### **Обработчик событий**

Интерактивные документы создаются с помощью форм, в которой происходит перехват и обработка события, которое задается в параметрах форм. Имя параметра обработки события начинается с приставки on, за которой следует имя самого события.

Приведем примеры основных событий и элементы документов, в которых эти события могут происходить.

Событие	Объекты	Когда происходит событие?
Abort	image	Отказ от загрузки изображения
Blur	window, элементы форм	Потеря объектом фокуса
Change	text, texarea, select	Изменение значения элемента
Click	button, radio, checkbox, submit, reset, link	Щелчок на элементе связи
DragDrop	window	Перетаскивается мышью объект в окно браузера
Error	image, windows	Ошибка при загрузке документа или изображения
Focus	window, элементы форм	Окно или элемент формы получает фокус
KeyDown	document, image, link, textarea	Нажатие клавиши клавиатуры
KeyPress	document, image, link, textarea	Удержание клавиши клавиатуры
KeyUp	document, image, link, textarea	Отпускается клавиша клавиатуры
Load	тело документа	Загружается документ в браузер
MouseDown	document, button, link	Нажатие кнопки мыши
MouseOut	area, link	Перемещение курсора из области изображения или связи
MouseOver	link	Перемещение курсора над связью
MouseUp	document, button, link	Отпускается кнопка мыши
Move	window	Пользователь или сценарий перемещает окно
Reset	form	Нажатие на кнопку Reset формы
Resize	window	Пользователь или сценарий изменяет

		размеры окна
Select	text, textarea	Выбирается поле ввода элемента формы
Submit	form	Нажатие на кнопку Submit формы
Unload	тело документа	Пользователь закрывает окно

### Сценарии в HTML-документе

Для обеспечения интерактивных HTML-документов используются сценарии, которые пишутся на языке сценариев JavaScript. Сценарии могут располагаться непосредственно в HTML-документе между тегами `<script>` и `</script>`.

Одним из параметров тега `<script>` является `language`, который определяет язык сценариев. Для JavaScript значение этого параметра "JavaScript", который одновременно является значением параметра по умолчанию, т.е. его можно опустить для языка JavaScript.

Некоторые браузеры могут не поддерживать какие-то теги, поэтому страница может отображаться неверно. Для того, чтобы избежать такой ситуации, рекомендуется операторы языка сценариев ставить в теги комментариев `<!-- ...-->`. Для правильной работы интерпретатора требуется ставить перед закрывающим тегом комментария.

Документ может содержать несколько тегов `<script>`.

#### Функции

Основным элементом языка JavaScript является функция. Описание функции имеет следующий вид:

```
function F(V){S},
```

где `F` – идентификатор функции, задающий имя функции, к которому можно обращаться, `v` – список формальных параметров функции, которые перечисляются через запятую, `S` – тело функции, в котором описаны действия.

Вложенности функции не допускается. Параметры в описании функции могут отсутствовать. Обычно описание функций задается в разделе `<HEAD>` документа, что обеспечивает интерпретацию и сохранение в памяти всех функций при загрузке документа в браузер.

#### Объекты клиента

При интерпретации HTML-страницы браузером создаются объекты JavaScript. Взаимосвязь объектов между собой представляет собой иерархическую структуру. В вершине этой иерархии находится объект `window`, представляющий собой окно браузера, он является родителем для всех остальных объектов. Каждая страница объекта `document`, определяется содержимым самого документа и обладает свойствами, такими, как: цвет фона, цвет шрифта, и т.д.

Каждый объект обладает определенными свойствами и методами.

- `window` – объект окна браузера;
- `document` – объект отображаемого документа;

- history – объект, содержащий информацию об адресах ранее загружаемых страниц HTML;
- frame- объект фрейма (специальное представление данных);
- location – объект, связанный с URL-адресом отображаемого документа.

Любой элемент (объект) документа представляет собой массив. Индексация массива начинается с нуля, поэтому к элементам массива можно либо по имени, задаваемом в соответствующем теге параметром name, либо как к элементу массива в соответствии с иерархией объектов. Например: document.forms[0], document.forms[1] и т.д.

Каждый объект, кроме свойств обладает и методами, которые можно вызывать для выполнения определенных действий через объект. Рассмотрим основные свойства и методы некоторых объектов.

#### *Объект браузера*

В языке JavaScript определены объекты, которые называются объектами браузера. Управлять частями документа можно с помощью методов браузера. Свойство window.status можно использовать для изменения строки состояний.

Объект window имеет три метода:

1. window.alert отображает диалоговое окно, в которое помещается сообщение для пользователя. Данный метод используется при обработки полей формы.

Пример:

```
function sum()
{window.status = "Неверно выделены данные";
  alert ("Проверьте правильность введенных оценок");
window.status=""; }
```

2. Метод confirm отображает диалоговое окно подтверждения выполнения операции, которое содержит две кнопки OK и Cancel и позволяет выбрать один из вариантов. Если пользователь выбирает JR, то функция возвращает true, иначе false.

```
function rb()
{if (confirm ("Хотите закончить работу?"))
document.write ("Действия по завершению работы") else
  document.write ("Отмена выхода по завершению
работы") }
```

3. Метод prompt() используется для ввода диалогового окна запроса данных. При щелчке по кнопке OK введенные пользователем в текстовое поле данные отображаются в документе. Если выбрано Cancel, возвращается Null. Метод prompt() имеет второй параметр, с помощью которого задается значение по умолчанию HTML-код со сценарием, использующим метод prompt().

Кроме этого при выполнении сценария пользователь может создавать новые окна и загружать в них документы. Это делается с помощью метода open.

```
Например :  
window.open( "Anketa.htm", "mywin", "")
```

Данный метод имеет 3 параметра:

- URL-документа, загружаемого во вновь открываемое окно. Если URL не указано, то создается пустое окно.
- Название вновь создаваемого окна.
- Задаёт параметры окна по умолчанию.

Закреть окно браузера можно с помощью метода close.

#### *Объект history*

Объект history представляет адреса ранее загруженных HTML-страниц. Объект имеет свойства, которые хранят информацию о ранее загруженных страницах, а также метод, которые позволяют загружать предыдущие и следующие страницы. К элементам объекта history можно обращаться по индексу, в этом случае history рассматривается как массив, где текущая страница представлена элементом history[0], предыдущая – history[-1], следующая – history[1].

Объект history имеет следующие свойства и методы.

1. Свойство length содержит длину списка адресов посещаемых страниц.
2. Свойство current - адрес текущей страницы.
3. Свойство next - адрес следующей страницы к которой перейдет пользователь по кнопке ВПЕРЕД.
4. Свойство previous - адрес страницы по которой перейдет пользователь по кнопке НАЗАД.

Кроме этого существуют методы:

1. Метод go, который позволяет перейти на указанную страницу из списка посещаемых страниц. Например: history.go(-1), history.go(2) и т.д.
2. Метод forward, который аналогичен свойству next.
3. Метод back, аналогичный свойству previous.

#### *Объект location*

Объект location связан с URL-адресом отображаемого документа.

Данный объект имеет два метода.

- Метод reload, который перезагружает текущий документ.
- Метод replace, который заменяет текущую страницу, на адрес страницы, который указан в качестве параметра. Данный метод не изменяет историю браузера.

#### *Объект frame*

Объект frame связан со специальным способом представления данных в виде фреймов, который задается в теге <FRAMESET>, <FRAME> и <NOFRAMES>. В этом случае окно браузера разбивается на отдельные прямоугольные части, размер которых и их количество задается в параметра тегов. Фреймы позволяют изменять содержимое этих окон независимо друг от

друга или устанавливать связь между ними, т.е. при изменении параметров одного окна, изменять параметры другого.

Объект `frame` свойства: `top`, `left`, `right`. Доступ к каждому фрейму осуществляется с помощью значения параметра `target`, а свойство фрейма `location` определяет URL-адреса загружаемых во фрейм страниц.

Для работы с фреймами можно использовать имена, задаваемые параметром `name` или по индексу.

Например:  
`top.frame[0].location.`

Для того, чтобы выйти на фреймовую структуру следующего уровня, необходимо использовать свойство `parent`.

### *Объект document*

Свойство объекта `document` определяется содержимым самого документа: шрифт, цвет фона, формы, изображения. Объект `document`, в зависимости от своего содержимого, может иметь объекты, являющиеся для него подчиненными (в соответствии с иерархической структурой).

Кроме этого данный объект имеет свои методы: `write` и `writeln`, запись в документ текста, указанного в параметрах этих методов. Это позволяет формировать в интерактивном режиме выводимый в документ необходимые данные.

Все элементы документа представляют из себя также массивы, поэтому обращение к ним возможно как по имени, которое задается параметром `name`, либо по индексу.

Например:  
`document.form[0].elements[0]`

Таким образом, необходимо помнить, что при разработке своего сценария вы должны понимать с каким объектом вы собираетесь работать, представлять его структуру и правильно использовать те методы и свойства, которые вам предоставлены.

### *Переменные*

Переменные в JavaScript могут быть определены назначением или при помощи оператора `var`:

```
i=10; var a; var a=10;
var id = window.open(); var m = new Array();
```

Т.е., если мы инициализируем переменную при её создании, то ключевое слово `var` разрешается опустить.

Как видно из примеров, переменные могут принимать самые разные значения, при этом тип переменной определяется контекстом.

Переменная может являться свойством окна. Например, мы можем открыть окно, определить в нем новую переменную и использовать ее:

```

        wid
window.open("", "kuku", "width=200,height=100,statusbar")
;
wid.document.open();
wid.document.write("<html><head>");
wid.document.write("<script>var                t;</script>");
wid.document.write("</head><body>");
wid.document.write("<center>Новое                окно<br>");
wid.document.write("<form>");
wid.document.write("<input type=button value='Закреть
окно'                onClick=window.close();></form>");
wid.document.write("</center></body></html>");
wid.document.close();

...
<a href="javascript:wid.t=window.prompt("Type new
status
value:");wid.defaultStatus=t;wid.focus();void(0);>...</
a>

```

Существуют ли в JavaScript различные типы переменных? По всей видимости, да. При объявлении переменной тип не указывается. Тип значения определяется контекстом. Поэтому существует соблазн предположить, что все переменные одного и того же типа. Однако очевидно, что присваивание переменной значения объекта окна (window.open()) или объекта даты(Date()), или любого другого значения (строки, числа), порождает создание совершенно разных структур в памяти.

Однако, одна и та же переменная принимать значения разных типов. Это означает, что JavaScript всё-таки поддерживает полиморфизм, т.е. существуют два разных объекта с одинаковыми именами и система в них не путается.

### *Массивы*

Массивы делятся на встроенные (document.links[], document.images[], ...) и определяемые пользователем (автором документа). Встроенные массивы мы подробно обсуждаем в разделах "Программируем картинки", "Программируем формы" и "Программируем гипертекстовые переходы". Поэтому подробно остановимся на массивах, определяемых пользователем. Для массивов определено несколько методов:

- join()
- reverse()
- sort()

И свойство length, которое позволяет получить число элементов массива.

Метод join() позволяет объединить элементы массива в одну строку.

Метод reverse() применяется для изменения на противоположный порядок элементов массива внутри массива.

Метод sort() сортирует элементы массива.

Для определения массива пользователя существует специальный конструктор:

```
a = new Array(); b = new Array(10); c = new Array(10, "Это значение");
```

Пример использования:

```
<script>
c= new Array(30, "Это значение"); </script>
<form><input size=& {c[0]}; value=& {c[1]};
onFocus="this.blur();" > </form>
```

Как видно из этого примера, массив может состоять из разнородных элементов. Массивы не могут быть многомерными.

*Method sort()*

Как это принято в современных интерпретируемых языках, например в Perl, метод sort() позволяет отсортировать элементы массива в соответствии с некоторой функцией сортировки, чье имя используется в качестве аргумента метода:

```
a = new Array(1,6,9,9,3,5); function g(a,b)
{
  if(a > b) return 1; if(a < b) return -1; if(a==b)
return 0;
}
b = a.sort(g);
```

В результате выполнения этого кода получим массив следующего вида:

```
b[0]=1
b[1]=3
b[2]=5
b[3]=6
b[4]=9
b[5]=9
```

## **Функции и объекты пользователя**

### *Функции*

Язык программирования не может обойтись без механизма многократного использования кода программы. Такой механизм обеспечивается процедурами или функциями. В JavaScript функция выступает в качестве одного из основных типов данных. Одновременно с этим в JavaScript определен объект Function.

Вобщем случае любой объект JavaScript определяется через функцию. Для создания объекта используется конструктор, который в свою очередь вводится через Function.

```
function f_name(arg1, arg2, ...)
{
  /* function body */
}
```

```
}
```

Вследующем примере демонстрируется работа с функциями JavaScript:

```
<html>
  <head><title>Работа с функциями</title></head>
<body>
  <input type="button" value="Выполнение функции 1"
onclick="func_1()">      <input type="button"
value="Выполнение функции 2 с передачей аргумента"
onclick="func_2('argument_1','argument_2')">
  </body>
  <script> func_3(a,b){ sum = a + b; return sum;
  }
  function func_1(){
    alert("Запущена функция 1");
    returnValue = func_3(3,4); //Функция возвращает
сумму двух чисел alert(returnValue);
  }
  function func_2(str1, str2){ alert("Запущена
функция 2"); alert("Аргумент 1 равен " + str1);
alert("Аргумент 2 равен " + str2);
  }
  </script>
</html>
```

### *Объекты*

Объект - это ключевой, главный тип данных JavaScript. Тип данных Object сам определяет объекты.

Для определенности сначала рассмотрим пример произвольного, определенного пользователем объекта, потом определимся с тем, что же это такое:

```
function Rectangle(a,b,c,d)
{
  this.x0 = a; this.y0 = b; this.x1 = c; this.y1 =
d;
  this.area = new Function("return Math.abs(this.x0-
this.x1)*Math.abs(this.y0-this.y1)");this.perimeter =
new Function("return (Math.abs(this.x0-
this.x1)+Math.abs(this.y0-this.y1))*2");
}
c = new Rectangle(0,0,100,100);
document.write(c.area());
```

Функция Rectangle() - это конструктор объекта класса Rectangle, который определяется пользователем. Конструктор позволяет создать реальный объект данного класса. Ведь функция - это не более чем описание



некоторых действий. Для того чтобы эти действия были выполнены, необходимо передать управление функции. В нашем примере это делается при помощи оператора `new`. Он вызывает функцию и тем самым создает реальный объект.

Создаются четыре переменных: `x0,y0,x1,y1` - это свойства объекта `Rectangle`. К ним можно получить доступ только в контексте объекта данного класса.

```
Например:  
up_left_x = c.x0;  
up_left_y = c.y0;
```

Кроме свойств мы определили внутри конструктора два объекта типа `Function()`, применив встроенные конструкторы языка JavaScript, - `area` и `perimeter`. Это методы объекта данного класса. Вызвать эти функции можно только в контексте объекта класса `Rectangle`:

```
sq = c.area();  
length = c.perimeter();
```

Таким образом, объект - это совокупность свойств и методов, доступ к которым можно получить только, создав при помощи конструктора объект данного класса и используя его контекст.

### *Прототип*

Обычно мы имеем дело со встроенными объектами JavaScript. Собственно, все, что изложено в других разделах курса - это обращение к свойствам и методам встроенных объектов. В этом смысле интересно свойство объектов, которое носит название `prototype`. Прототип - это другое название конструктора объекта конкретного класса. Например, если мы захотим добавить метод к объекту класса `String`:

```
String.prototype.out=newFunction("a","a.write(this  
)");  
...  
"Привет дуралии".out(document);
```

Для объявления нового метода для объектов класса `String` мы применили конструктор `Function`. Есть один небольшой, но существенный нюанс: новыми методами и свойствами будут обладать только те объекты, которые порождаются после изменения прототипа объекта. Все встроенные объекты создаются до того, как JavaScript программа получит управление, что существенно ограничивает применение свойства `prototype`.

## **Введение в PHP**

Чтобы проверить работу примеров необходимо вначале установить веб-сервер. Затем, в любом текстовом редакторе набрать код программы и сохранить его в файле с расширением `*.php`. Этот файл необходимо разместить в директории веб-сервера (как правило, в директории `www`). Далее необходимо запустить браузер и в адресной строке набрать `http://localhost/my_program.php/`

Программы PHP могут выполняться двумя способами: как сценарное приложение Web-сервером и как консольные программы. Поскольку, нашей задачей является программирование web-приложений, мы преимущественно будем рассматривать первый способ.

Рассмотрим процесс выполнения php-сценария при обращении браузера к серверу. Итак, вначале браузер запрашивает страницу с расширением .php, после чего web-сервер пропускает программу через машину PHP и выдаёт результат в виде html-кода. Причем, если взять стандартную страницу HTML, изменить расширение на .php и пропустить её через машину PHP, последняя просто перешлёт её пользователю без изменений. Чтобы включить в этот файл команды PHP, необходимо заключить команды PHP в специальные теги, которых различают 4 вида (они эквивалентны и можно использовать любые): Инструкция обработки XML:

```
<?php
...
?>
Инструкция обработки SGML: <?
...
?>
Инструкция обработки сценариев HTML: <script
language = "php">
...
</script>
Инструкция в стиле ASP: <%
...
%>
```

Вообще говоря, внутри какого-либо блока кода можно выйти из PHP, при условии, что дальше мы войдем в него снова и закончим код. Т.е., возможна следующая конструкция:

```
<?
if(5<3){
echo("<p>Hello, world!<p>"); ?>
<p>Hello!</p>
//эта строка не интерпретируется как код PHP
//и выводится только если блок кода выполняется
<?
echo("<p>Hello, world!<p>");
}
?>
```

Команда echo в PHP применяется для вывода фактически всего, что встречается на web-страницах (текст, разметку HTML, числа). Смысл ее действия, мы думаем, понятен из приведенного примера.

Одним из главных достоинств PHP является тот факт, что он внедряется прямо в HTML-код, поэтому программисту не приходится писать

программу с множеством команд для простого вывода HTML. Код HTML и PHP можно чередовать по мере необходимости. PHP позволяет написать фрагмент следующего вида:

```
<html>
<title><? print "Hello world!"; ?></title>
</html>
```

Сообщение "Hello world!" выводится в заголовке web-страницы. Интересно то, что команда print внутри конструкции, которая обычно называется экранирующими последовательностями PHP (<?...?>), представляет собой законченную программу. Ни длинного кода инициализации, ни включения библиотек — программа состоит лишь из того кода, который непосредственно решает поставленную задачу!

Пример, приведенный ниже, наглядно показывает, как легко PHP интегрируется с HTML-кодом.

## Управляющие конструкции PHP

### Конструкция if

Синтаксис конструкции if аналогичен конструкции if в языке Си:

```
<?php
if (логическое выражение) оператор; ?>
```

Согласно выражениям PHP, конструкция if содержит логическое выражение. Если логическое выражение истинно (true), то оператор, следующий за конструкцией if будет исполнен, а если логическое выражение ложно (false), то следующий за if оператор исполнен не будет. Приведем пример:

```
<?php
if ($a > $b) echo "значение a больше, чем b"; ?>
```

### *Цикл со счетчиком for*

Цикл со счетчиком используется для выполнения тела цикла определенное число раз.

С помощью цикла for можно (и нужно) создавать конструкции, которые будут выполнять действия совсем не такие тривиальные, как простая переборка значения счетчика.

Синтаксис цикла for такой:

```
for (инициализирующие_команды; условие_цикла;
команды_после_итерации) { тело_цикла; }
```

Цикл for начинает свою работу с выполнения инициализирующих\_команд. Данные команды выполняются только один раз. После этого проверяется условие\_цикла, если оно истинно (true), то выполняется тело\_цикла. После того, как будет выполнен последний оператор тела, выполняются команды\_после\_итерации. Затем снова проверяется условие\_цикла. Если оно истинно (true), выполняется тело\_цикла и команды\_после\_итерации, и.т.д.

```
<?php for ($x=0; $x<10; $x++) echo $x; ?>
```

### *Цикл перебора массивов foreach*

Данный цикл предназначен специально для перебора массивов. Синтаксис цикла foreach выглядит следующим образом:

```
foreach (массив as $ключ=>$значение) команды;
```

Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара ключ=>значение оказывается в переменных \$ключ и \$значение. Приведем пример работы цикла foreach:

```
<?php
$names["Иванов"] = "Андрей"; $names["Петров"] =
"Борис"; $names["Волков"] = "Сергей"; $names["Макаров"]
= "Федор"; foreach ($names as $key => $value) { echo
"<b>$value $key</b><br>";
}
?>
```

Рассмотренный сценарий выводит: Андрей Иванов Борис Петров Сергей Волков Федор Макаров

### **Работа с MySQL**

MySQL – это одна из самых популярных и самых распространенных СУБД (система управления базами данных) в интернете. Она не предназначена для работы с большими объемами информации, но ее применение идеально для интернет сайтов, как небольших, так и достаточно крупных.

MySQL отличается хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей. Поддержка сервера MySQL автоматически включается в поставку PHP.

Приложение на PHP, использующее для хранения информации базу данных (в частности MySql) всегда работает быстрее приложения,

построенного на файлах. Дело в том, что базы данных написаны на языке C++, и написать на PHP программу, которая работала бы с жёстким диском эффективнее базы данных - задача неразрешимая по определению, поскольку программы на PHP в принципе работают медленнее, чем программы на C++, так как PHP - интерпретатор, а C++ - компилятор.

Таким образом, основное достоинство базы данных заключается в том, что она берёт на себя всю работу с жёстким диском и делает это очень эффективно.

Создать базу данных, таблицы и заполнить их записями можно несколькими путями: работая с сервером MySQL напрямую, используя программу веб-сервера phpMyAdmin, а также с помощью php-скрипта.

Создадим базу данных и одну таблицу в phpMyAdmin. Для запуска программы в адресной строке браузера нужно набрать <http://localhost/phpMyAdmin.php/>

Рассмотрим основные функции PHP, применяемые для работы с MySQL сервером.

#### *Функции соединения с сервером MySQL*

Основной функцией для соединения с сервером MySQL является `mysql_connect()`, которая подключает скрипт к серверу баз данных MySQL и выполняет авторизацию пользователя базой данных. Синтаксис у данной функции такой:

```
mysql_connect ([string $hostname] [, string $user]
[, sting $password]);
```

Как вы наверно заметили, все параметры данной функции являются необязательными, поскольку значения по умолчанию можно прописать в конфигурационном файле `php.ini`. Если вы хотите указать другие имя MySQL-хоста, пользователя и пароль, вы всегда можете это сделать. Параметр `$hostname` может быть указан в виде: хост:порт.

Функция возвращает идентификатор (типа `int`) соединения, вся дальнейшая работа осуществляется только через этот идентификатор. При следующем вызове функции `mysql_connect()` с теми же параметрами новое соединение не будет открыто, а функция возвратит идентификатор существующего соединения.

Для закрытия соединения предназначена функция `mysql_close(int $connection_id)`.

#### *Функция выбора базы данных*

Функция `mysql_select_db (string $db [, int $id])` выбирает базу данных, с которой будет работать PHP скрипт. Если открыто не более одного соединения, можно не указывать параметр `$id`.

```
//Попытка установить соединение с MySQL: if
(!mysql_connect($server, $user, $ password)) { echo
"Ошибка подключения к серверу MySQL"; exit;
}
```

### *Функции выполнения запросов к серверу баз данных*

Все запросы к текущей базе данных отправляются функцией `mysql_query()`. Этой функции нужно передать один параметр - текст запроса. Текст запроса может содержать пробельные символы и символы новой строки (`\n`). Текст должен быть составлен по правилам синтаксиса SQL. Пример запроса:

```
$q = mysql_query("SELECT * FROM mytable");
```

Приведенный запрос должен вернуть содержимое таблицы `mytable`. Результат запроса присваивается переменной `$q`. Результат - это набор данных, который после выполнения запроса нужно обработать определенным образом.

### *Функции обработки результатов запроса*

Если запрос, выполненный с помощью функции `mysql_query()` успешно выполнен, то в результате клиент получит набор записей, который может быть обработан следующими функциями PHP:

- `mysql_result()` - получить необходимый элемент из набора записей;
- `mysql_fetch_array()` - занести запись в массив;
- `mysql_fetch_row()` - занести запись в массив;
- `mysql_fetch_assoc()` - занести запись в ассоциативный массив;
- `mysql_fetch_object()` - занести запись в объект.

Также можно определить количество содержащихся записей и полей в результате запроса. Функция `mysql_num_rows()` позволяет узнать, сколько записей содержит результат запроса:

```
$q = mysql_query("SELECT * FROM mytable");  
echo "В таблице mytable ".mysql_num_rows($q)."  
записей";
```

Запись состоит из полей (колонок). С помощью функции `mysql_num_fields()` можно узнать, сколько полей содержит каждая запись результата:

```
$q = mysql_query("SELECT * FROM mytable");  
echo "В таблице mytable ".mysql_num_fields($q)."  
полей ";
```

У нас также есть возможность узнать значение каждого поля. Это можно сделать с помощью следующей функции:

```
mysql_result (int $result, int $row, mixed  
$field);
```

Параметр функции `$row` задает номер записи, а параметр `$field` - имя или порядковый номер поля.

Предположим, SQL-запрос вернул какой-либо набор данных. Вывести результат в браузер можно следующим образом:

```

$rows      =      mysql_num_rows($q);      $fields      =
mysql_num_fields($q); echo "<pre>";
    for      ($c=0;      $c<$rows;      $c++)      {      for      ($cc=0;
$cc<$fields;      $cc++)      {      echo      mysql_result($q,      $c,
$cc)."\t"; echo "\n";
    }
    }
echo "</pre>";

```

Пример использования функции `mysql_fetch_array()`:

```

$q      =      mysql_query("SELECT      *      FROM      mytable      WHERE
month=\"январь\" AND day=\"10\"); for ($c=0; $c
{
    $f      =      mysql_fetch_array($q);
    echo      "$f[email].\"      \"$f[name].\"      \"$f[month].\"
\".$f[day].\"<br>"; }

```

## Практикум

### Практическая работа № 1. Основы разметки гипертекста HTML

*Цель работы:*

Изучение основ стандартного языка разметки HTML для создания статических Web-страниц.

*Задание*

1. В текстовом редакторе набрать Web-страницу и сохранить как текст в файле

```

HTMLSimpl.
<html>
    <head> <title> Добро пожаловать в Web - страницу
</title> </head>
    <body align = "center" alink = "tomato" link =
"FFFFF00"> Здесь содержимое страницы </body>
</html>

```

Посмотреть этот файл в Internet Explorer и Netscape Navigator. Находясь в браузере, посмотреть исходный текст.

2. Для Web-страницы HTMLSimpl задать атрибуты тэга `<body>`, описанные в теоретической части ("Основы разметки гипертекста HTML"), и выяснить какие из них поддерживаются браузером Internet Explorer, а какие браузером Netscape Navigator.
3. Для Web-страницы HTMLSimpl изменить цвет фона, используя константы, описанные в теоретической части (п.4.1 "Основы разметки гипертекста HTML"), а также задать фон в виде картинки из файла.
4. С помощью графического редактора создать изображение и вставить его в Web- страницу, используя тэг `<img>` и указав в атрибуте `<src>` этого тэга полный путь файла, содержащего созданное изображение.

В эту же страницу вставить заголовки всех шести уровней и использовать тэг <br/> для размещения текста и изображения на экране. Вставить также текст с несколькими параграфами (абзацами), выделяя каждый параграф тэгами <p> и </p>. Вставить гиперссылку с помощью тэгов <a> и </a>:

```
<a href = "http://www.microsoft.com" Microsoft>
Нажмите на ссылку </a> Вставить также форматированный
текст, используя тэги <pre> и </pre>.
```

5. Изучить назначение атрибутов тэга <a>: accesskey, class, datafld, datasrc, href, id, lang, langvage, methods, name, rel, rev, style, taget, title, vgn.
6. Разместить на Web-странице несколько фрагментов текста и несколько изображений, располагая их последовательно или мозаикой и изменяя атрибуты тэга <font>: class, color, face, id, lang, language, size, style, title, point\_size, weight. Убедиться, что действие тэга <font> прекращаются тэгом </font>.
7. Ввести фрагмент HTML и посмотреть на экране вид таблицы:

```
<table border="1"> <tr align = "center">
<th colspan = "3"> Это таблица </th> </tr>
<tr align = "center">
<td> Это </td> <td> ячейки </td> <td> для </td>
</tr>
<tr align = "center">
<td> данных </td> <td> в </td> <td> таблице </td>
</tr>
</table>
```

8. Создать таблицу, в которой будут использованы все виды списков.
9. Создать статическую Web-страницу по заданной тематике, полученной у преподавателя, разместив на ней все изученные элементы, в том числе все виды списков. Перед каждым новым элементом вставить необходимые пояснения. Проверить работу электронной почты и все виды гиперссылок.
10. Создать Web-страницу (задание п.9) в среде Dremweaver, сравнить коды Web- страниц, полученные в п.9 и в п.10, сделать выводы.
11. Оформить отчет, в котором должен быть представлен HTML-код страницы, и отчитаться преподавателю.

## **Практическая работа № 2. Разработка сценариев Web - страниц**

### *Цель работы:*

Изучение основ программирования на языке JavaScript для создания сценариев, способов внедрения сценариев в Web - страницы, приемов использования свойств, методов и событий, а также их связывания с элементами управления на Web - странице, такими как текстовые поля, кнопки, флажки, переключатели и списки.



### Теоретическая часть

Имя параметра обработки события начинается с приставки on, за которой следует имя события. Рассмотрим некоторые наиболее часто встречающиеся события и элементы документов HTML, в которых эти события могут происходить.

События	Объекты	Когда происходят события?
Abort	image	Отказ от загружаемого изображения
Blur	window, элементы формы	Потеря объектом фокуса
Change	text, textarea, select	Изменение значения элемента
Click	button, radio, checkbox, submit, reset, link	Щелчок на элементе или связи
Error	image, window	Ошибка при загрузке документа или изображения
Focus	window, элементы формы	Окно или элемент формы получения фокуса

Например:

```
<input type="button" value="Вычислить"
onClick="sumball()">
```

В этом случае при нажатии на кнопку «Вычислить» будет вызвана функция sumball(). Основным элементом языка JavaScript является функция, которая имеет следующий формат:

```
function F(V) {S},
```

где F – идентификатор функции, V – список параметров, S – тело функции. Вложенности функций не допускается.

Для создания кода на JavaScript в документе используются тэги <script> и </script>, между которыми вставляется код функций.

*Пример 1.*

Реализация инициализации Web-страницы браузером с использованием JavaScript выглядит следующим образом:

```
<html>
  <head> <title> Инициализация на JavaScript</title>
  <script language="JavaScript">
    function Page_Initialize()
    {document.form1.hitext.value="Привет от JavaScript!";}
  </script></head>
  <body onLoad="Page_Initialize()"> <form
name="form1">
  <center>
    <input type="text" name="hitext" size="25">
  </center></form> </body> </html>
```

В этой реализации использован тэг <form>, который имеет следующие атрибуты: action, class, enctype, id, language, method = get | post, name, style, target, title, событие="сценарий", onreset, onsubmit.

### *Пример 2.*

Установить на Web-странице текстовое поле и кнопку "Показать сообщение", нажатие на которую вызывает появление сообщения в текстовом поле. С использованием JavaScript реализация задания будет такой:

```
<html> <title> Работа с кнопками </title> <body>
<center>
  <form name="form1">
    <input type="text" name="Textbox" size="25">
<br><br>
    <input type="button" value="Показать сообщение"
onClick="ShowMessage(this.form)"> </form> </center>
</body>
  <script language="JavaScript"> function
ShowMessage(form1)
  {document.form1.Textbox.value = "Привет"}
</script></html>
```

### *Пример 3.*

Установить на Web странице 5 флажков с именами "Флажок1" "Флажок2" и т.д., а также текстовое поле для указания вновь выбранного флажка. С использованием VBScript и браузера Internet Explorer данное задание реализуется следующим образом:

```
<html>
<head>
  <title>Разработка сценариев</title> <script
language="javascript"> function Check1Clicked(){
document.getElementById("TextBox").value="Выбран
флажок1!";
}
function Check2Clicked(){
document.getElementById("TextBox").value="Выбран
флажок2!";
}
function Check3Clicked(){
document.getElementById("TextBox").value="Выбран
флажок3!";
}
function Check4Clicked(){
document.getElementById("TextBox").value="Выбран
флажок4!";
}
function Check5Clicked(){
document.getElementById("TextBox").value="Выбран
флажок5!";
}
```

```

</script>
</head><body>
<h1>Выберите флажок</h1>
<table border bgcolor = "cyan" width = "200">
<tr><th><input type="checkbox" name="Check1"
onClick="Check1Clicked()"> Флажок 1 </th></tr>
<tr><th><input type="checkbox" name="Check2"
onClick = "Check2Clicked()"> Флажок 2 </th></tr>
<tr><th><input type="checkbox" name="Check3"
onClick = "Check3Clicked()"> Флажок 3 </th></tr>
<tr><th><input type="checkbox" name="Check4"
onClick="Check4Clicked()"> Флажок4</th></tr>
<tr><th><input type="checkbox" name="Check5"
onClick="Check5Clicked()"> Флажок5</th></tr> </table>
<input type="text" id="TextBox" Size="30">
</body></html>

```

#### *Пример 4.*

Установить на Web-странице раскрывающийся список выбора с пятью опциями, имеющими названиями "Опция1", "Опция2", "Опция3", "Опция4" и "Опция5", а также текстовое поле для отображения выбранной опции. Далее приведены фрагменты страницы:

```

.....
<script>
function SelectClicked(){
document.getElementById("TextBox").value="Вы
выбрали                опцию                "                +
(document.getElementById("Select1").selectedIndex+1);
}
</script>
.....
<select id="Select1" onClick="SelectClicked()">
<option value="1">Опция 1
<option value="2">Опция 2 <option value="3">Опция
3 <option value="4">Опция 4 <option value="5">Опция 5
</select>
<input name="TextBox" type="text" size = "20">

```

#### *Пример 5.*

Метод `confirm()` объекта `window` позволяет вывести на экран диалоговое окно с кнопками "ОК" и "Отмена". От выбора пользователя зависит результат возвращаемого этой функцией значения: `true`, если нажата кнопка ОК ; `false`, если нажата кнопка "Отмена".

Создадим страницу, при запуске которой будет выводиться диалоговое окно с предложением загрузки изображения `img_1.gif`. Если пользователь выбирает "Отмена", то изображение загружено не будет.

```
<html>
  <head> <title> Инициализация на JavaScript</title>
</head>
  <body>
    <script language="JavaScript">
      if(confirm("Загрузить изображение?")==true)
document.write("<img src='img_1.gif'>"); </script>
    </body> </html>
```

Метод write стандартного объекта документа document добавляет страницу строкой "<img src='img\_1.gif'>". Обратите внимание, что внешние кавычки являются двойными, а внутренние-одинарными.

#### *Задание*

1. Установить на вашей Web-странице текстовое поле с помощью тэга <input> и обеспечить его инициализацию при первом открытии Web-страницы браузером. Необходимо реализовать данное задание с использованием JavaScript (см. пример № 1).
2. Установить на Web-странице текстовое поле и кнопку "Показать сообщение", нажатие на которую вызывает появление сообщения в текстовом поле. (см. пример № 2)
3. Установить на Web-странице 5 флажков с именами "Флажок1" "Флажок2" и т.д., а также текстовое поле для указания вновь выбранного флажка. (см. пример № 3)
4. Установить на Web - странице 3 переключателя радио группы с именами "Переключатель 1", "Переключатель 2" и "Переключатель 3", а также текстовое для указания выбранного переключателя. (По аналогии с примером № 3)
5. Установить на Web-странице раскрывающийся список выбора с пятью опциями, имеющими названия "Опция1", "Опция2", "Опция3", "Опция4" и "Опция5", а также текстовое поле для отображения выбранной опции (см. пример № 4).
6. Вставить в страницу вызов функции confirm() для выбора загрузки изображения (см. пример № 5).
7. Изучить назначение функции alert() и привести пример использования на странице.
8. С помощью функции prompt() отобразить диалоговое окно для ввода текста, и отобразить результат ввода на странице.
9. Оформить отчет по лабораторной работе и отчитаться преподавателю.

### **Практическая работа №3. Динамическое изменение Web - страниц**

#### *Цель работы:*

Изучение средств языка JavaScript.

### *Теоретическая часть*

Для динамического изменения страниц, отображаемых в браузерах и для записи HTML - кода непосредственно в WEB - страницу используют метод write() объекта document. Тэг сценария записи должен размещаться сразу после тэга script вне какой-либо подпрограммы. Записываемый HTML - код заключается в кавычки. Если требуется вывести текст уже заключённый в кавычки, то кавычки нужно заменить на одиночные.

Текущую дату и время можно получить с помощью метода toLocaleString() объекта типа Date. Для этого создаётся экземпляр объекта:

```
obDate = new Date();  
stringDate = obDate.toLocaleString();  
document.write(stringDate);
```

В JavaScript условие заключается в круглые скобки, а список операторов вывода - в фигурные скобки. Операция AND обозначается как &&, а операция OR - как ||.

#### *Пример 1.*

Создать WEB - страницу для отображения поздравления в случае, если сегодня праздник - День Знаний.

```
<html>  
<head>  
<title>Динамическое изменение страниц</title>  
</head>  
<body>  
<script language="javascript"> var month=new  
Array();  
var day=new Array();  
/*задаём дату праздника*/ //День Знаний month=8;  
day=1;  
obDate=new Date();  
document.write("Текущая дата и время:  
"+obDate.toLocaleString()+"<br/>" );  
if( (obDate.getMonth()==month) &&  
(obDate.getDate()==day) ){ document.write("1 сентября!  
Ура! Сегодня праздник!");  
}  
else  
document.write("Сегодня нет праздника"); </script>  
</body>  
</html>
```

#### *Пример 2.*

В Internet Explorer каждый HTML - элемент имеет следующие свойства: innerText, outerText, innerHTML, outerHTML.

Их можно использовать для изменения содержимого соответствующего элемента. В частности, свойство innerText позволяет

изменять текст, расположенный между открывающими и закрывающими тэгами элемента, с которыми ведется работа, включая текст любых элементов, которые содержатся в данном элементе. Свойство `outerText` позволяет изменять весь текст данного элемента, включая открывающие и закрывающие тэги. Свойство `innerHTML` дает возможность изменять HTML-код, расположенный между открывающим и закрывающим тэгами элементов. И, наконец, свойство `outerHTML` позволяет изменять весь HTML-код данного элемента, включая его открывающий и закрывающий тэги.

Для выбора и изменения текста следует создать диапазон текста. В Internet Explorer диапазон ограничивает некоторую часть текста на странице и позволяет использовать функции - методы объекта типа диапазон `dim`. Например, требуемый текст в диапазоне ищется с помощью метода `findText()`, а обработку и изменение текста в этом диапазоне производит метод `pasteHTML()`. Сам диапазон создается методом `createTextRange()` и устанавливается оператором `set`.

Создать Web-страницу с кнопкой "Изменить текст", при нажатии на которую строка "Это текст" заменяется на строку "Это новый текст".

```
<html>
  <head> <title> Пример создания диапазона текста
</title> </head> <body><center>
  <input type="button" value="Изменить текст"
onClick="ChangeText()"> <br> <br>Это текст.</center>
</body>
  <script>
    function ChangeText(){
      var r=document.body.createTextRange();
r.findText("Это текст"); r.pasteHTML("Это новый
текст!");
    }
  </script>
</html>
```

### *Пример 3.*

Создать Web-страницу с использованием двух фреймов, причем в первом фрейме должны содержаться виды ресурсов самолетов из аэропорта, а во втором фрейме - расписания для каждого вида рейсов. Данное задание можно реализовать с использованием следующих 4-х файлов:

#### Файл `airport.html`

```
<!-- Это файл airport.html-->
<html> <head> <title> Использование фреймов
</title> </head> <!-- Распределение ширины страницы
между фреймами.--><frameset cols = "40%, 60%">
  <!-- Это 0-й фрейм--><frame src = "menu.htm"> <!--
Это 1-й фрейм-->
```

```

        <frame      src      =      "welcome.htm"      name="display">
</frameset>
</html>

```

#### Файл menu.htm

```

<!-- Файл menu.htm-->
<!-- Он реализует левый, то есть 0-й фрейм-->
<html> <head>
    <script>
        function Morning() {
parent.frames[1].document.open();
parent.frames[1].document.write("<center>");
parent.frames[1].document.write("<table      border='1'
bgcolor='#FFFF00'>");
parent.frames[1].document.write("<tr><th
colspan='2'>Утренние      рейсы</th></tr>");
parent.frames[1].document.write("<tr><th>Петербург</th>
<th>6:30      AM</th></tr>");
parent.frames[1].document.write("</table>");
    parent.frames[1].document.write("</center>");
    parent.frames[1].document.close();
        }
        function Afternoon() {
parent.frames[1].document.open();
parent.frames[1].document.write("<center>");
parent.frames[1].document.write("<table      border='1'
bgcolor='#FFFF00'>");
parent.frames[1].document.write("<tr><th
colspan='2'>Дневные      рейсы</th></tr>");
parent.frames[1].document.write("<tr><th>Орел</th><th>1
2:00      PM</th></tr>");
parent.frames[1].document.write("</table>");
parent.frames[1].document.write("</center>");
parent.frames[1].document.close();
        }
        function Evening() {
parent.frames[1].document.open();
parent.frames[1].document.write("<center>");
parent.frames[1].document.write("<table      border='1'
bgcolor='#FFFF00'>");
parent.frames[1].document.write("<tr><th
colspan='2'>Вечерние      рейсы</th></tr>");
parent.frames[1].document.write("<tr><th>Новороссийск</
th><th>6:30      PM</th></tr>");
parent.frames[1].document.write("</table>");

```

**Файл welcome.htm**

```
<!-- Следующий файл welcome.htm-->
<!-- Он реализует правый, то есть 1-й фрейм-->
<html> <body> <center>
    <h1>Добро      пожаловать      на      аэродром</h1>
</center></body></html>
```

**Файл hours.htm**

```
<!-- Это файл hours.htm--><html>
<body>
    <center><h1>Время работы с 6 до 22 </h1></center>
</body>
</html>
```

**Задание**

1. Включить в Web-страницу реализацию работы со временем (см. пример 1) в соответствии с номером вашего варианта.

№ варианта	Задание
1	В зависимости от времени суток вывести "Доброе утро" "Добрый день" или "Добрый вечер".
2	Определить дату и поздравить с праздником, если он есть.
3	Спросить у пользователя дату его рождения и, если она совпадает с текущей, то вывести поздравления.
4	Первые 10 дней каждого месяца сообщать "Начался новый месяц", следующие 10 дней - "Месяц продолжается", последние 10 дней - "Месяц заканчивается".
5	От 7:00 до 9:00 выдать "Пора вставать", от 10:00 до 17:00 - "Надо работать", от 18:00 до 21:00 - "Можешь отдохнуть".
6	Составить расписание работы магазина (в зависимости от времени - магазин либо работает, либо закрыт на обед, либо не работает вообще; в зависимости даты может не работать по случаю праздника и т.п.)



2. Включить в Web-страницу динамическое обновление данных (пример 2) в соответствии с номером вашего варианта.

№ варианта	Задание
1	Создать кнопку с надписью "Нажмите на кнопку", при нажатии на которую ее название будет меняться на фразу "Вы на меня нажали!"
2	Создать кнопку "Поиск текста", при нажатии на которую будет произведен поиск текста, введенного пользователем в окно редактирования. По результатам поиска выдать либо "Текст найден" либо "Текст не найден"
3	Изменить в заданном тексте слово, введенное пользователем, на слово "замена!"
4	Поменять в заданном тексте все слова "нет" на слово "да"
5	В текстовом поле пользователь задает слово или фразу, которую необходимо найти в документе. Если оно/она найдено(а), то автоматически создается кнопка перехода на первое вхождение в текстовом поле фразы.

3. Включить в Web-страницу реализацию фреймовой структуры (пример 3) в соответствии с номером вашего варианта.

№ варианта	Задание
1	Разработать прототип небольшого телефонного справочника. По фамилии выдавать номер телефона.
2	Разработать прототип справочника предприятий города. То есть по названию выдавать адрес и профиль работы.
3	Разработать прототип справочника по ВУЗам города. По названию - адрес, телефон и названия факультетов.
4	Разработать прототип "редактора", который позволяет открывать одновременно до трех текстовых документов (текст документов задается разработчиком).
5	Разработать фреймовую структуру, которая бы демонстрировала возможности использования имен окон.

#### **Практическая работа №4. Работа с мышью и клавиатурой**

##### *Цель работы:*

Овладение приемами работы с мышью и клавиатурой для выполнения разнообразных манипуляций над текстом страницы.

##### *Теоретическая часть*

Работа с мышью и клавиатурой, а также таблицы стилей для текста реализуются в Internet Explorer и Netscape Navigator совершенно по-разному. И хотя все предлагаемые задания являются общими для обоих браузеров, конкретные их решения будут различными.

При выполнении операций с мышью, производимых над документом, могут возникать следующие события:

- `onMouseOver` - указатель мыши находится в области окна документа;
- `onMouseOut` - указатель мыши находится за пределами области окна документа;
- `onMouseDown` - нажата какая-либо кнопка мыши:
  - `owindow.event.button=1` - левая;
  - `owindow.event.button=2` - правая;
  - `owindow.event.button=4` - средняя;
  - `window.event.x`, `window.event.y` - координаты точки для указателя мыши;
  - `window.event.shiftKey=1` - для Shift, 2- для Ctrl, 4- для Alt;
- `onMouseMove` - указатель мыши перемещается.

Имя обработчика такого события для какого-либо объекта состоит из имени этого объекта, знака подчеркивания и имени события. Затем для некоторых событий могут следовать круглые скобки.

При работе с мышью в Netscape Navigator также используются события `onMouseDown`, `onMouseUp`, `onMouseOver` и `onMouseOut`, но способ их обработки значительно отличается от Internet Explorer. Обработчикам событий здесь передается объект `event`, имеющий следующие атрибуты:

- `type` - тип события;
- `target` - объект, которому было послано сообщение о событии;
- `layerX` - абсцисса указателя относительно слоя, в котором произошло событие;
- `layerY` - ордината указателя;
- `pageX` - абсцисса указателя относительно страницы;
- `pageY` - ордината указателя;
- `screenX`, `screenY` - координаты указателя мыши относительно экрана;
- `which` - ASCII-код нажатой клавиши;
- `modifiers` - модификаторы клавиш: `Alt_Mask`, `Control_Mask`, `Shift_Mask`, `Meta_Mask`;
- `data` - массив строк, в которых содержатся URL- адреса перемещенных объектов при обработке события `onDragDrop`.

#### *Задание*

1. Создать Web-страницу с полем вывода, в котором отображается положение указателя мыши и состояние ее клавиш.
2. Создать произвольную Web- страницу с управляющей кнопкой для выбора всех элементов этой страницы.
3. Создать Web-страницу с четырьмя гиперссылками, размер каждой из которых увеличивается, как только на этой гиперссылке останавливается курсор.
4. Создать Web-страницу, обеспечивающую ввод символов с клавиатуры в Internet Explorer.

5. Создать Web-страницу с заголовком, который подчеркивается, как только на него устанавливается курсор мыши, при этом использовать таблицу стилей.
6. Создать собственный объект Circle, в конструкторе которого будет передаваться значение периметра окружности. Реализовать 2 метода этого объекта: подсчёт длины окружности и площади круга.

### **Практическая работа №5. Дизайн сайта**

*Цель работы:*

Овладение навыками проектирования веб-сайта

*Задание*

1. Согласовать с преподавателем тему сайта.
2. Продумать модель и структуру сайта.
3. Произвести разметку сайта модель и структуру сайта.
4. Разработать цветовую схему сайта.
5. Наполнить сайт контентом(содержанием).
6. Показать сайт преподавателю и сдать отчет.

### **Практическая работа №6. Основы программирования на языке**

#### **PHP**

*Цель работы:*

Ознакомление с синтаксисом и возможностями языка PHP.

*Пример*

Создать файл \*.php демонстрирующий интеграцию кода html с php

```
<?
//Присвоить значения нескольким переменным
$site_title = "PHP Recipes";
$bg_color = "white";
$user_name = "Chef Luigi";
?>
<html>
<head>
<title><? print $site_title; ?></title>
</head>
<body bgcolor="<? print $bg_color; ?>" > <?
//Вывести приветствие пользователю print "Hello,
".$user_name;
?>
</body>
</html>
```

*Задание*

1. Набрать код примера 1. Изменить название страницы, фоновый цвет и имя пользователя.
2. Продемонстрировать работу операторов if, for, while, foreach.
3. Реализовать передачу данных из html-формы в php-скрипт. Скрипт обрабатывает данные и возвращает ответ.

## **Практическая работа №7. Работа с MySQL через PHP**

### *Цель работы:*

Овладение приемами работы с базой данных MySQL и отображение данных таблиц на веб-странице.

### *Задание*

1. С помощью phpMyAdmin создать новую базу данных и таблицу.
2. Занести несколько записей в таблицу
3. С помощью PHP отобразить все записи таблицы.
4. Осуществить выборку данных по какому-либо критерию(фильтру)
5. Реализовать параметрический запрос(значение параметра определяется выпадающим списком <select>)

Все ответы от MySQL отображать на странице в виде таблиц с заголовками отобранных полей(использовать тэг <table>)

6. Оформить и сдать отчёт.

## **Практическая работа № 8. Установка и настройка сервера IIS**

### *Теоретические сведения*

#### *Назначение и применение IIS*

IIS (Internet Information Services) – распространяемый вместе с операционными системами семейства Windows NT набор серверов для Интернет-служб компании Microsoft. IIS поддерживает сетевые протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP. Последняя версия IIS 7 поставляется вместе операционными системами Windows 7/2008/8/2012.

Для того чтобы выяснить, установлен IIS на компьютере или нет, необходимо просмотреть окно «Компоненты Windows», которое может быть получено через раздел «Программы» пользовательского интерфейса «Панель управления» операционной системы Windows (рис. 1.1).



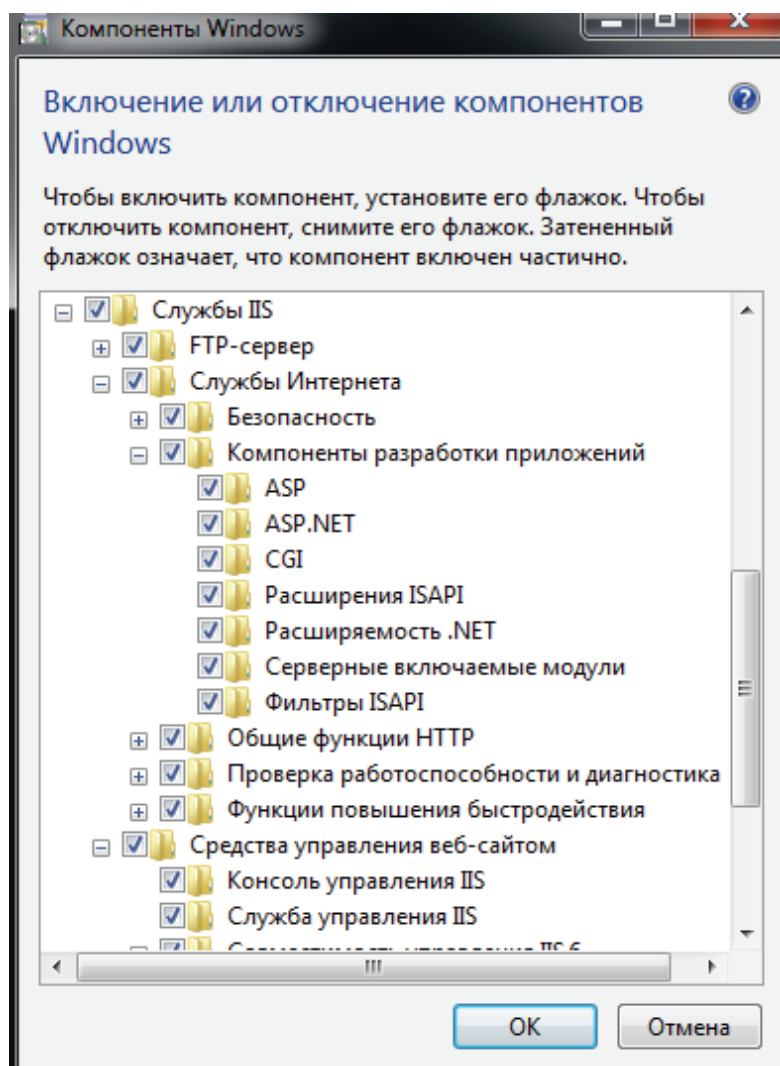


Рисунок 2 Содержимое узла «Компоненты разработки приложений»

Составной частью IIS является приложение IIS Manager, предназначенное для управления IIS. IIS Manager может быть запущен через команду строку (рис. 1.3). На рис. 1.4 представлено стартовое окно IIS Manager. Приложение позволяет подключаться к локальному или удаленным серверам IIS, добавлять, настраивать и удалять сайты, импортировать или экспортировать приложения и т. п. Более подробно ознакомиться с возможностями приложения IIS Manager можно в источник.

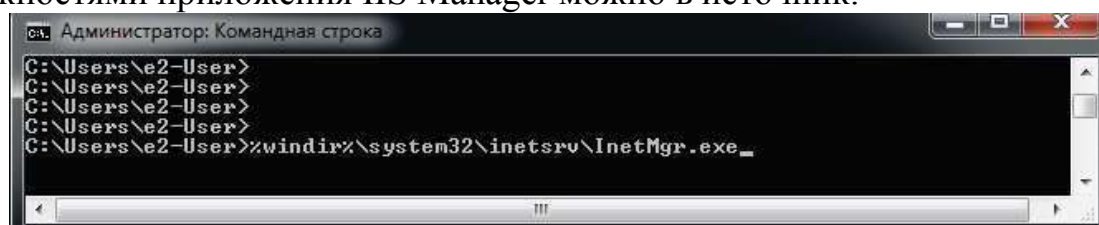


Рисунок 3 Запуск приложения IIS Manager

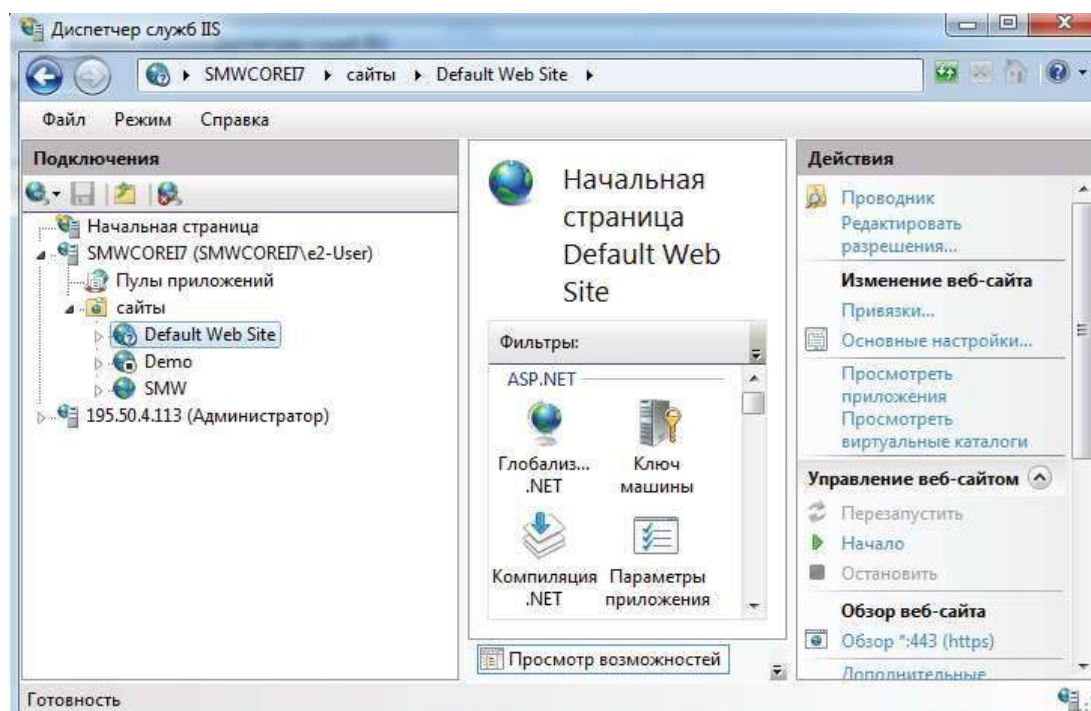


Рисунок 4 Стартовое окно приложения IIS Manager

### Установка и настройка IIS

Установка IIS может быть выполнена как в процессе инсталляции операционной системы, так и после нее. В любом случае, все сводится к выбору пунктов меню в окне «Компоненты Windows» (рис. 1.1 и 1.2). В том случае, если установка и настройка IIS осуществляется уже в рамках установленной операционной системы Windows, доступ к этому окну возможен в следующей последовательности: Панель управления| Программы| Программы и компоненты| Включение и отключение компонентов Windows.

После того как в окне «Компоненты Windows» были выбраны и установлены необходимые программные компоненты сервера IIS, рекомендуется выполнить перезагрузку операционной системы.

После того как Windows с новыми программными компонентами будет загружен, следует проверить работоспособность приложения IIS Manager (см. п. 1.1) и выполнить HTTP-запрос к предустановленному (для проверки работоспособности IIS) сайту. Для этого следует в адресной строке браузера набрать `http://localhost`. В том случае, если установка IIS прошла успешно, в окне браузера отобразится web-страница, подобная той, что представлена на рис. 1.5.



Рисунок 5 Проверка работоспособности IIS 7

### *Технология ASP.NET*

Технология ASP.NET (Active Server Pages в среде .NET) является развитием ей предшествующей технологии ASP, часто называемой классической ASP. ASP.NET – технология создания web-приложений и web-сервисов. На данный момент последней версией этой технологии является ASP.NET 4.5.

ASP.NET является частью программной платформы Microsoft .NET и, кроме того, неразрывно связана с двумя другими продуктами Microsoft: web-сервером IIS и интегрированной средой разработки Visual Studio. IIS для ASP.NET-приложения выступает в качестве оболочки (среды выполнения), основным назначением которой является прием и отправка HTTP-сообщений. Visual Studio 2008/2010/2012 представляет собой инструментальное средство, позволяющее разрабатывать, выполнять отладку и публиковать на сервер приложения ASP.NET.

Технология ASP.NET предоставляет возможность разрабатывать два типа приложений: web-приложения и web-сервисы.

Web-приложения – приложения, обеспечивающие работу вебсайтов. В большинстве своем, работа web-приложений сводится к приему http-запросов, обработке (иногда достаточно сложной) и формированию http-ответов. При этом запросы, как правило, поступают от браузеров, которые отображают пользовательский интерфейс приложения с помощью html-



страниц. Совокупность страниц, которые могут быть пересланы сервером браузеру для отображения, называют сайтом. Следует отметить, что для разработки сайтов Visual Studio представляет два вида web-приложений: Web Forms и MVC.

Web Forms – это вид web-приложения (ему соответствует свой шаблон проекта в Visual Studio), в основе которого лежит механизм ViewState, позволяющий сохранить состояние элементов управления web-страницы. Web Forms применяется для разработки не очень больших сайтов, не требующих большого числа страниц и глубокой их иерархии.

В основе создания MVC-приложения лежит концепция разделения web-приложения на компоненты, отвечающие за представление данных (model), отображение данных (view) и управление (controller). В состав ASP.NET входит набор библиотек (называемый ASP.NET MVC Framework), обеспечивающих поддержку разработки MVC приложений. Как правило, приложения этого вида разрабатываются в том случае, если необходимо разработать большой сайт, требующий высокого уровня масштабирования. На сегодняшний день последней действующей версией является ASP.NET MVC 4.0.

Web-сервис – это особый тип приложения, представляющего собой удаленный (размещаемый на серверных компьютерах) программный компонент, для доступа к методам которого применяется протокол SOAP. Методы web-сервиса могут быть вызваны из .NET приложения способом, практически не отличающимся от вызова метода обыкновенного (локального) программного объекта.

В настоящее время компания Microsoft разработала новую технологию – Windows Communication Foundation (WCF), обобщающую понятие web-сервиса.

В рамках данного пособия рассматриваются web-приложения вида Web Forms.

### *Задания*

#### *Задание 1. Установка и настройка IIS:*

1. добейтесь отображения на мониторе компьютера окна «Компоненты Windows» (рис. 1.1, 1.2);
2. позиционируйте дерево настроек в окне «Компоненты Windows» таким образом, чтобы были видны первоначальные настройки пункта «службы IIS»;
3. получите скриншот окна «Компоненты Windows» с первоначальными настройками служб IIS и включите его в отчет по контрольной работе;
4. отметьте все необходимые компоненты IIS и выполните их установку;
5. перезагрузите операционную систему.

#### *Задание 2. Проверка работоспособности IIS:*

1. с помощью командной строки запустите приложение IIS Manager (рис. 1.3) и убедитесь, что на мониторе отобразилось окно, аналогичное представленному на рис.1.4;

2. получите скриншот окна «Диспетчер служб IIS» и включите его в отчет по контрольной работе;
3. запустите браузер, в адресной строке введите `http://localhost`; убедитесь, что в браузере отобразилось окно, аналогичное представленному на рис. 1.5;
4. получите скриншот окна браузера со стартовой страницей IIS 7 и включите его в отчет по контрольной работе.

### *Задание 3. Контрольные вопросы:*

1. поясните термины: «сервер», «клиент», «браузер», «HTTP», «HTML», «web-приложение»;
2. поясните термины: «IIS», «IIS Manager» «ASP.NET», «Web Forms», «MVC»;
3. поясните термины: «SOAP», «web-сервис», «WCF».

## **Практическая работа № 9. Разработка простейшего приложения Web Forms ASP.NET**

### *Теоретические сведения*

### *Создание приложения Web Forms ASP.NET*

Для создания web-приложения Web Forms ASP.NET необходимо с помощью Visual Studio создать проект соответствующего типа. На рис. 2.1 представлено окно Visual Studio 2010 для создания web-проекта.

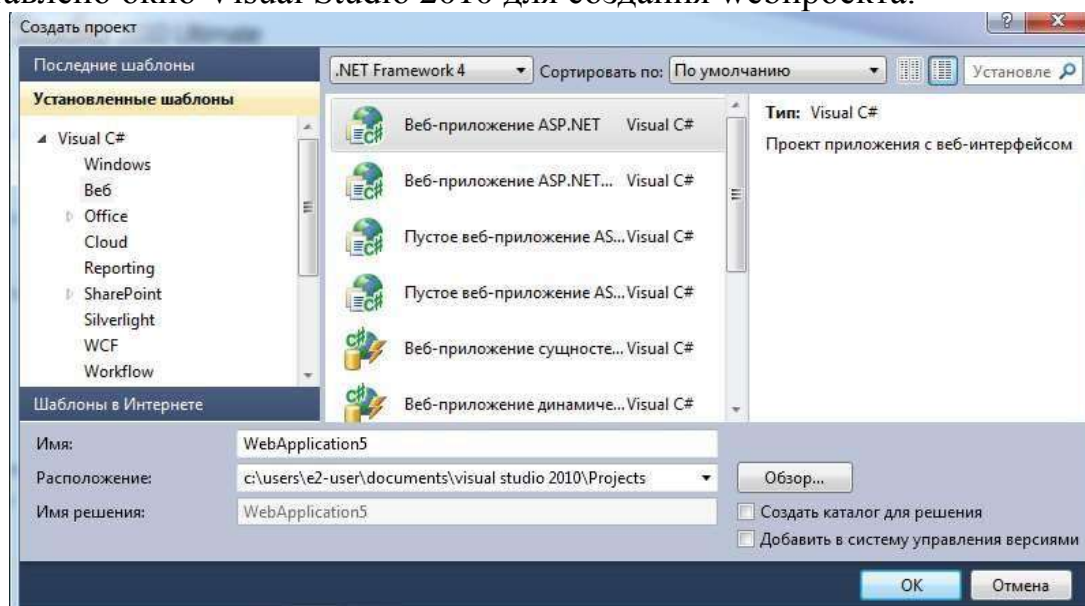


Рисунок 6 Окно Visual Studio 2010 для создания web-проекта

Окно позволяет выбрать шаблон проекта в категории «Веб». Первый сверху шаблон «Веб-приложение ASP.NET» позволяет создать приложение Web Forms. При нажатии клавиши «ОК» создается шаблонное приложение, которое может быть выполнено в отладочном режиме.

На рис. 2.2 представлена страница, отображаемая шаблонным приложением в окне браузера при запуске в отладочном режиме.

# Мое приложение ASP.NET

Домашняя

О программе

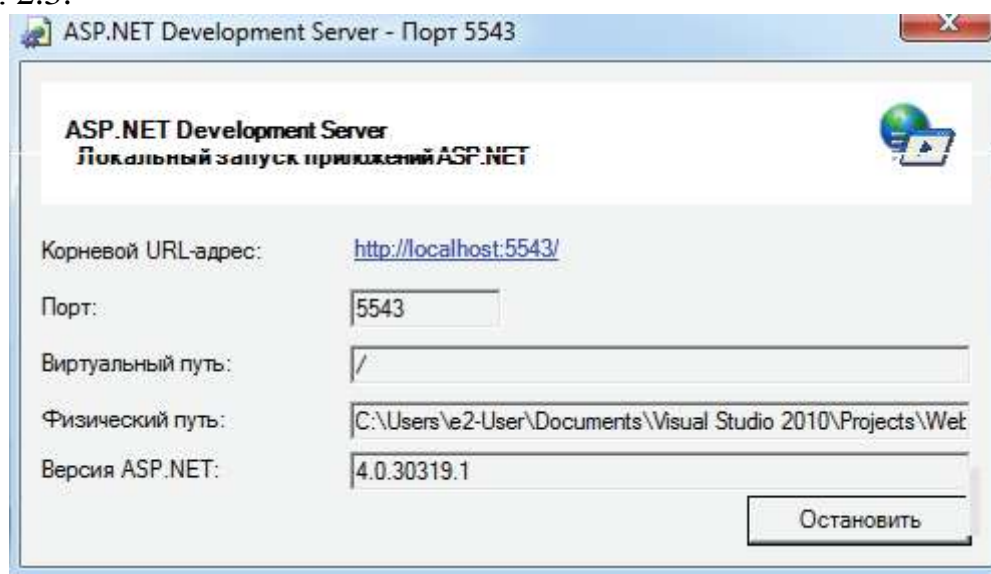
## ДОБРО ПОЖАЛОВАТЬ в ASP.NET!

Для получения дополнительных сведений об ASP.NET посетите веб-сайт [www.asp.net](http://www.asp.net).

Кроме того, [документация по ASP.NET доступна на MSDN](#).

*Рисунок 7 Стартовая страница шаблонного приложения*

При разработке и отладке ASP.NET-приложения разработчику требуется многократно запускать это приложение в режиме отладки. Для этого в рамках Visual Studio существует специальный компонент, называемый ASP.NET Development Server. Сервер стартует автоматически при запуске разрабатываемого web-приложения на выполнение спомощью Visual Studio. Находится сервер в области уведомлений (system tray) операционной системы. При двойном нажатии левой клавишей мыши по значку сервера, на мониторе отобразится окно, аналогичное представленному на рис. 2.3.



*Рисунок 8 Окно ASP.NET Development Server*

## Публикация web- приложения на IIS

После того как web-приложение разработано, оно может быть опубликовано на сервере IIS. Процесс публикации заключается в переносе папок и файлов проекта web-приложения Visual Studio в папку сервера IIS.

Публикация может быть выполнена несколькими способами.

В простейшем случае ее можно осуществить простым копированием папок и файлов. Для этого следует выполнить следующую последовательность действий.

1. С помощью приложения IIS Manager создать (добавить) новый сайт. На рис. 2.4 представлено окно создания нового сайта. При этом необходимо

указать имя сайта, выбрать пул приложений (система общих настроек для web-приложений), физический путь для размещения папок и файлов web-приложения, а также установить привязки (тип протокола, номер TCP-порта и IP-адрес, который будет прослушивать сайт).

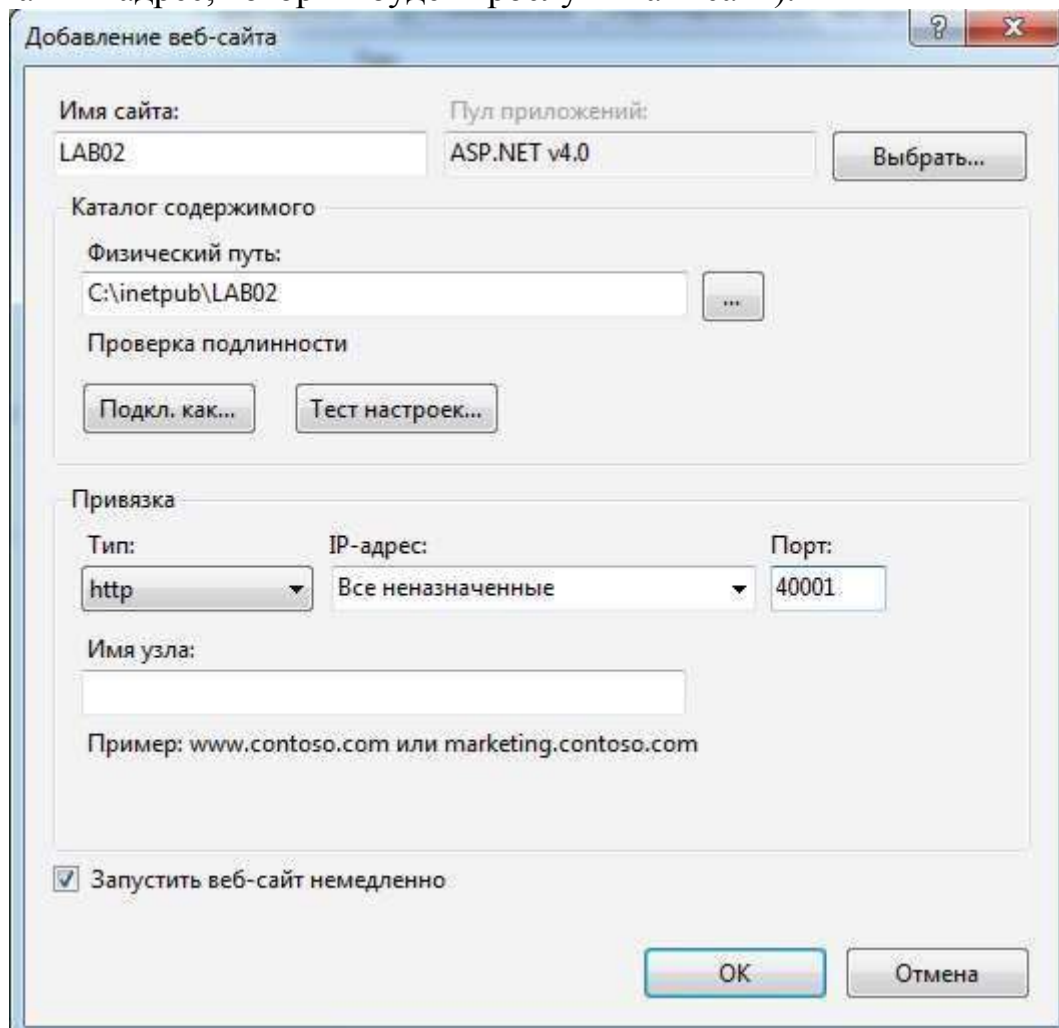


Рисунок 9 Окно ASP.NET Development Server

2. Скопировать папки и файлы проекта web-приложения из папок Visual Studio в папку созданного на IIS сайта.
3. Проверить работоспособность сайта. Для этого следует на компьютере с установленным сервером IIS запустить браузер и в адресной строке указать `http://localhost:40001`, где 40001 – номер порта, указанный при создании сайта (рис. 2.5).

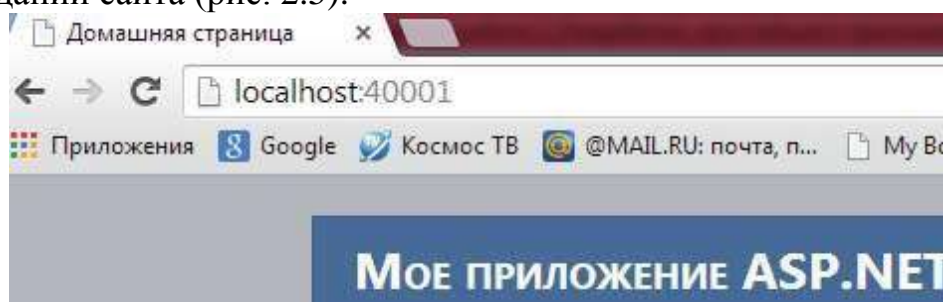


Рисунок 10 Вызов приложения с помощью браузера

4. Удалить лишние файлы из папки сайта. После удаления лишних файлов (исходных кодов) проекта, папка сайта будет выглядеть примерно так, как представлено на рис. 2.6.

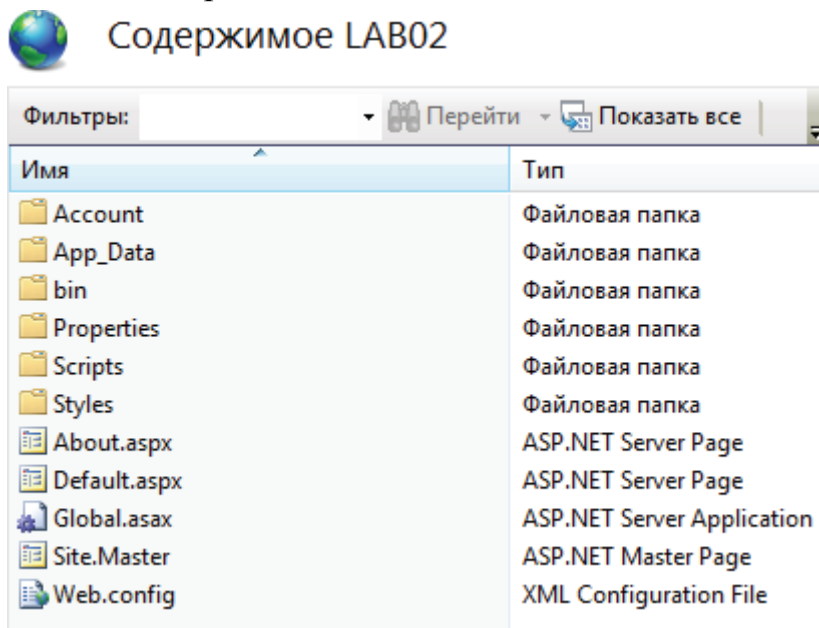


Рисунок 11 Содержимое физической папки сайта сервера IIS

Другие способы публикации требуют установки дополнительного пакета, который может быть получен с сайта Microsoft Download Center (<http://www.microsoft.com/ru-ru/download>). После установки этого пакета появляется возможность публиковать web-приложения еще несколькими способами. В любом случае требуется, чтобы на IIS был предварительно создан сайт, в папку которого будут копироваться файлы приложения.

С помощью контекстного меню (рис. 2.7) может быть создан пакет развертывания.

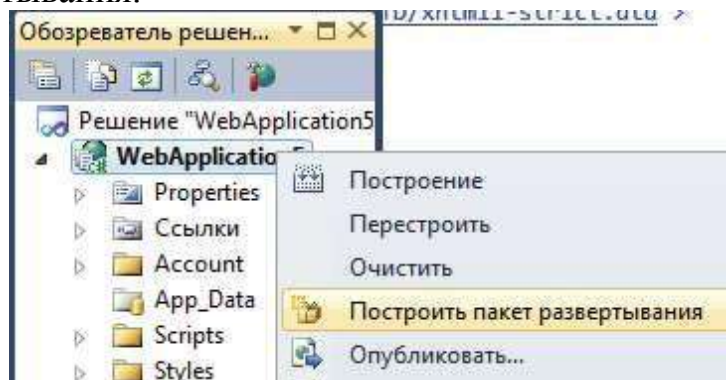


Рисунок 12 Построение пакета развертывания

Пакет формируется в папке `obj\Debug\Package` проекта и представляет собой файл с расширением `zip`, имя которого совпадает с именем проекта. Этот файл используется для импорта web-приложения на сервер IIS. Импорт может быть выполнен с помощью контекстного меню приложения IIS Manager (рис. 2.8).



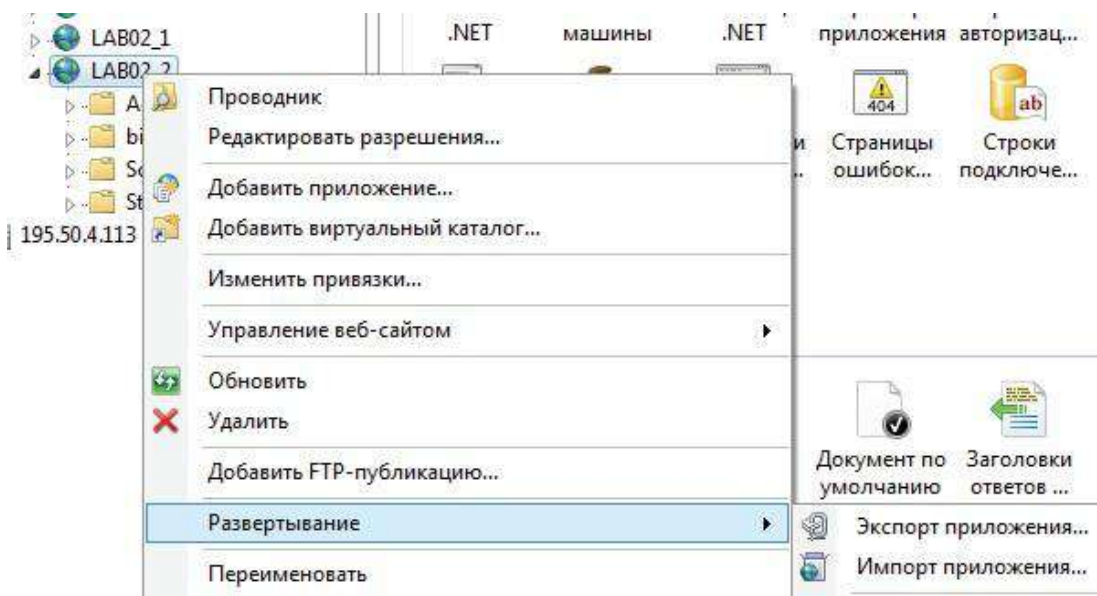


Рисунок 13 Импорт приложения с помощью приложения IIS Manager

Успешное завершение импорта web-приложения равносильно его публикации.

#### Задания

##### Задание 4. Создание простейшего приложения Web Forms ASP.NET:

5. запустите Visual Studio и создайте шаблонное приложение Web Forms ASP.NET;
6. сделайте скриншот окна «Обозреватель решений» проекта и поместите его в отчет по контрольной работе;
7. откройте в проекте файл Default.aspx и замените текст «Добро пожаловать в ASP.NET!» на текст, содержащий ваше имя, фамилию и отчество;
8. запустите созданное web-приложение на выполнение в отладочном режиме;
9. сделайте скриншот окна браузера, отображающего стартовую страницу приложения, и поместите его в отчет по контрольной работе;
10. откройте окно отладочного сервера ASP.NET Development Server, сделайте скриншот и поместите его в отчет по контрольной работе.

##### Задание 5. Публикация приложения на IIS:

1. запустите приложение IIS Manager и с его помощью создайте сайт;
2. опубликуйте в этот сайт созданное в задании 4 приложение методом простого копирования файлов;
3. убедитесь в работоспособности сайта;
4. с помощью приложения IIS Manager создайте еще один сайт;
5. опубликуйте в последний созданный сайт созданное в задании 4 приложение методом отличным от простого копирования файлов;
6. сделайте скриншот окна приложения IIS Manager, который бы демонстрировал наличие двух созданных в этом задании сайтов; поместите скриншот в отчет по контрольной работе.

*Задание 6. Контрольные вопросы:*

1. поясните термины: «шаблон приложения», «ASP.NET Development Server»;
2. поясните термины: «сайт», «публикация приложения»;
3. что такое пакет развертывания приложения, как он может быть сформирован и как применен?
4. перечислите параметры, которые необходимо указать при создании сайта с помощью приложения IIS Manager;
5. перечислите известные вам способы публикации web-приложений.

**Практическая работа № 10. Исследование структуры приложения Web Forms ASP.NET**

*Теоретические сведения*

*Структура приложения Web Forms ASP.NET*

Структура web-приложения в полной мере отображается в окне «Обозреватель решений» Visual Studio (рис. 3.1).

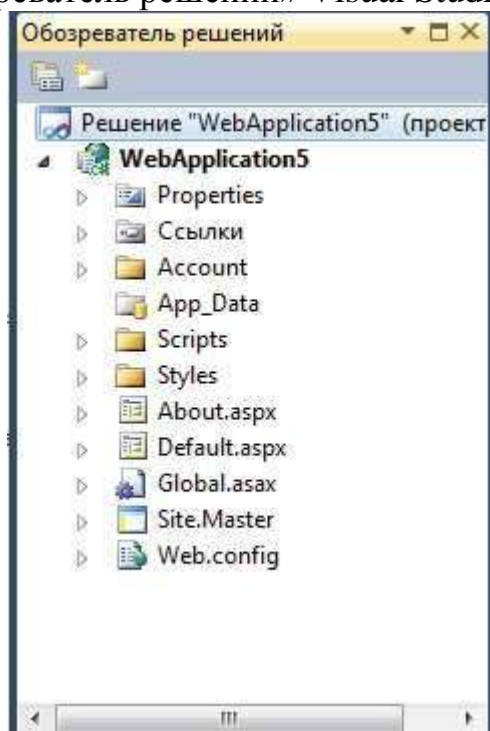


Рисунок 14 Окно «Обозреватель решений» Visual Studio

Приложение состоит из нескольких компонентов, представляющих собой файлы, расположенные в папках со стандартными именами.

Файлы с расширением aspx – web-страницы ASP.NET. На рис. 3.2 представлена стартовая страница Default.aspx шаблонного приложения.

```

<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="WebApplication5._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent"></asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
  <h2> Добро пожаловать в ASP.NET! </h2>
  <p>
    Для получения дополнительных сведений об ASP.NET посетите веб-сайт
    <a href="http://www.asp.net" title="Веб-сайт ASP.NET">www.asp.net</a>.
  </p>
  <p>
    Кроме того, <a href="http://go.microsoft.com/fwlink/?LinkID=152368"
    title="Документация по ASP.NET на MSDN">документация по ASP.NET доступна на MSDN</a>.
  </p>
</asp:Content>

```

Рисунок 15 Файл Default.aspx

Файлы с расширением aspx содержат разметку web-формы. Для разметки могут использоваться теги языка HTML и специальные aspx-теги, представляющие элементы управления ASP.NET. Все aspx страницы начинаются с директивы Page, атрибуты которой содержат основные параметры страницы.

Файлы с расширением master называются мастер-страницами. Мастер страницы представляют собой шаблоны, позволяющие создавать множество web-форм с одинаковой структурой. На рис. 3.2 атрибут Master PageFile директивы Page указывает имя файла, содержащего мастер-страницу, которая используется при отображении формы.

Файлы с расширением cs – исходные коды приложения. Как правило, эти файлы связаны с aspx-страницами или другими компонентами web-приложения. Некоторые cs-файлы генерируются Visual Studio автоматически. Они, как правило, имеют расширение designer.cs.

Файл Global.asax содержит код обработчиков событий, реагирующих на глобальные события приложения. Разработчик может написать код, который ASP.NET будет выполнять при запуске приложения при его завершении, при возникновении необработанной ошибки и т. п.

Файл Web.config – конфигурационный файл. Файл содержит набор параметров (в формате XML) уровня приложения, отвечающие за настройку всех аспектов выполнения приложения, начиная с безопасности и заканчивая отладкой и управлением состоянием.

В таблице описано назначение основных папок приложения Web Forms ASP.NET.

Таблица 1 Назначение папок приложения Web Forms ASP.NET

Наименование папки	Назначение
App_Data	Используется для хранения данных, включая файлы SQL Server Express и файлы XML
Scripts	Используется для хранения JavaScript скриптов (js-файлов), применяемых в web-формах приложения
Styles	Используется для хранения каскадных таблиц стилей (css-файлов)
Account	Часть шаблонного приложения (css-файлы), позволяющая



	организовать аутентификацию и авторизацию пользователей сайта
Bin	Все скомпилированные сборки .NET приложения

Кроме перечисленных типов файлов и стандартных папок, ASP.NET-приложение может содержать и другие компоненты. С полным перечнем компонентов ASP.NET-приложения можно ознакомиться в источнике.

#### *Модель событий приложения Web Forms ASP.NET*

Разработка web-приложения – это, по сути, разработка обработчиков событий. Код, разрабатываемый программистом, получает управление от ASP.NET только в результате какого-то события, возникшего в результате действий пользователя, приведших к отправке на сервер http-запроса или события, возникшего на стороне сервера.

Важно понимать, что каждый http-запрос, поступающий со стороны сервера начитает новый цикл жизни web-приложения. Говорят, что приложение не помнит своего состояния. Другими словами, если в программе установить значение некоторой переменной, то это не значит, что в следующем цикле обработки (при последующем запросе) эта переменная будет иметь установленное ранее значение.

В большинстве случаев разработчик программирует обработку событий двух типов: событий Global.asax и событий aspx-страницы.

На рис. 3.3 приведен код класса Global, методы которого являются обработчиками событий Global.asax. В комментариях перечислены события приложения, при которых выполняется код каждого обработчика.

```
namespace WebApplication5
{
    public class Global : System.Web.HttpApplication
    {

        void Application_Start(object sender, EventArgs e)
        { /*Код, выполняемый при запуске приложения*/}

        void Application_End(object sender, EventArgs e)
        { /*Код, выполняемый при завершении работы приложения */}

        void Application_Error(object sender, EventArgs e)
        { /* Код, выполняемый при возникновении необрабатываемой ошибки */}

        void Session_Start(object sender, EventArgs e)
        { /* Код, выполняемый при запуске нового сеанса */ }

    }
}
```

Рисунок 16 Обработчики событий Global.asax

События aspx-страницы отражают ее жизненный цикл. На рис. 3.4 представлен код класса \_Default, содержащий методы-обработчики событий страницы. Комментарии поясняют последовательность вызова обработчиков и события, которые они обрабатывают. Третьими по очереди осуществляется обработка событий элементов управления, которые инкапсулируются aspx-

страницей. Количество таких обработчиков зависит от элементов управления, расположенных на странице, и событий, которые они могут генерировать.

```
namespace WebApplication5
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Init(object sender, EventArgs e)
        { /* 1.вызывается при инициализации страницы */ }

        protected void Page_Load(object sender, EventArgs e)
        { /* 2. вызывается при загрузке страницы */ }

        // 3. обработчики событий
        // элементов управления
        // расположенных на этой странице

        protected void Page_PreRender(object sender, EventArgs e)
        { /* 4. вызывается перед формированием HTML-кода */ }

        protected void Page_Unload(object sender, EventArgs e)
        { /* 5. вызывается после отправки страницы */ }
    }
}
```

Рисунок 17 Обработчики событий Global.asax

Следует отметить, что с точки зрения ASP.NET aspx-страница является таким же элементом управления как, скажем, клавиша. Просто арх-страница – это элемент управления, который «умеет» инкапсулировать другие элементы. Поэтому любой элемент управления имеет ту же модель событий, что и aspx-страница.

### Задания

*Задание 7. Исследование структуры приложения и модели событий Web Forms ASP.NET:*

1. создайте приложение Web Forms ASP.NET с помощью Visual Studio;
2. измените страницу Default.aspx так, чтобы она была такой же, как на рис. 3.4;

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="WebApplication5._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent"></asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2> Практическая работа № 3 </h2>
    <p>
        <br /><br />
        <asp:Button ID="Button1" runat="server" Text="Button1"
            onclick="Button1_Click" />
        <asp:Button ID="Button2" runat="server" Text="Button2" style="margin-left: 34px"
            onclick="Button2_Click" />
        <br /><br />
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </p>
</asp:Content>
```

Рисунок 18 Файл Default.aspx

3. откорректируйте программный код в файле Default.aspx.cs, чтобы он стал таким же, как на рис. 3.6; отличаться может имя пространства имен

(указывается в операторе namespace), которое в шаблонном приложении совпадает с именем приложения, задаваемым при создании;

```
namespace WebApplication5
{
    public partial class _Default : System.Web.UI.Page
    {
        private int k = 0;
        protected void Page_Init(object sender, EventArgs e)
        { this.Label1.Text += "Init[" + (++k).ToString() + "]-"; }

        protected void Page_Load(object sender, EventArgs e)
        { this.Label1.Text += "Load[" + (++k).ToString() + "]-"; }

        protected void Button1_Click(object sender, EventArgs e)
        { this.Label1.Text += "Click1[" + (++k).ToString() + "]-"; }

        protected void Button2_Click(object sender, EventArgs e)
        { this.Label1.Text += "Click2[" + (++k).ToString() + "]-"; }

        protected void Page_PreRender(object sender, EventArgs e)
        { this.Label1.Text += "PreRender[" + (++k).ToString() + "]-"; }

        protected void Page_Unload(object sender, EventArgs e)
        { this.Label1.Text += "Unload[" + (++k).ToString() + "]-"; }
    }
}
```

Рисунок 19 Файл Default.aspx.cs

4. запустите приложение на выполнение в отладочном режиме; убедитесь, что в окне браузера отображается форма такая же, как на рис. 3.7;

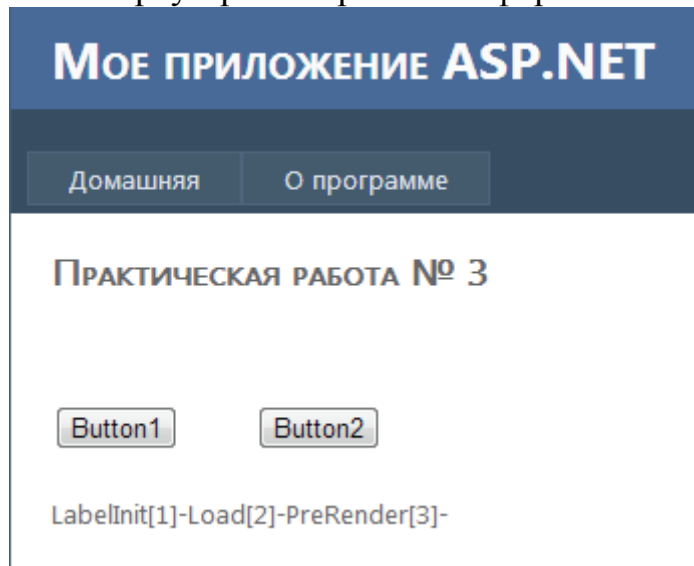


Рисунок 20 Отображение формы Default.aspx в окне браузера

1. сделайте скриншот окна браузера, отображающего страницу Default.aspx приложения и поместите его в отчет по контрольной работе;
2. с помощью мыши нажмите клавишу Button1; обратите внимание на изменение текста строки, расположенной под кнопками;
3. сделайте скриншот окна браузера, отображающего страницу Default.aspx приложения и поместите его в отчет по контрольной работе;

4. с помощью мыши нажмите клавишу Button2; обратите внимание на изменение текста строки, расположенной под кнопками;
5. сделайте скриншот окна браузера, отображающего страницу Default.aspx приложения и поместите его в отчет по контрольной работе;
6. опубликуйте приложение на IIS и убедитесь в его работоспособности.

*Задание 8. Контрольные вопросы:*

1. перечислите все типы файлов, которые могут входить в приложение Web Forms ASP.NET, поясните их назначение;
2. перечислите все наименования стандартных папок приложения Web Forms ASP.NET, поясните их назначение;
3. перечислите в порядке их появления все события, которые могут быть обработаны методами класса Global (файл Global.asax.cs);
4. перечислите все события страницы в порядке их появления;
5. объясните принцип формирования строки на форме приложения, разработанного в задании 7.

**Практическая работа № 11. Разработка и применение HTTP-обработчика ASP.NET**

*Теоретические сведения*

*Разработка HTTP-обработчика ASP.NET*

Каждый поступающий в приложение ASP.NET запрос предварительно обрабатывается специально предназначенным для этого компонентом, который называется обработчиком HTTP. Обработчики создаются для каждого типа файлов. В состав ASP.NET входит несколько стандартных обработчиков. Например, для обработки файлов с расширениями aspx, asmx, ashx и т.д.

Сопоставление обработчиков расширениям можно увидеть с помощью приложения IIS Manager. Для этого следует выбрать с помощью курсора мыши сайт, а затем сделать двойной щелчок на иконке с надписью «сопоставление обработчиков» (рис. 4.1).



## Сопоставления обработчиков

Эта функция предназначена для указания ресурсов (библиотек DLL и управляемого кода), которые обрабатывают определенные типы запросов.

Сгруппировать по: Состояние				
Имя	Путь	Со...	Тип пути	Обработчик
TRACEVerbHandler	*	Вк...	Не указывается	ProtocolSupportM...
TraceHandler-Integrated-4.0	trace.axd	Вк...	Не указывается	System.Web.Handl...
TraceHandler-Integrated	trace.axd	Вк...	Не указывается	System.Web.Handl...
svc-ISAPI-4.0_64bit	*.svc	Вк...	Не указывается	IsapiModule
svc-ISAPI-4.0_32bit	*.svc	Вк...	Не указывается	IsapiModule
svc-Integrated-4.0	*.svc	Вк...	Не указывается	System.ServiceMoc...
StaticFile	*	Вк...	Файл или кат...	StaticFileModule, D...
SSINC-stm	*.stm	Вк...	Файл	ServerSideIncludeV...
SSINC-shtml	*.shtml	Вк...	Файл	ServerSideIncludeV...

Рисунок 21 Отображение обработчиков в приложении IIS Manager

При выполнении приложения Web Forms ASP.NET в командной строке запрашивается страница с расширением aspx. ASP.NET обрабатывает этот запрос с помощью стандартного (входящего в состав ASP.NET) HTTP-обработчика, который и управляет всем жизненным циклом приложения.

Разработка HTTP-обработчиков считается низкоуровневым программированием ASP.NET и применяется для решения специфических задач: например, для скачивания с сервера файла, считывания видео или аудио потоков.

С точки зрения программиста HTTP-обработчик — это класс, реализующий интерфейс IHttpHandler. На рис. 4.2 представлен пример простейшего HTTP-обработчика.

```

public class FirstHandler : IHttpHandler           // HTTP-обработчик
{
    public bool IsReusable { get { return true; } } // можно повторно использовать обработчик ?
    public void ProcessRequest(HttpContext context) // обработка запроса
    {
        HttpResponse response = context.Response; // объект ОТВЕТ
        HttpRequest request = context.Request;    // объект ЗАПРОС
        string parm = (request.Params["parm"] == null? "null":request.Params["parm"]); // параметр есть ?
        response.Write("Http-обработчик FirstHandler, parm=" + parm); // вывод на форму
    }
}

```

Рисунок 22 Пример HTTP-обработчика

Интерфейс IHttpHandler предполагает реализацию свойства IsReusable и метода ProcessRequest.

Свойство IsReusable принимает два логических значения: истина или ложь. Истинное значение указывает на возможность повторного применения экземпляра класса обработчика в другом потоке. В альтернативном случае для каждого потока будет создаваться собственный экземпляр класса обработчика.

Связь классов-обработчиков с методами HTTP-запроса и расширениями запрашиваемой страницы описывается в разделе `handlers` конфигурационного файла `Web.config` приложения (рис. 4.3).

```
<system.webServer>
  <modules runAllManagedModulesForAllRequests="true" />

  <handlers>

    <add verb="GET" path="*.fotproject" type="Lib4.FirstHandler" name="GETProject" />

    <add verb="POST" path="*.fotvalidate" type="Lib4.TwoHandler" name="POSTValidate" />

  </handlers>

</system.webServer>
```

Рисунок 23 Фрагмент файла `Web.config`, описывающий HTTP-обработчики

Два тега `add` описывают два HTTP-обработчика, реализованных как классы `FirstHandler` и `TwoHandler`, расположенные в пространстве имен `Lib4` (атрибут `type`). Для каждого обработчика указывается HTTP-метод (`verb`), расширение, запрашиваемой страницы (`patch`), а также имя (`name`), отображаемое приложением `IIS Manager` в окне «Сопоставление обработчиков» (рис. 4.1).

*Разработка HTTP-клиента, взаимодействующего с HTTP-обработчиком ASP.NET*

В простейшем случае GET-запрос HTTP-обработчику может быть отправлен с помощью браузера. Для этого достаточно в адресной строке браузера правильно набрать соответствующий обработчику URL. Для формирования POST-запроса, следует разработать web форму, содержащую тег `form` со значением `POST` в атрибуте `method` и правильным URL в атрибуте `action`. На рис. 4.4 приведен пример web-формы, формирующая POST-запрос.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title> HTTP-обработчик KRAS_12_FOT</title></head>
<body>
  <form id="f1" action= "http://smwcorei7:40007/FOT.fotvalidate"
    method = "post" enctype = "multipart/form-data" >
    <br /> <input type="file" id="FOTFile" name="FotFile"
      style="border: 3px solid; width: 400px;" />
    <input id="Submit1" type="submit" value="отправить" />
  </form>
</body>
</html>
```

Рисунок 24 Web-форма, формирующая POST-запрос

В том случае, если запрос необходимо сформировать в приложении на языке `C#` следует воспользоваться возможностями предоставляемыми встроенным классом `System.Net.WebClient`. На рис. 4.5. представлен пример применения класса `WebClient` для формирования POST-запроса,



пересылающего серверу (HTTP-обработчику) файл с клиентского компьютера.

```
static void Main(string[] args)
{
    WebClient client = new WebClient();
    byte[] array = client.UploadFile("http://localhost:40007/FOT.fotproject", // http-обработчик
                                     @"D:\KRAS_12\FOT.project");
    String s = Encoding.Default.GetString(array, 0, array.Length);      // обработка ответа
}
```

Рисунок 25 Пример применения встроенного класса WebClient

### Задания

#### Задание 9. Разработка HTTP-обработчика ASP.NET:

1. разработайте HTTP-обработчик, реагирующий на GET-запрос страниц с расширением bstu; HTTP-обработчик должен принимать значения двух параметров: faculty и course; в результате выполнения в окне браузера должна отобразиться следующая строка:

GET: bstu, faculty = val_parm1, course = val_parm2, где val_parm1 и val_parm2 - значения первого и второго параметров.
--

2. разработайте HTTP-обработчик, реагирующий на POST-запрос страниц с расширением bstu; HTTP-обработчик должен принимать значения двух параметров: faculty и course; в результате выполнения в окне браузера должна отобразиться следующая строка:

POST: bstu, faculty = val_parm1, course= val_parm2, где val_parm1 и val_parm1 - значения первого и второго параметров.
--

3. опубликуйте HTTP-обработчики на IIS;
4. с помощью приложения IIS Manager отобразите окно «сопоставление обработчиков», найдите разработанные в предыдущих пунктах задания обработчики, сформируйте скриншот и поместите его в отчет по контрольной работе;
5. с помощью браузера выполните GET-запрос к HTTP-обработчику и убедитесь в его работоспособности;
6. разработайте web-форму, формирующую POST-запрос к HTTP обработчику; опубликуйте эту web-форму IIS; выполните с помощью браузера запрос к форме, сформируйте POST-запрос и убедитесь в работоспособности обработчика; текст разработанной web-формы поместите в отчет по контрольной работе;
7. поместите исходные коды HTTP-обработчиков в отчет по контрольной работе.

#### Задание 10. Разработка клиента, взаимодействующего с HTTP-обработчиком ASP.NET:

1. разработайте консольное приложение на языке C#, формирующее GET и POST-запросы к HTTP-обработчикам, разработанным в задании 9; приложение должно отображать ответ, опрaвленный обработчиками в окне консоли;

2. выполните приложение, разработанное в предыдущем пункте задания; убедитесь в работоспособности приложения и HTTP обработчиков; сформируйте скриншот, отражающий результат выполнения приложения; поместите скриншот и исходный код приложения в отчет по контрольной работе.

*Задание 11. Контрольные вопросы:*

1. поясните понятие «HTTP-обработчик ASP.NET»;
2. каким образом можно отобразить перечень HTTP-обработчиков, действующих в рамках сайта?
3. поясните структуру HTTP-обработчика (интерфейс, назначение методов);
4. где располагается информация связывающая класс HTTP обработчика, метод HTTP-запроса и расширение страниц;
5. в каких случаях целесообразно применять HTTP-обработчики?

**Практическая работа № 12. Применение серверных HTML-элементов управления ASP.NET**

*Теоретические сведения*

*Серверные HTML-элементы управления*

Для разработки пользовательского интерфейса web-приложения шаблона Web Forms ASP.NET программист использует серверные элементы управления. С их помощью разработчик приложения формирует изображение, отображающееся в окне браузера. Каждому серверному HTML-элементу управления соответствует с одной стороны asp-тег, который может быть размещен на aspx-странице, с другой – класс, производный от класса Control из пространства имен System.Web.UI.

Некоторые серверные элементы управления являются вспомогательными, не генерируют HTML-код и, соответственно, не отображаются браузером. Другие, наоборот, помимо HTML-кода, генерируют JavaScript-код, позволяющий придать динамичность пользовательскому интерфейсу.

С подробной классификацией серверных элементов управления можно ознакомиться в перечне просмотров список в панели элементов (Toolbox) Visual Studio (рис. 5.1).



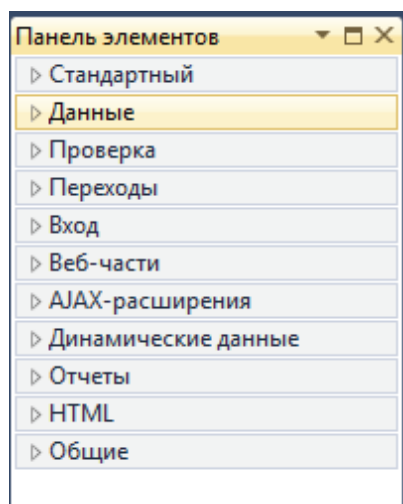


Рисунок 26 Панель элементов Visual Studio

Серверные HTML-элементы управления являются наиболее простыми в использовании элементами управления ASP.NET. Основной их особенностью является то, что каждому HTML-элементу соответствует один тег HTML. Классы, соответствующие HTML-элементам, являются производными от класса `HtmlControl` из пространства имен `System.Web.UI`. Каждый элемент может генерировать одно из двух серверных (обработанных на сервере) событий: `ServerClick` или `ServerChange`.

Сполным перечнем HTML-элементов можно ознакомиться, открыв вкладку с надписью HTML в панели элементов Visual Studio (рис. 5.2).

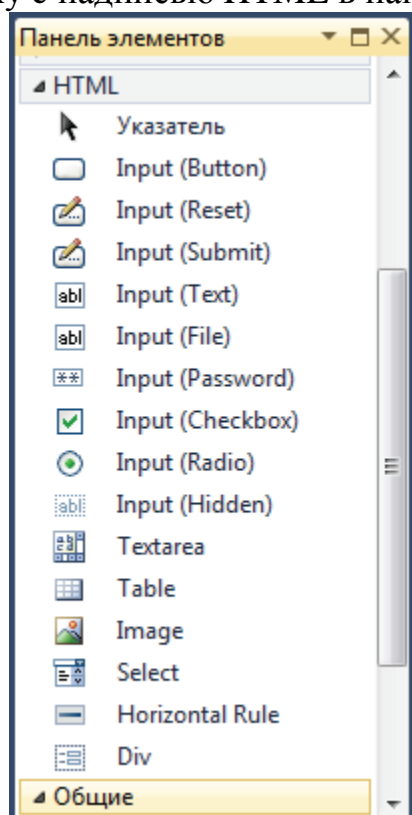


Рисунок 27 Перечень HTML-элементов в панели элементов Visual Studio

### *Применение серверных элементов управления HTML*

На рис. 5.3 представлен пример aspx-страницы, содержащей два серверных элемента HTML: `HtmlInputText` (представлен тегом `<input type="text">`) и `HtmlInputButton` (`<input type="button">`). Элемент `HtmlInputText` генерирует событие `ServerChange`, возникающее при изменении содержимого текстового значения соответствующего тега.

Элемент `HtmlInputButton` генерирует событие `ServerClick`, возникающее при щелчке мышью по кнопке соответствующего тега.

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication7._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <p>

        <input id="Text1" type="text" runat="server" onserverchange="Text1_OnServerChange" />
        <input id="Button1" type="button" value="button" runat="server"
            onserverclick="Button1_OnServerClick"/>

    </p>
</asp:Content>
```

Рисунок 28 Aspx-страница, содержащая серверные элементы управления HTML

На рис. 5.4 представлен код класса `_Default`, содержащий два метода-обработчика событий `ServerChange` и `ServerClick`.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    public void Text1_OnServerChange(Object sender, EventArgs e) // serverchange-обработчик
    {
        this.Response.Write("Text1_OnServerChange");
    }
    public void Button1_OnServerClick(Object sender, EventArgs e) // serverclick-обработчик
    {
        this.Response.Write("Button1_OnServerClick");
    }
}
```

Рисунок 29 Обработчики событий серверных элементов управления HTML

### *Задания*

#### *Задание 12. Применение серверных HTML-элементов управления:*

1. исследуйте раздел HTML панели инструментов Visual Studio;
2. разработайте web-приложение, использующее все серверные HTML-элементы управления; код обработчиков событий должен выводить на форму сообщение следующего вида:

*html-тег: событие ,*

где html-тег – наименование тега, событие – обработанное событие;

3. текст кода aspx-страницы и C#-кода поместить в отчет по контрольной работе.

#### *Задание 13. Контрольные вопросы:*

1. поясните понятие «серверный элемент управления ASP.NET»;

2. назовите общий базовый класс для всех серверных элементов управления ASP.NET;
3. назовите особенности HTML-элементов управления ASP.NET;
4. назовите общий базовый класс для всех серверных HTML-элементов управления ASP.NET;
5. перечислите все серверные HTML-элементы управления, генерирующие событие ServerChange;
6. перечислите все серверные HTML-элементы управления, генерирующие событие ServerClick.

### **Практическая работа № 13. Применение серверных базовых WEB-элементов управления ASP.NET**

#### *Теоретические сведения*

#### *Серверные базовые web-элементы управления*

Серверными web-элементами управления называются элементы управления, которые являются производными от класса WebControl из пространства имен System.Web.UI.WebControls. Среди этих элементов выделим следующие: Button, CheckBox, FileIpload, HiddenField, HyperLink, Image, ImageButton, ImageMap, Literal, Label, LinkButton, Panel, RadioButton, Table, TableCell, TableRow, TextBox. Эти элементы называют базовыми web-элементами управления ASP.NET. В панели элементов Visual Studio web-элементы расположены во вкладке «Стандартный» (рис. 6.1).

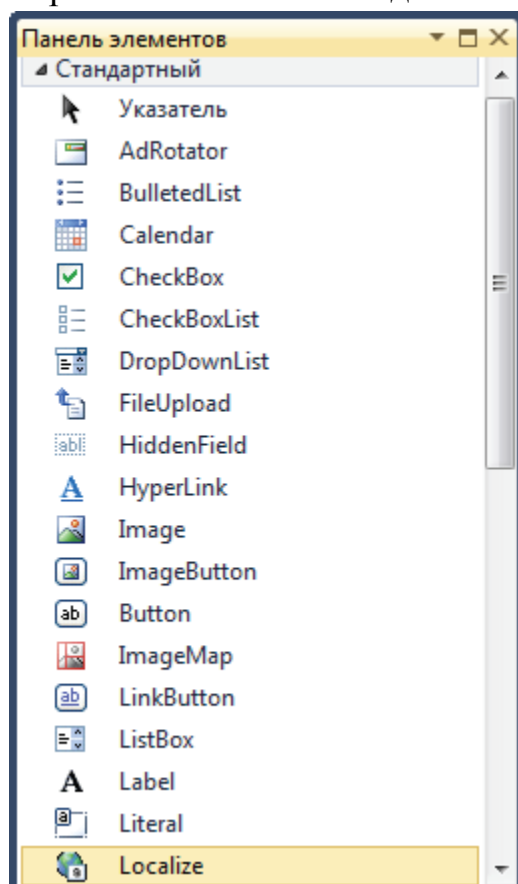


Рисунок 30 Раздел «Стандартный» панели элементов Visual Studio

На aspx-странице серверные web-элементы управления имеют префикс asp. На рис. 6.2 представлен пример использования двух серверных web-элементов: TextBox и Button. Важной особенностью web элементов является большое количество свойств и событий, поддерживаемых этими элементами. С помощью Visual Studio можно отобразить окна с перечнем свойств и событий, генерируемых web элементами (рис. 6.3). Visual Studio позволяет не только просматривать свойства и события, но и задавать значения свойствам и формировать методы-обработчики событий.

```
<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication7._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <p>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="Кнопка" onclick="Button1_Click" />
    </p>
</asp:Content>
```

Рисунок 31 Aspx-страница, содержащая серверные web-элементы TextBox и Button

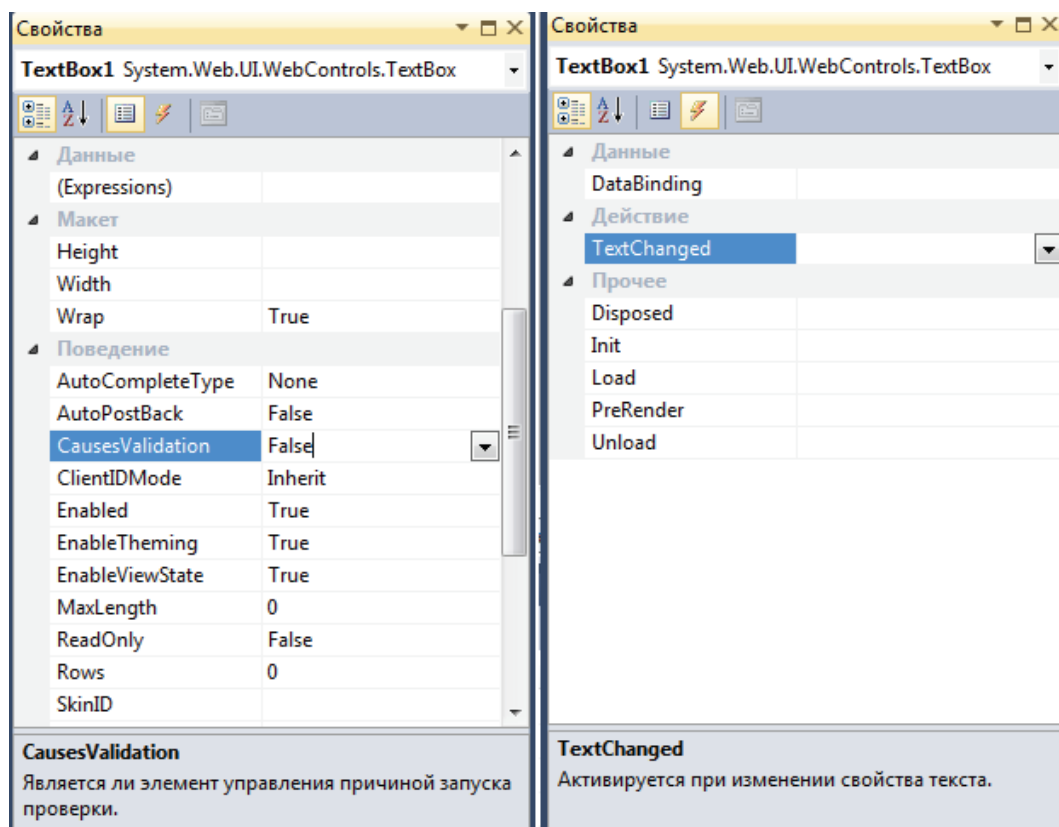


Рисунок 32 Свойства и события элемента TextBox

Результатом обработки сервером aspx-страницы будет HTML и JavaScript-код, который может интерпретироваться браузером. Обычно отправка на сервер с клиента на сервер осуществляется браузером в случае нажатия пользователем кнопки `<input type="submit">`. Этот тег может быть сгенерирован несколькими web-элементами. Например, элементом Button.

Лишь после отправки данных на сервере будут обработаны события, связанные с другими элементами.

Следует обратить внимание на свойство `AutoPostBack`, которым обладают некоторые серверные web-элементы. Установив значение `true` для этого свойства, можно осуществить отправку данных на сервер без нажатия кнопки. Такой эффект достигается с помощью JavaScript-кода, который генерируется ASP.NET при трансляции aspx-страницы.

#### *Применение серверных базовых web-элементов управления*

Visual Studio предоставляет разработчику мощный инструмент, позволяющий быстро разрабатывать приложения шаблона Web Forms. На рис. 6.4 изображена aspx-страница, отображаемая Visual Studio в режиме конструктора.

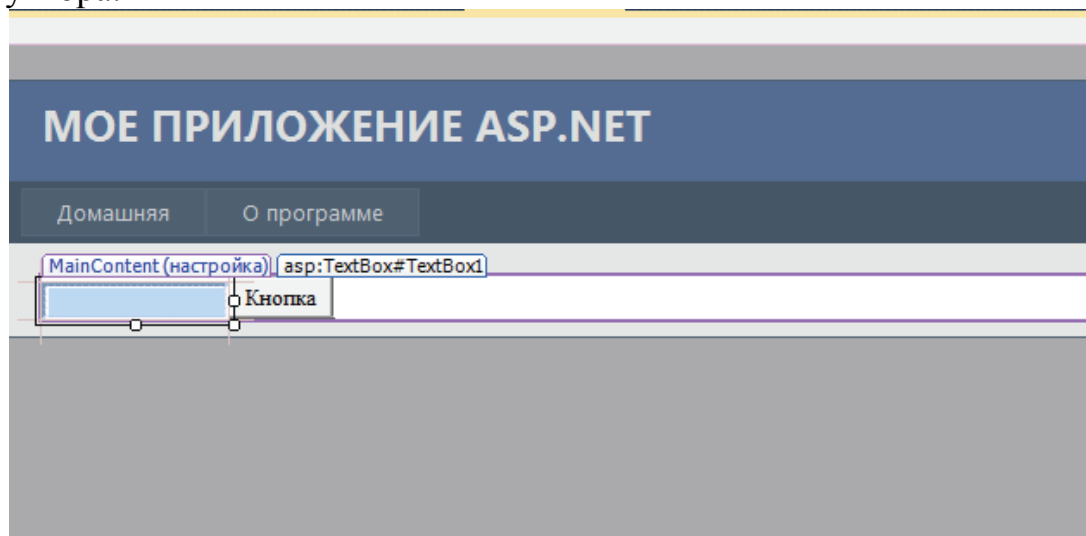


Рисунок 33 Aspx-страница в режиме конструктора

В этом режиме, разработчик может перетягивать элементы управления с панели управления прямо на форму, динамически менять размеры и месторасположение элементов на форме, с помощью контекстного меню устанавливать свойства элементов и создавать шаблоны обработчиков событий.

Принципы обработки событий для web-элементов практически ничем не отличаются от обработки событий серверных HTML элементов. На рис. 6.5. приведен пример обработчика события Click для web-элемента Button.

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_Click(object sender, EventArgs e) // обработчик события
    {
        this.Response.Write("<br /> Button1_Click");
        this.Response.Write("<br /> TextBox1.Text = '"+TextBox1.Text);
    }
}
```

Рисунок 34 Пример обработчика события для web-элемента Button

## Задания

**Задание 14. Применение серверных базовых web-элементов управления:**

1. исследуйте раздел «Стандартный» панели инструментов Visual Studio и найдите базовые web-элементы;
2. разработайте web-приложение, демонстрирующее применение базовых web-элементов, перечисленных в таблице; в приложении должны быть реализованы обработчики событий указанных в этой же таблице.

*Таблица 2 Элементы и события приложения*

Элементы	События
Button	Click
TextBox	TextChanged
CheckBox	CheckedChanged
RadioButton	CheckedChanged
HyperLink	

3. разработайте еще одно web-приложение демонстрирующее применение свойства AutoPostBack для web-элемента CheckBox;
4. тексты программ включите в отчет по контрольной работе.

**Задание 15. Контрольные вопросы:**

1. перечислите отличия между серверными HTML и web-элементами управления ASP.NET;
2. перечислите все базовые web-элементы управления и поясните их назначение и принцип применения;
3. назовите общий базовый класс для всех серверных web-элементов управления ASP.NET;
4. назовите общие свойства для всех серверных web-элементов управления;
5. поясните назначение свойства AutoPostBack web-элементов и механизм его реализации.

## **Практическая работа № 14. Применение серверных полнофункциональных элементов управления ASP.NET**

### *Теоретические сведения*

#### *Серверные полнофункциональные элементы управления*

Полнофункциональные элементы управления – это серверные web-элементы управления ASP.NET, моделирующие сложные структуры пользовательского интерфейса. В состав ASP.NET входит много полнофункциональных элементов разных категорий. В этой практической работе будут рассматриваться только универсальные элементы: AdRotator, Calendar, View, MultiView, Wizard, Menu и TreeView. Первые пять элементов управления можно обнаружить в разделе «Стандартный» панели элементов Visual Studio, а два последних – в разделе «Переходы».

Элемент AdRotator позволяет создавать рекламный баннер, в котором в соответствии с расписанием, задаваемом в виде XML-файла, меняются изображения.

Calendar – элемент, отображающий на web-странице календарь, позволяющий перемещаться по датам, выбирать одну дату или диапазон дат.

Элементы View и MultiView применяются совместно и позволяют в одной области окна браузера разместить несколько слоев (элемент View) с отображаемыми элементами управления, а также осуществлять переключение (MultiView) между этими слоями, делая один из слоев видимым, а остальные нет.

Элемент Wizard позволяет в пошаговой удобной для клиента форме ввести или выбрать данные.

Элементы TreeView и Menu позволяют предложить пользователю выбор данных в форме древовидного меню или осуществлять переходы между страницами в соответствии с логикой приложения.

#### *Применение полнофункциональных элементов управления*

Применение полнофункциональных элементов сводится к трем действиям:

1. разместить элемент на web-форме (перетащить с помощью мыши из панели элементов);
2. настроить вид и свойства элемента;
3. разработать обработчики событий элемента.

На рис. 7.1 изображен элемент Calendar в режиме «Конструктор» Visual Studio. Меню, которое отображается в окне «Автоформат» позволяет выбрать вид, в котором календарь будет отображаться на aspx-странице.

С помощью контекстного меню можно просмотреть и настроить свойства полнофункционального элемента. На рис. 7.2 отображено окно свойств элемента Calendar. С помощью этого же окна можно создать шаблоны методов обработчиков событий этого элемента.

На рис. 7.3 представлен пример обработчика события SelectionChanged, возникающего при выборе пользователем даты или интервала дат.



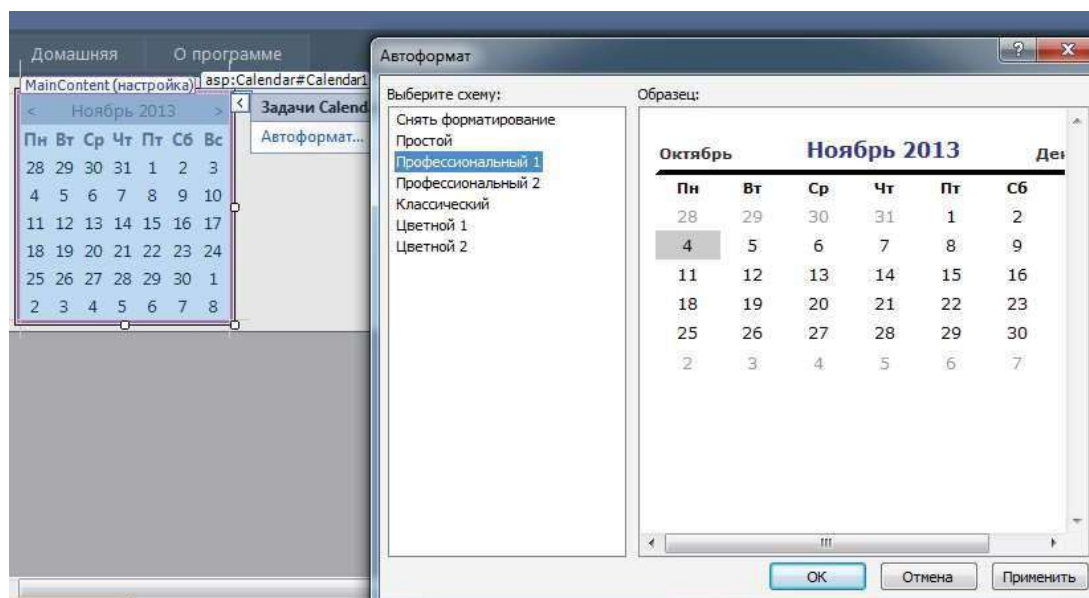


Рисунок 35 Настройка внешнего вида полнофункционального элемента Calendar

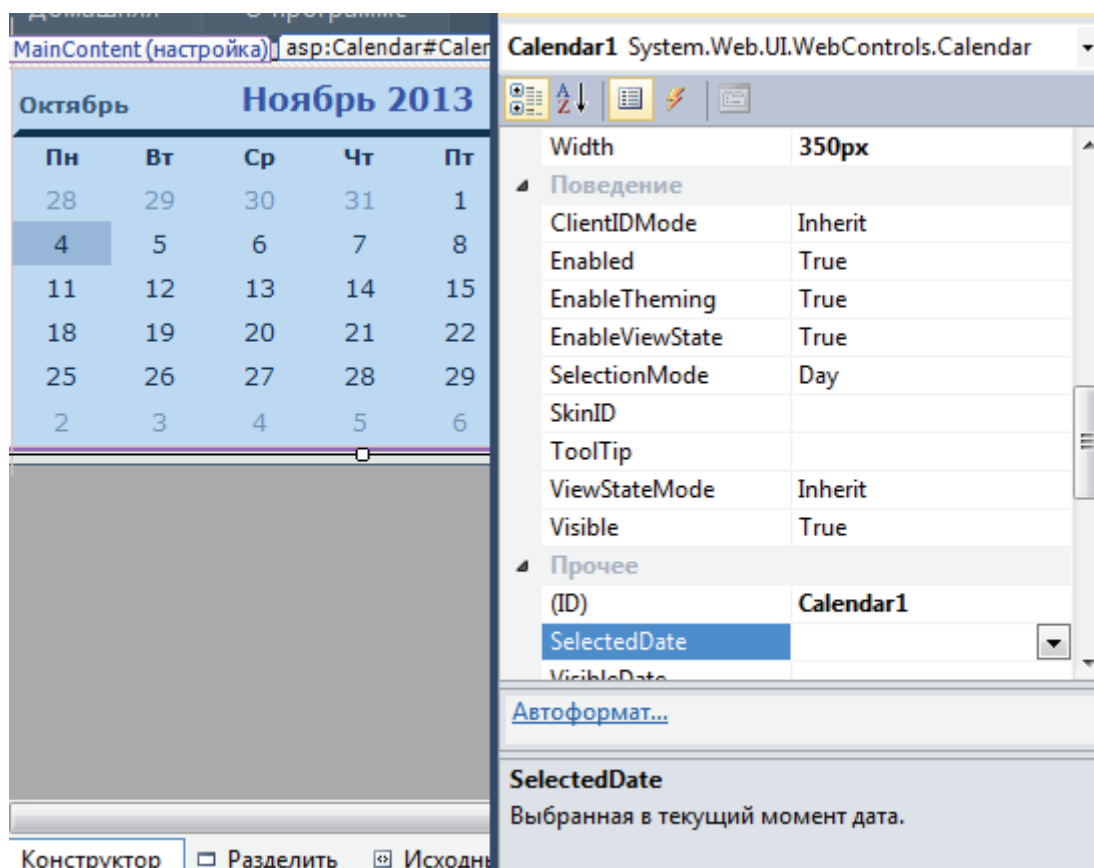


Рисунок 36 Настройка свойств полнофункционального элемента Calendar



```

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) {...}
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        foreach (DateTime d in Calendar1.SelectedDates)
        {
            this.Response.Write(d.ToString());
        }
    }
}

```

Рисунок 37 Пример обработчика события SelectionChanged полнофункционального элемента Calendar

### Задания

**Задание 16. Применение серверных базовых web-элементов управления:**

1. разработайте приложение, поддерживающее работу шести web-форм, имена и назначение которых приведены в таблице;

Таблица 3Аsrх-страницы приложения

Наименование страницы	События
Default	Стартовая страница, содержит пять элементов меню, соответствующих другим страницам и позволяющим на них перейти
View	Страница, демонстрирующая работу элементов View и MultiView. В рамках одного MultiView должно осуществляться переключение между тремя различными View, содержащими различные элементы.
Calendar	Страница, демонстрирующая работу элемента Calendar.
Wizard	Страница, демонстрирующая работу элемента Wizard. Wizard должен иметь не менее пяти шагов с вводом данных с помощью элементов TextBox, CheckBox и группы элементов RadioButton.
AdRotator	Страница, демонстрирующая работу элемента Wizard. Элемент должен отображать три различных графических изображения с частотой соответственно 50, 30 и 20.
TreeView	Страница, демонстрирующая работу элемента TreeView. Элемент должен содержать не менее трех уровней иерархии в дереве выбора

2. каждый полнофункциональный элемент, разработанного приложения, должен иметь не менее одного обработчика любого (на выбор студента) события;
3. тексты программ включите в отчет по контрольной работе.

### Задание 17. Контрольные вопросы:

1. поясните понятие «полнофункциональный элемент ASP.NET»;

2. перечислите все известные вам полнофункциональные элементы управления и поясните их назначение;
3. назовите общий базовый класс для всех полнофункциональных элементов управления ASP.NET.

## **Практическая работа № 15. Применение серверных элементов управления проверкой достоверности ASP.NET**

### *Теоретические сведения*

#### *Серверные элементы управления проверкой достоверности*

Серверные элементы управления проверкой достоверности (далее просто элементы проверки) предназначены для проверки вводимых пользователем данных. Сами элементы проверки не отображаются браузером, но в окне браузера может быть выведен результат их работы – сообщение об обнаруженной ошибке ввода данных. Как правило, проверке подвергаются данные, вводимые с помощью элементов управления TextBox, но проверка применима и к другим элементам управления, предназначенным для ввода данных: ListBox, DropDownList, RadioButtonList, HtmlInputText, HtmlTextArea и HtmlSelect.

Элементы проверки можно обнаружить в разделе «Проверка» панели элементов Visual Studio (рис. 8.1).

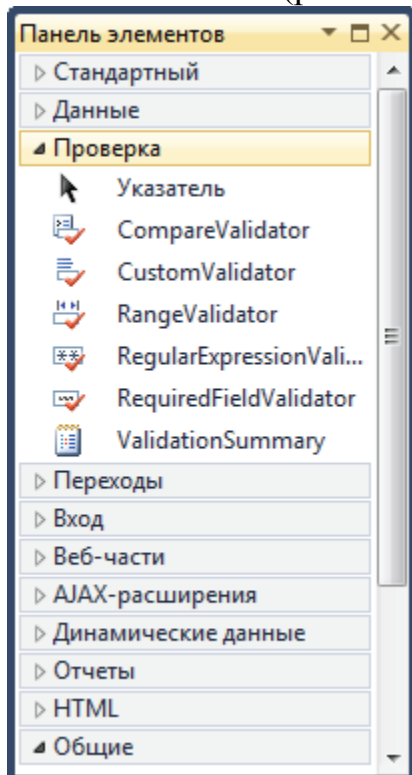


Рисунок 38 Раздел «Проверка» панели элементов Visual Studio

ASP.NET предлагает к применению шесть элементов проверки: CompareValidator, CustomValidator, RangeValidator, RegularExpressionValidator, RequiredFieldValidator и ValidationSummary. Все классы элементов проверки находятся в пространстве имен System.Web.UI.WebControls и являются производными от класса BaseValidator.

`CompareValidator` – элемент проверки, позволяющий сравнить вводимое значение с каким-то другим фиксированным значением либо, что встречается чаще, со значением, содержавшимся в другом элементе управления.

`CustomValidator` – элемент проверки, позволяющий разработчику применить собственную процедуру проверки. Как правило, к этому виду элементов проверки прибегают в том случае, если нет возможности осуществить проверку с помощью другого.

`RangeValidator` – элемент проверки, предназначенный для контроля диапазона вводимых данных;

`RegularExpressionValidator` – элемент проверки, позволяющий сравнить вводимое значение с образцом, записанным с помощью регулярного выражения.

`RequiredFieldValidator` – элемент проверки значения на пустоту. `ValidationSummary` – элемент, не выполняющий никакой проверки, а предназначенный для объединения сообщений нескольких элементов проверки в общую группу для их совместного отображения.

Несколько элементов проверки, расположенных на одной форме, могут быть объединены в группу (свойство `Validation Group`), которая может быть связана с определенным элементом, генерирующим `submit`.

Следует отметить, что проверка данных осуществляется два раза: первый раз на стороне клиента (для этого генерируется специальный JavaScript-код), второй – после успешной проверки на стороне клиента и нажатия клавиши `submit` на стороне сервера.

#### *Применение элемента управления проверкой достоверности*

На рис. 8.2 представлен пример `aspx`-страницы, содержащей два элемента типа `TextBox` (с идентификаторами `TextBox1` и `TextBox2`) и один элемент проверки (`CompareValidator1`) типа `CompareValidator`.

ASP-тег	<code>CompareValidator</code>	описывает	элемент
проверки <code>CompareValidator1</code> , который сравнивает значение, введенное			
элементом с идентификатором <code>TextBox2</code> (атрибут <code>ControlToValidate</code> ) со			
значением введенным элементом идентификатором <code>TextBox1</code>			
(ControlToCompare).			

```

<%@ Page Title="Домашняя страница" Language="C#" MasterPageFile="~/Site.master"
    AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplication7._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">

    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" Text="Button" />

    <asp:CompareValidator ID="CompareValidator1" runat="server"
        ErrorMessage="Ошибка: Второе значение <= первого"
        ControlToCompare="TextBox1" ControlToValidate="TextBox2"
        Operator="GreaterThan"></asp:CompareValidator>

</asp:Content>

```

Рисунок 39 Пример применения элемента проверки типа CompareValidator

В случае, если значение не удовлетворяет условию проверки (Operator), формируется соответствующее сообщение (ErrorMessage). На рис. 8.3. представлена aspx-страница (рис. 8.2) после ее интерпретации браузером.

Рисунок 40 Отображение сообщения, сформированного элементом проверки

В результате сравнения значений, введенных в TextBox-элементы, элемент проверки (CompareValidator1) сформировал сообщение.

#### Задания

**Задание 18. Применение серверных элементов управления проверки достоверности:**

1. разработайте приложение, обеспечивающее ввод и контроль данных с помощью элементов проверки, описанных в таблице;

Таблица 4 Вводимые данные

Данные	Проверка вводимых данных
Фамилия	Обязательный ввод, только буквы русского языка
Имя	Обязательный ввод, только буквы русского языка
Отчество	Обязательный ввод, только буквы русского языка
Дата рождения	Обязательный ввод, дата, не превышающая текущую дату
Адрес e-mail	В соответствии с правилами записи e-mail-адресов
Пароль	Обязательный ввод, неотображаемый ввод, не меньше се-

2. примените элемент ValidationSummary для формирования протокола ошибок ввода;
3. текст программы включите в отчет по контрольной работе.

### *Задание 19. Контрольные вопросы:*

1. поясните понятие «элемент управления проверкой достоверности ASP.NET»;
2. перечислите все известные вам элементы управления проверкой достоверности и поясните их назначение;
3. назовите общий базовый класс для всех элементов управления проверкой достоверности ASP.NET;
4. для чего применяются группы проверки достоверности?

## **Практическая работа № 16. Применение серверных элементов управления AJAX ASP.NET**

### *Теоретические сведения*

#### *Серверные элементы управления AJAX*

AJAX (Asynchronous JavaScript and XML) – методология построения интерфейса web-приложения, позволяющая асинхронно выполнять запросы к серверу, получать и обрабатывать ответы. В результате, при получении данных от сервера web-страница не перегружается полностью, а обновляется только ее часть. Такой подход позволяет с одной стороны ускорить выполнение запроса (за счет снижения объема пересылаемых данных), с другой – улучшить внешний вид интерфейса.

Методология AJAX основывается на возможностях объекта браузера XMLHttpRequest (позволяет формировать асинхронные запросы и обрабатывать ответы), на стандарте XML и формате JSON (форматирование пересылаемых данных), модели DOM (программный интерфейс для доступа к содержимому HTML, XHTML и XML-документов).

В панели элементов Visual Studio элементы AJAX находятся в разделе «AJAX-расширение» (рис. 9.1).

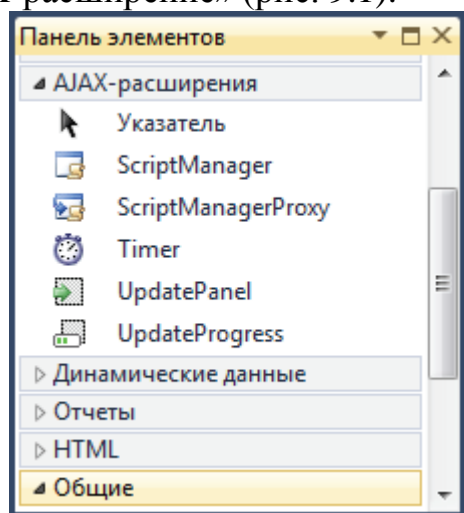


Рисунок 41 Раздел «AJAX-расширение» панели элементов Visual Studio

Элемент ScriptManager является вспомогательным, но обязателен на aspx-странице при применении других AJAX-элементов. ScriptManager не генерирует HTML-код и поэтому не имеет визуального представления. Основное его назначение – формирование ссылок на JavaScript-библиотеки AJAX ASP.NET.

Элемент ScriptManagerProxy применяется в тех случаях, когда элемент ScriptManager располагается на мастер-странице.

Элемент UpdatePanel позволяет организовать частичное обновление aspx-страницы на основании обработки ответа асинхронного запроса.

Элемент UpdateProgress работает в сочетании с частичной визуализацией, осуществляемой UpdatePanel, и предназначен для отображения сообщения и/или изображения в процессе длительного выполнения асинхронного запроса.

Элемент Timer функционально аналогичен UpdatePanel, но выполнение асинхронного запроса осуществляется автоматически в результате срабатывания таймера.

### *Применение серверных элементов управления AJAX*

На рис. 9.2 представлена aspx-страница, использующая серверный элемент UpdatePanel.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
    Inherits="WebApplication9.WebForm1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>

                    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                    <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
                    <asp:Button ID="Button1" runat="server" Text="+" onclick="Button1_Click" />

                </ContentTemplate>
            </asp:UpdatePanel>
        </div>
    </form>
</body>
</html>
```

Рисунок 42. Aspx-страница, использующая серверный элемент UpdatePanel

Тело тега UpdatePanel содержит три элемента типа TextBox и элемент типа Button. В окне браузера страница будет иметь примерно такой вид, как на рис. 9.3.



Рисунок 43 Отображение aspx-страницы (рис. 9.2) в окне браузера

Нажатие кнопки приведет к асинхронной отправке данных на сервер и выполнению кода обработчика (рис. 9.4), а также к обновлению области aspx-страницы, заключенной между начальным и конечным тегами элемента UpdatePanel.

```

public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {}

    protected void Button1_Click(object sender, EventArgs e)
    {
        int x = 0, y = 0;
        if (Int32.TryParse(this.TextBox1.Text, out x) && Int32.TryParse(this.TextBox2.Text, out y))
        {
            this.TextBox3.Text = (x + y).ToString();
        }
    }
}

```

Рисунок 44 Код обработчика Click-события кнопки (рис. 9.3)

### Задания

#### Задание 20. Применение серверных элементов управления AJAX:

1. разработайте приложение, обеспечивающее ввод двух числовых значений с помощью элементов TextBox и вывод произведения этих значений с помощью элемента Label; вычисление и отображение результата должно выполняться автоматически (обратите внимание: кнопка на форму не выводится) каждые 5 секунд; используйте AJAX-элемент Timer;
2. текст программы включите в отчет по контрольной работе.

#### Задание 21. Контрольные вопросы:

1. поясните понятия «методология AJAX», «объект XMLHttpRequest», «язык XML», «формат JSON», «модель DOM», «асинхронный запрос»;
2. поясните, какой эффект в web-приложении достигается с помощью элементов AJAX;
3. перечислите все известные вам элементы управления AJAX и поясните их назначение.

## Практическая работа № 17. Кэширование страниц ASP.NET

### Теоретические сведения

#### Кэширование aspx-страниц

Важнейшей задачей, стоящей перед разработчиком web-приложения, является обеспечение приемлемого времени ответа на запросы клиента. Одним из слагаемых величины времени ответа является промежуток, затрачиваемый на подготовку данных, отображаемых на web-странице. Этот промежуток может складываться из математических вычислений, выполнения запроса к базе данных и других слагаемых.

С другой стороны, высокий уровень актуальности отображаемой на странице информации требуется не так часто. Другими словами, не всегда для пользователя web-приложения важно: когда на странице вычислены данные: в текущий момент или пятью секундами раньше.

В таких случаях однажды сформированная страница может быть сохранена в оперативной памяти, и если другой запрос вызывает ту же страницу, она может быть извлечена и отправлена клиенту. Через некоторое заданное время сохраняемая страница считается устаревшей и при очередном



запросе она снова формируется и сохраняется. Таким образом может быть снижено время, необходимое на ответ клиенту и в большинстве случаев можно сэкономить вычислительный ресурс. Процесс временного хранения, извлечения и обновления страниц называется кэшированием вывода, а специальный программный объект ASP.NET, методы которого позволяют выполнять эти операции, называют кэшем вывода.

Кэш вывода представляет собой ассоциативную память. Он позволяет запоминать страницу, связав ее с некоторым ключевым значением, а также извлекать или обновлять страницу по ключу. В зависимости от того, каким образом формируется значение ключа для сохраняемой страницы, различают несколько видов кэширования: кэширование страницы, кэширование по параметрам, кэширование по заголовкам, пользовательское кэширование.

При простом кэшировании страницы, ключом является URL страницы.

Ключ кэширования по параметрам состоит из URL и значений определенных параметров запроса.

Кэширование по заголовкам использует ключ, состоящий из URL и значений определенных заголовков запроса.

В случае пользовательского кэширования ключ формируется функцией пользователя, которая должна быть размещена в файле Global.asax.

#### *Применение кэширования aspx-страниц*

На рис. 10.1 представлена aspx-страница, для которой применяется кэширование по параметру, а на рис. 10.2 пример ее отображения в окне браузера. Следует обратить внимание на директиву OutputCache aspx-страницы. С ее помощью задаются параметры кэширования страницы: интервал устаревания кэша (параметр Duration) и имя параметра (VaryByParam), значения которого являются составной частью ключа кэширования.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebCache.aspx.cs"
    Inherits="Cache.WebCache" %>
<%@ OutputCache Duration = "10" VaryByParam="parm1" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server"><title> Cache/Substitution </title></head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Literal ID="Literal1" Text = "Cache DateTime.Now" runat="server"/>
            <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
            <br />
            <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
            <asp:Literal ID="Literal2" Text = "Substitution DateTime.Now" runat="server"/>
            <asp:Substitution ID="Substitution1" runat="server" MethodName="GetCurDateTime" />
        </div>
    </form>
</body>
</html>
```

Рисунок 45 Кэширование aspx-страницы



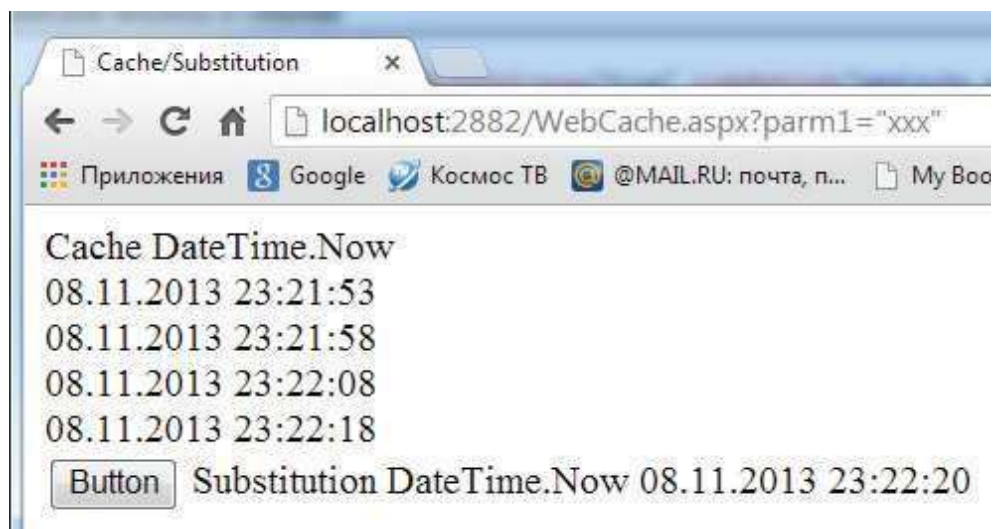


Рисунок 46 Отображение браузером aspx-страницы, представленной на рис. 10.1

Кроме того, на aspx-странице (рис. 10.1) применен элемент Substitution, позволяющий вычислить значение (имя функции, указанное значением атрибута MethodName), заменяющее элемент Substitution и не участвующее в процессе кэширования. Часто такое использование элемента Substitution называют после кэшевой подстановкой.

Результат, полученный на рис. 10.2 отображает результат, который может быть получен при многократном нажатии кнопки с частотой примерно два раза в секунду. При этом в столбце, озаглавленном «Cache DateTime.Now» строки добавляются по одной каждые 10 секунд (значение Duration), а в строке, подписанной «Substitution DateTime.Now», каждую секунду (элемент Substitution).

На рис. 10.3 представлен код обработчиков событий aspx страницы (рис. 10.1). Обратите внимание, что код в методе Page\_Load выполняется при каждом нажатии кнопки и если бы не применялось кэширование, строки в столбце «Cache DateTime.Now» (рис. 10.2) добавлялись бы каждую секунду. При этом функция GetCurDate будет выполняться при каждом нажатии кнопки (элемент Substitution) и результат ее выполнения отображаться сразу, без задержки.

```
public partial class WebCache : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        this.Label1.Text += ("<br />" + DateTime.Now.ToString());
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
    }
    private static string GetCurDateTime(HttpContext ctx)
    {
        return DateTime.Now.ToString();
    }
}
```

Рисунок 47 Код обработчиков событий aspx-страницы, представленной на рис. 10.1

## *Задания*

### *Задание 22. Применение кэширования aspx-страниц:*

1. разработайте приложение, применяющее кэширование на 5 секунд aspx-страницы по параметру; с помощью вывода в окно браузера, продемонстрируйте, что кэширование действительно выполняется;
2. добавьте в aspx-страницу разработанного приложения элемент Substitution, применение которого демонстрировало бы обновление части aspx-страницы без задержек, связанных с кэшированием;
3. текст кода программы включите в отчет по контрольной работе.

### *Задание 23. Контрольные вопросы:*

1. поясните понятие «кэширование вывода»;
2. поясните принцип реализации кэширования вывода в ASP.NET;
3. поясните, в каких случаях целесообразно применять кэширование вывода
4. перечислите все типы кэширования вывода в ASP.NET поясните принцип их реализации;
5. поясните термин «после-кэшевая подстановка».

## **Лабораторный практикум.**

### **Лабораторная работа №1. Web-сервер Apache**

#### *Цель работы:*

Получить практические навыки в установке и выполнении базовой настройки web-сервера Apache.

#### *Задание (типовое):*

Установить web-сервер Apache, проверить правильность установки, выполнить настройку web-сервера, протестировать работу web-сервера, удалить web-сервер Apache.

Порядок выполнения лабораторной работы:

1. Создать каталог disc:\webprog.
2. Установить web-сервер Apache в каталог disc:\webprog\Apache2.2 как консольное приложение.
3. Запустить web-сервер (Пуск/Все программы/Apache HTTP Server 2.2/Control Apache Server/Start Apache in Console).
4. Проверить правильность установки web-сервера, набрав в строке адреса браузера адрес <http://127.0.0.1:8080>.
5. Если web-сервер не запускается, посмотреть причину незапуска в файле disc:\webprog\Apache2.2\logs\error.log
6. Для остановки web-сервера использовать комбинацию клавиш Ctrl+C.
7. Ознакомиться с документацией по web-серверу Apache. Для этого в файле httpd.conf убрать комментарий в строке с директивой #Include conf/extra/httpd-manual.conf. Документация будет доступна по адресу <http://127.0.0.1:8080/manual>

8. Создать два виртуальных хоста на одном IP-адресе 127.0.0.1, настроив их на разные порты, например, 8081 и 8082. Расположить корневые каталоги документов хостов соответственно в каталогах disc:\webprog\vh1 и disc:\webprog\vh2.
9. Файлы для регистрации доступа access.log и ошибок error.log расположить в каталоге disc:\webprog\vhlogs.
10. Создать файлы с описанием групп пользователей и отдельных пользователей, и расположить их в каталоге disc:\webprog\vhsecurity.
11. При настройке виртуальных хостов изменить, при необходимости, настройки для корневого каталога web-сервера.
12. В корневом каталоге для документов виртуального хоста vh1 создать несколько каталогов и файлов. Определить различные права доступа к различным каталогам и файлам:
  - доступ разрешен всем;
  - доступ разрешен отдельным пользователям;
  - доступ разрешен группе пользователей;
  - доступ разрешен всем зарегистрированным пользователям;
  - доступ запрещен всем.
13. Протестировать работу SSI (Server Side Includes) — директив включения на стороне сервера.
14. В корневом каталоге для документов виртуального хоста vh2 организовать расширенную индексацию.
15. Перенести некоторые директивы (например, директивы для определения прав доступа, служебной индексации и т.п.) из основного конфигурационного файла в файлы .htaccess, расположенные непосредственно в каталогах, для которых выполняются настройки.
16. Удалить web-сервер Apache (Пуск/Панель управления/Установка и удаление программ).
17. Удалить каталог disc:\webprog.

*Содержание отчета (отчет в электронном виде):*

- отчет сохранить в файле с именем АВТ-000 Иванов (лр1).doc;
- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- дерево созданных каталогов;
- конфигурационные файлы;
- файлы с именами и группами пользователей;
- выводы по работе.

## *Теоретические сведения*

### *Web-сервер Apache*

Apache — один из популярных web-серверов в мире. В настоящее время программное обеспечение Apache установлено более чем на половине серверов.

Для настройки web-сервера Apache используются конфигурационные файлы:

- основной конфигурационный файл `httpd.conf`, расположенный в каталоге `conf`;
- дополнительные конфигурационные файлы, расположенные в каталоге `conf\extra` и подключаемые основному конфигурационному файлу `httpd.conf` по мере необходимости с помощью директивы `Include`;
- конфигурационные файлы `.htaccess`, расположенные непосредственно в каталогах, для которых выполняются настройки.

В конфигурационных файлах с помощью директив определяется, как web-сервер должен работать с ресурсами, отвечая на запрос, указывается, с какими файлами пользователи могут выполнять определенные операции. Настройка конфигурационных файлов web-сервера Apache — самый ответственный шаг после его установки.

Web-сервер Apache читает основной конфигурационный файл `httpd.conf` однократно при запуске. Если web-сервер работает, то при изменении конфигурационного файла `httpd.conf` следует перезапустить web-сервер.

В конфигурационном файле `httpd.conf` и файлах `.htaccess` содержатся директивы, которые управляют работой web-сервера Apache.

В конце основного конфигурационного файла `httpd.conf` перечислены директивы `Include`, позволяющие подключить дополнительные конфигурационные файлы из каталога `conf\extra`.

### *Виртуальные хосты*

Web-сервер Apache позволяет настроить виртуальные хосты.

Виртуальные хосты позволяют разместить более чем один web-сайт, используя один экземпляр web-сервера. Виртуальный хост может быть как «привязанным к IP-адресу» (IP-based), что позволяет использовать отдельный IP-адрес для каждого web-сайта, так и «привязанным к имени» (name-based), что позволяет использовать один и тот же IP-адрес для нескольких web-сайтов, различая виртуальных хосты по именам или номерам портов.

Для организации виртуальных хостов используются директивы `Listen`, `NameVirtualHost` и блочная директива `VirtualHost` (все примеры приведены для name-based виртуального хоста, определяемого номером порта и web-сервера Apache, установленного как консольное приложение).

Директива `Listen` задает номер порта, который «слушает» web-сервер. В конфигурационном файле может присутствовать несколько директив `Listen`.

<code>Listen 8081</code>
--------------------------

Директива `NameVirtualHost` позволяет создать name-based виртуальный хост со своим номером порта.

```
NameVirtualHost 127.0.0.1:8081
```

Блочная директива `<VirtualHost>` позволяет задать директивы, определяющие режимы работы виртуального хоста.

```
<VirtualHost 127.0.0.1:8081>
CustomLog ../.../access.log common
ErrorLog ../.../error.log
DocumentRoot ../.../www
<Directory ../.../www>
Options ...
...
</Directory>
<Files ../.../test.html>
...
</Files>
...
</VirtualHost>
```

Для настройки виртуального хоста можно использовать практически все директивы web-сервера Apache. Узнать, разрешена ли директива для использования в блочной директиве `</VirtualHost>` можно в локальной документации, доступной по адресу <http://127.0.0.1:8080/manual>. Директиву разрешено использовать в блочной директиве `</VirtualHost>` в случае, если в описании директивы в разделе Context указан virtual host.

Рекомендуется для каждого виртуального хоста с помощью директивы `DocumentRoot` задавать отдельный каталог для документов web-сайта, так как именно по этой причине и создаются виртуальные хосты.

Файлы регистрации доступа и ошибок могут быть одними и теми же для нескольких виртуальных хостов.

Блочные директивы `<Directory>` и `<Files>` предназначены для задания директив, применяемых к соответствующим каталогам и файлам (например, для организации доступа к каталогу или файлу).

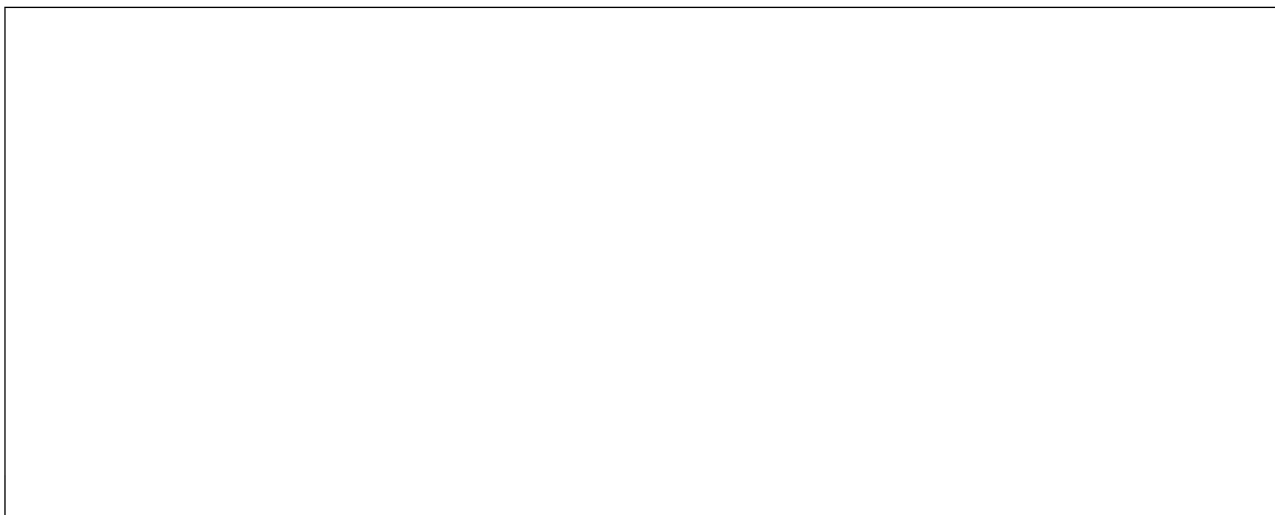
#### *Организация доступа*

Для организации доступа к каталогам и файлам используются директивы `Allow`, `Deny`, `AuthType`, `AuthName`, `AuthGroupFile`, `AuthUserFile` и `Require`.

Директивы `Allow`, `Deny` позволяют открыть / закрыть доступ для всех пользователей или пользователям, пришедшим с определенного хоста, домена или IP-адреса.

```
Allow from all / Deny from all
Allow from apache.org / Deny from apache.org
Allow from .net / Deny from .net
Allow from 192.168.1.104 / Deny from 192.168.1.104
Allow from 192.168 / Deny from 192.168
```

Порядок применения директив Allow и Deny определяется директивой Order.



Директивы AuthType, AuthName, AuthGroupFile, AuthUserFile и Require позволяют открыть / закрыть доступ для зарегистрированных пользователей.

Директива AuthType задает тип контроля полномочий.

```
AuthType Basic
```

Директива AuthName задает область, в которой действительны имена и пароли пользователей.

```
AuthName Test
```

Директивы AuthGroupFile и AuthUserFile задают имена текстовых файлов, в которых содержится информация о группах и пользователях, входящих в группы и именах пользователей и паролях. Файлы имеют следующий формат:

```
group1:user1 user2 ...
group2:user3 user4 ...
...

user1:password1
user2:password2
...
```

Пароли пользователей могут храниться как в незашифрованном, так и в зашифрованном виде. Для шифрования паролей используется утилита bin\htpasswd.exe. Для получения справочной информации по работе с утилитой следует запустить утилиту с ключом -?.

Имена файлов групп и пользователей выбираются произвольно, как и их расположение, единственное соображение безопасности заключается в том, что каталог с файлами лучше располагать выше каталога, заданного директивой DocumentRoot.

Директива `Require` определяет права доступа для отдельных пользователей, групп пользователей, всех зарегистрированных пользователей.

```
Require user user1 user2 ...
#доступ разрешен перечисленным пользователям

Require group group1 group2 ...
#доступ разрешен перечисленным группам
пользователей

Require valid-user
#доступ разрешен всем зарегистрированным
пользователям
```

### *Служебная индексация*

В случае если каталог, заданный директивой `DocumentRoot`, не содержит индексного файла (директива `DirectoryIndex`), web-сервер Apache создает служебный индексный файл. Параметр `Indexes` директивы `Options` разрешает формирование служебного индексного файла.

Для изменения вида служебного индексного файла можно включить расширенную индексацию директивой `IndexOptions`.

```
IndexOptions FancyIndexing
```

Для расширенной индексации можно использовать директивы `AddIcon`, `AddDescription`, `HeaderName`, `ReadmeName`, `IndexIgnore` (описание директив для расширенной индексации см. в локальной документации в разделе «Reference Manual/Run-time Configuration Directives»).

### *Директивы включения на стороне сервера (SSI — Server Side Includes)*

Директивы SSI, содержащиеся в документах, позволяют реализовать на серверной стороне выполнение некоторых действий и включить результаты этих действий в документы перед их отправкой клиенту. Аналогичные возможности, гораздо более широкие, предоставляют серверные скриптовые языки.

Формат директивы:

```
<!--#directive attribute=value attribute=value ... -
->
```

Директивы SSI оформляются как комментарии.

Для настройки web-сервера Apache для работы с директивами SSI используются директивы `Options`, `AddOutputFilter` и `AddType`.

```
Options Includes
#разрешает использование директив SSI

AddOutputFilter INCLUDES .ssi
#задает соответствие между расширением имени файла
и фильтром,
```

```
#который будет обрабатывать ответ сервера перед  
отправкой клиенту
```

```
AddType text/html .ssi  
#задает соответствие между расширением имени файла  
и media-типом
```

Описание директив включения на стороне сервера см. в локальной документации (расположено в разделе «Server Side Includes (SSI)/Basic SSI directives»).

## **Лабораторная работа №2. Статический html-документ**

*Цель работы:*

Изучить основы языка разметки гипертекста HTML 4.

*Задание:*

Создать html-документ, в разметке документа использовать:

- тег для определения кодировки кириллицы <meta>;
- тег комментария <! ----- >;
- теги форматирования текста: <p>, <br>, <div>, <span>, <hr>, <h1> ÷ <h6>, <b>, <i>, <u>, <sub>, <sup>, <pre>, <tt>, продемонстрировать отличия тегов <p> и <br>, <div> и <span>;
- тег для разметки изображения <img>;
- тег для разметки гиперссылок <a>, разметить ссылки на другой документ, в пределах размечаемого документа, на email;
- с помощью параметров тега <body> изменить цвет фона документа, цвет текста, цвета не посещённых и посещённых ссылок документа, используя цвета из web-безопасной (гарантированной) палитры;
- теги для разметки списка, таблицы, формы в соответствии с вариантом.



Вариант 1:

HTML 4

← → ↻ 🔍

т. Зима

- Декабрь
- Январь
- Февраль

ті. Весна

тіі. Лето

тііі. Осень


Новосибирск

☐ Омск

☒ Новосибирск

☐ Томск

\*\*\*\*\*

Новосибирск ▼

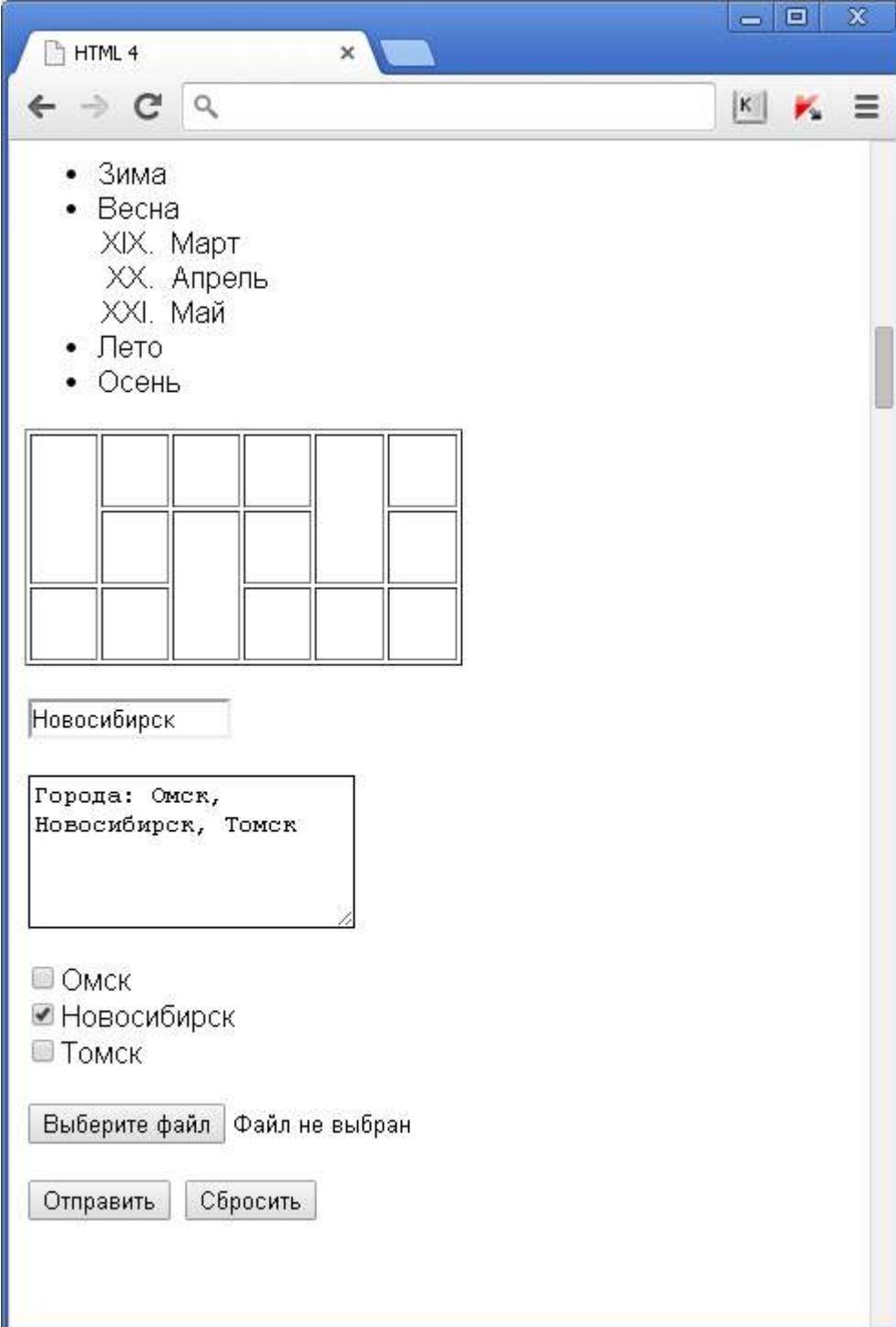
Омск

Новосибирск

Томск

Отправить Сбросить

Вариант 2:



The screenshot shows a web browser window with a single tab titled "HTML 4". The address bar is empty. The page content includes a list of seasons, a calendar grid, a city selection dropdown, a list of cities with checkboxes, a file upload button, and two action buttons.

- Зима
- Весна
  - XIX. Март
  - XX. Апрель
  - XXI. Май
- Лето
- Осень


Новосибирск

Города: Омск,  
Новосибирск, Томск

☐ Омск  
☒ Новосибирск  
☐ Томск

Выберите файл    Файл не выбран

Вариант 3:

с. Зима  
сі. Весна  
сіі. Лето  
    ▪ Іюнь  
    ▪ Іюль  
    ▪ Август  
сііі. Осень


Города: Омск,  
Новосибирск, Томск

☐ Омск  
☒ Новосибирск  
☐ Томск

.....

Новосибирск ▼  
Омск  
Новосибирск  
Томск

Отправить Сбросить

Вариант 4:

HTML 4

← → ↻ 🔍 K 🚫 ☰

y. Зима  
z. Весна  
aa. Лето  
ab. Осень

- Сентябрь
- Октябрь
- Ноябрь


Новосибирск

☐ Омск

☒ Новосибирск

☐ Томск

☐ Омск

☒ Новосибирск

☐ Томск

Выберите файл

Файл не выбран

Отправить

Сбросить

Вариант 5:

HTML 4

← → ↻ 🔍 K 🔴 ☰

АА. Зима

- Декабрь
- Январь
- Февраль

АВ. Весна

АС. Лето

АД. Осень


Города: Омск,  
Новосибирск, Томск

☐ Омск

☒ Новосибирск

☐ Томск

.....

Новосибирск ▼

- Омск
- Новосибирск
- Томск

Отправить Сбросить

Вариант 6:

The screenshot shows a web browser window with a single tab titled "HTML 4". The address bar is empty. The page content includes a list of seasons with sub-items, a calendar grid, a city selection dropdown, a radio button group for cities, a password field, a file selection button, and two action buttons.

HTML 4

← → ↻ 🔍

- Зима
- Весна
  - X. Март
  - Y. Апрель
  - Z. Май
- Лето
- Осень


Новосибирск

☐ Омск  
☒ Новосибирск  
☐ Томск

.....

Выберите файл    Файл не выбран

Вариант 7:

The screenshot shows a web browser window with a single tab titled "HTML 4". The address bar is empty. The page content includes a list of items with sub-items, a grid of 12 empty boxes, a text area with pre-filled city names, a list of cities with checkboxes, a file selection button, a dropdown menu, and two action buttons.

k. Зима  
l. Весна  
m. Лето  
    ▪ Июнь  
    ▪ Июль  
    ▪ Август  
n. Осень


Города: Омск,  
Новосибирск, Томск

☐ Омск  
☒ Новосибирск  
☐ Томск

Выберите файл    Файл не выбран

Новосибирск ▼  
Омск  
Новосибирск  
Томск

Отправить    Сбросить

Вариант 8:

HTML 4

← → ↻ 🔍

- Зима
- Весна
- Лето
- Осень

тгхiv. Сентябрь

тгхv. Октябрь

тгхvi. Ноябрь


Новосибирск

☐ Омск

☒ Новосибирск

☐ Томск

.....

Новосибирск ▼

Омск

Новосибирск

Томск

Отправить Сбросить



Вариант 9:

HTML 4

← → ↻ 🔍 K 🚫 ☰

XIV. Зима

- Декабрь
- Январь
- Февраль

XV. Весна

XVI. Лето

XVII. Осень


Новосибирск

Города: Омск,  
Новосибирск, Томск

☐ Омск  
☒ Новосибирск  
☐ Томск

Выберите файл    Файл не выбран

*Вариант 10:*

HTML 4

← → ↻ 🔍

С. Зима  
D. Весна  
    ◦ Март  
    ◦ Апрель  
    ◦ Май  
E. Лето  
F. Осень


Города: Омск,  
Новосибирск, Томск

☐ Омск  
☒ Новосибирск  
☐ Томск

.....

Новосибирск ▼  
Омск  
Новосибирск  
Томск

Отправить Сбросить

*Порядок выполнения лабораторной работы:*

1. Запустить текстовый редактор.
2. Разметить html-документ в соответствии с заданным вариантом. Для справки о тегах и их атрибутах можно использовать справочник, расположенный по адресу <http://htmlbook.ru>.
3. Протестировать созданный документ, по крайней мере, в двух браузерах, отметить различия в отображении документа.

*Содержание отчета (отчет в электронном виде):*

- отчет сохранить в файле с именем АВТ-000 Иванов (лр2).doc;

- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- html-разметка созданного документа;
- скриншоты html-документа для различных браузеров;
- выводы по работе.

#### *Теоретические сведения*

Язык разметки гипертекста HTML (Hypertext markup language) — язык разметки, используемый для создания гипертекстовых html-документов, отображаемых браузером.

Гипертекст — форматированный текст, содержащий ссылки на другие документы (гиперссылки).

Разметка — вставка в текст дополнительных служебных символов, каждый из которых является командой, указывающей браузеру, как следует отображать документ.

Язык разметки гипертекста HTML не является языком программирования.

Основным элементом языка разметки гипертекста HTML является тег (tag).

Теги содержат указания браузеру о способах отображения документа.

С помощью тегов в html-документ вставляются файлы, содержащие дополнительные данные (например, графику) и размечаются гиперссылки, посредством которых данный html-документ связывается с другими html-документами.

Как правило, теги состоят из начального и конечного элементов, между которыми размещаются текст и другие элементы html-документа. Имя конечного тега совпадает с именем начального, но перед именем конечного тега ставится косая черта.

Базовый синтаксис тега:

<pre>&lt;name&gt; Содержимое тега &lt;/name&gt;</pre>
---

где <name> — это начальный элемент тега, содержащий имя тега, а </name> — конечный элемент тега.

В начальном элементе тега может располагаться перечень атрибутов тега. Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими пробелами. Порядок записи атрибутов в начальном элементе тега значения не имеет. Значение атрибута, если имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута — одно слово или число, то его можно указать после знака равенства, не заключая в кавычки. Все остальные значения необходимо заключать в кавычки, особенно если они содержат пробелы. Если атрибут не указан, браузером используется его значение по умолчанию.

Регистр символов в именах тегов и атрибутов не учитывается.

```
<name      attribute_1="value_1"      attribute_2      ...  
attribute_n="value_n">  
Содержимое тега  
</name>
```

Конечные теги никогда не содержат атрибутов.

При использовании вложенных тегов их нужно закрывать, соблюдая правильную вложенность.

В некоторых случаях конечные теги можно опускать. Тем не менее, рекомендуется использовать конечные элементы тегов, чтобы избежать ошибок в отображении html-документа браузером.

Некоторые теги, не имеющие конечного элемента, называются автономными тегами.

```
<name>  
  
<name      attribute_1="value_1"      attribute_2      ...  
attribute_n="value_n">
```

Html-документ состоит из заголовка документа и тела документа.

```
<html>  
<head>  
<title>Название html-документа</title>  
Заголовок html-документа  
</head>  
<body>  
Тело html-документа  
</body>  
</html>
```

Весь html-документ заключается в тег <html>. Html-документ состоит из заголовка и тела, которые выделяются, соответственно, тегами <head> и <body>. В заголовке, с помощью тега <title>, указывается название html-документа, а также другие данные, которые браузер будет использовать при отображении документа.

Тело html-документа — та его часть, в которую помещается собственно содержимое html-документа. Тело включает предназначенный для отображения текст и управляющую разметку документа (теги), которые используются браузером.

Перечень тегов языка HTML и их атрибутов можно посмотреть в справочнике <http://htmlbook.ru>.

Спецификация языка разметки гипертекста HTML 4.01 расположена на сайте WWW-консорциума по адресу <http://www.w3.org/TR/1999/REC-html401-19991224>.

### **Лабораторная работа №3. Каскадные таблицы стилей CSS**

*Цель работы:*

Изучить основы каскадных таблицы стилей CSS 3.

*Задание:*

Изменить html-документ, полученный в результате выполнения лабораторной работы №2 «Создание статического html-документа», изменив в нем с помощью каскадных таблиц стилей:

- текст (шрифт, размер, цвет, поля, обрамление);
- гиперссылки (цвет непосещенных и посещенных ссылок);
- документ (фон);
- список (маркеры или нумерацию);
- таблицу (границы, фон).

Использовать определение стилей для тегов и классы стилей, псевдоклассы.

Использовать три способа определения каскадных таблиц стилей:

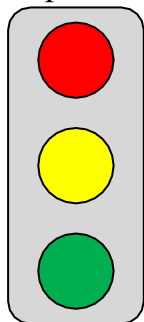
- с помощью тега `<link>`;
- с помощью тега `<style>`;
- с помощью параметра `style` тега.

Продемонстрировать действие приоритетов при применении различных способов определения CSS;

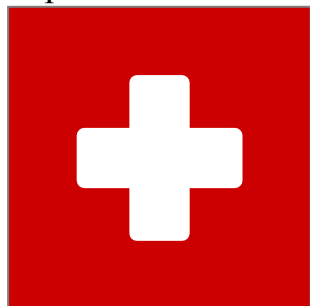
Создать два слоя, частично перекрывающихся друг на друга.

Создать изображение в соответствии с вариантом, используя только свойства CSS.

Вариант 1:

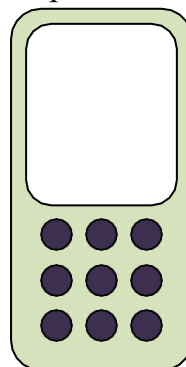


Вариант 2:

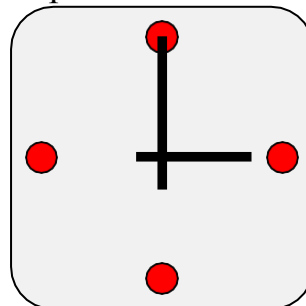


Вариант 3:

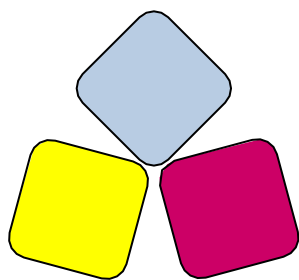
Вариант 6:



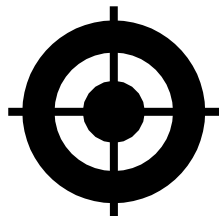
Вариант 7:



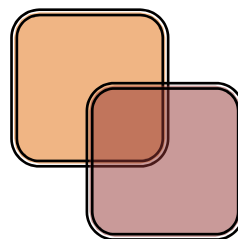
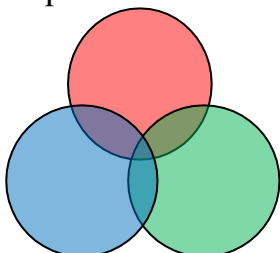
Вариант 8:



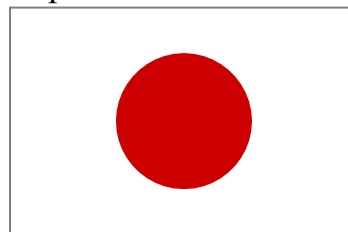
Вариант 4:



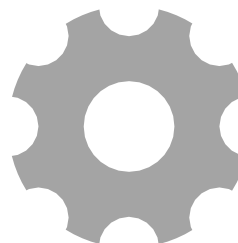
Вариант 5:



Вариант 9:



Вариант 10:



*Порядок выполнения лабораторной работы:*

1. Запустить текстовый редактор.
2. Изменить с помощью каскадных таблиц стилей html-документ в соответствии с заданным вариантом. Для справки о свойствах и их значениях можно использовать справочник, расположенный по адресу <http://htmlbook.ru>.
3. Протестировать созданный документ в браузере.

*Содержание отчета (отчет в электронном виде):*

- отчет сохранить в файле с именем АВТ-000 Иванов (лр3).doc;
- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- код созданного html-документа, включая созданные стили;
- скриншот html-документа;
- выводы по работе.

*Теоретические сведения*

Каскадные таблицы стилей CSS (Cascading style sheets) — формальный язык описания внешнего вида документа, созданного с использованием языка разметки гипертекста.

Каскадные таблицы стилей позволяют разделить описание логической структуры html-документа (выполненное с помощью языка разметки) и описание внешнего вида html-документа (выполненное с помощью CSS).

Существует три способа определения стилей: 1) в отдельном файле, подключаемом к html-документам, 2) с помощью тега <style> непосредственно в некотором html-документе и 3) с помощью атрибута style непосредственно в некотором теге.

Наиболее высокий приоритет имеет стиль, определенный в теге, затем следует определение стиля с помощью тега style и самым низким приоритетом обладают свойства, определенные в отдельном файле.

Каскад приоритетов особенно удобен при разработке больших проектов, например, сайтов, состоящих из большого числа html-документов. В этом случае общее оформление может быть вынесено в отдельный файл, в html-документе могут быть внесены изменения в стиль документа с помощью тега <style>, атрибут тега style позволяет изменить оформление одного тега.

Стили определяются парами свойств и значений, перечень пар заключается в фигурные скобки и пары разделяются точкой с запятой:

```
{property_1:value_1;   property_2:value_2;   ...   ;  
property_n:value_n}
```

где property — это свойство, а value — значение свойства.

Стиль можно определить для конкретного тега, например, задать для тега <body> отображение белого текста на черном фоне:

```
body  
{background-color:black; color:white}
```

Можно определить «чистый» стиль, не привязанный заранее к конкретному тегу, в этом случае речь идет об определении класса стиля:

```
.small_silver  
{font-size:10px; color:silver}  
или  
#big_gold  
{font-size:150px; color:#D7B56D}
```

Применение класса стиля:

```
<p class=small_silver>Текст светло-серого цвета  
размером 10 пиксел</p>  
или  
  
<p id=big_gold>Текст светло-желтого цвета размером  
150 пиксел</p>
```

Описание стилей для тегов или классов стилей выполняется одинаково как в отдельном файле, так и в теге <style>.

Файл со стилями должен иметь расширение \*.css и быть подключен к html-документу с помощью тега <link>, расположенного в теге <head>.

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

Тег <style> также должен быть расположен в теге <head>, после тега <link>.

Стили, определяемые непосредственно в теге с помощью атрибута style:

```
<p style="text-decoration-line:underline;
color:rgb(255,0,0)">Подчеркнутый текст красного
цвета</p>
```

Возможно задание различных стилей отображения одного и того же html-документа в различных средах представления, например, на экране или печати с помощью атрибута media тега <link>.

Файл screen.css

```
body
{color:silver; background:black}
.forprint
{display:none}
```

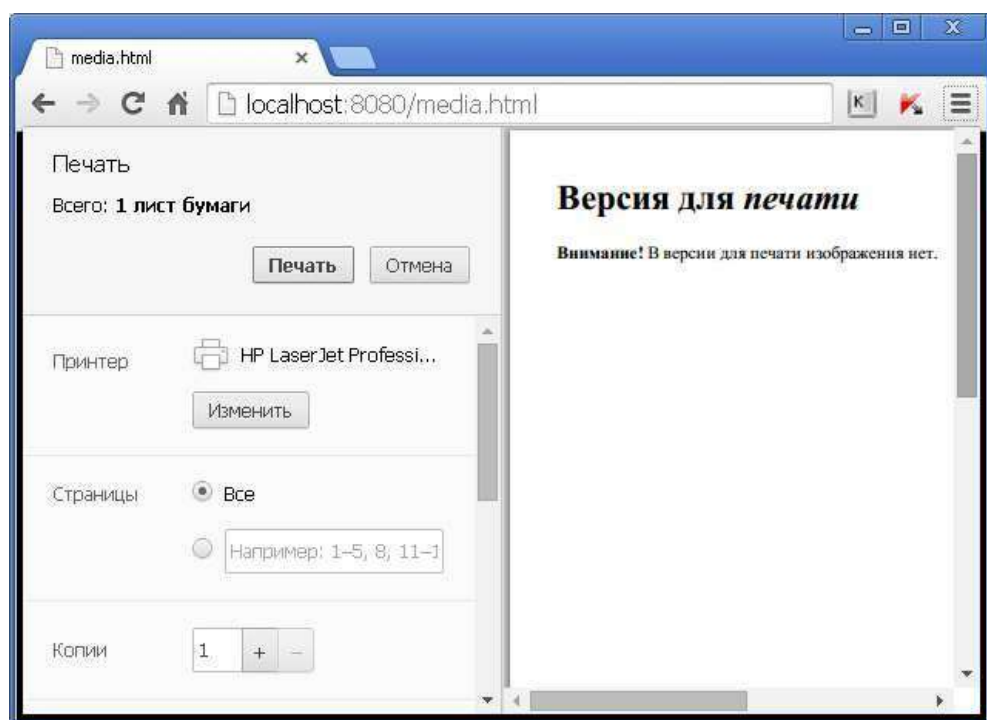
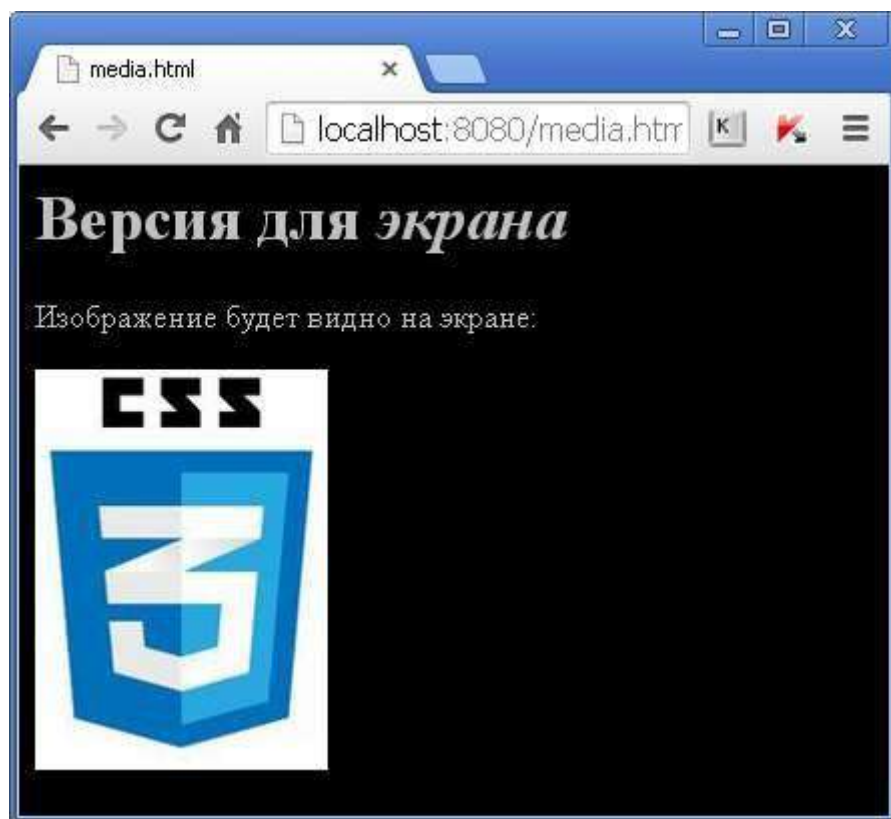
Файл print.css

```
body
{color:black; background:white}
.forscreen
{display:none}
```

Файл media.html

```
<html>
<head>
<link href="screen.css" rel="stylesheet"
type="text/css" media="screen">
<link href="print.css" rel="stylesheet"
type="text/css" media="print">
</head>
<body>
<h1>Версия для <i class=forscreen>экрана</i><i
class=forprint>печати</i></h1>
<div class=forscreen>Изображение будет видно на
экране:
<p><img src=css3.jpg height=200px>
</div>
<div class=forprint><b>Внимание!</b> В версии для
печати изображения нет.</div>
</body>
</html>
```





#### Лабораторная работа №4. Динамический html-документ

##### Цель работы:

Изучить основы клиентского скриптового языка JavaScript, работу с объектной моделью документа DOM (Document Object Model), работу с cookie, познакомиться с возможностями, предоставляемые фреймворком jQuery.

### *Задание:*

Создать клиентский скрипт на языке JavaScript, выполняющий действия в соответствии с вариантом. Использовать возможности, предоставляемые объектной моделью документа DOM, использовать фреймворк jQuery (или аналог).

### *Вариант 1:*

Сборка мозаики. Элементы мозаики перетаскиваются указателем мыши. Предусмотреть возможность автоматической сборки. Положение элементов в собранной мозаике фиксировано. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

### *Вариант 2:*

Калькулятор цвета. Отобразить таблицу, фоны ячеек которой окрашены в web-гарантированные цвета. По щелчку левой кнопки мыши на образце цвета изменяется цвет текста документа, по щелчку правой кнопки мыши — цвет фона документа, также появляется окно с шестнадцатеричным кодом цвета. Предусмотреть три поля для задания цветовых составляющих и отображения цвета, в отдельном, например, окне. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

### *Вариант 3:*

Игра «Жизнь».

Игра моделирует жизнь поколений гипотетической колонии живых клеток на прямоугольном игровом поле, которые выживают, размножаются или погибают в соответствии со следующими правилами.

Для каждого поколения (шага игры) применяются следующие правила: каждая живая клетка, количество соседей которой меньше двух или больше трёх, погибает; каждая живая клетка, у которой от двух до трёх соседей, живёт до следующего хода; каждая мёртвая клетка, у которой есть ровно три соседа, оживает. Соседи клетки – это все соседние с ней клетки по горизонтали, вертикали и диагонали, всего восемь соседей.

Правила применяются ко всему игровому полю одновременно, а не к каждой из клеток по очереди. То есть подсчёт количества соседей происходит в один момент перед следующим шагом, и изменения, происходящие в соседних клетках, не влияют на новое состояние клетки.

Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

### *Вариант 4:*

Создание эффекта анимированного текста. В тексте символ за символом изменяется цвет и размер очередного символа. Предыдущий символ становится прежним. Предусмотреть возможность выбора основного и дополнительного цвета и размера символов. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

*Вариант 5:*

За указателем мыши перемещаются часы и дата (предусмотреть возможность установки часов и даты). Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

*Вариант 6:*

Тест на скорость реакции. После щелчка по кнопке в тестовом поле случайным образом, через случайные промежутки времени появляются изображения, по которым нужно успеть щелкнуть. Попадание обозначается каким-либо образом (например, «взрывом» изображения). Тестирование можно прекратить щелчком по кнопке, но не ранее, чем через некоторый отрезок времени. Выводится результат — процент удачных щелчков. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

*Вариант 7:*

Три линейки с бегунками для каждой цветовой составляющей. Изменение положения каждого из бегунков влечет за собой изменение цвета фона документа. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

*Вариант 8:*

Калькулятор на четыре действия (с нажимающимися кнопками). Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

*Вариант 9:*

Игра «Падающие мячи». По игровому полю сверху вниз в случайном порядке падают мячи, которые нужно ловить корзиной, передвигаемой с помощью клавиатуры горизонтально вдоль нижней границы игрового поля. Игру можно начать и прекратить щелчком по соответствующей кнопке. Со временем скорость падения мячей увеличивается. После остановки игры выводится результат — процент пойманных мячей. Среди прочего использовать возможности, предоставляемые фреймворком jQuery.

*Вариант 10:*

Просмотр набора изображений со сменой подписей к изображениям с помощью кнопок «Назад» и «Далее». При просмотре первого изображения блокируется кнопка «Назад», при просмотре последнего — кнопка «Далее». Среди прочего использовать возможности, предоставляемые фреймворком jQuery.



*Фото 1. Фудзияма.*

*Порядок выполнения лабораторной работы:*

1. Создать html-документ.
2. Написать скрипт в соответствии с заданным вариантом. Для справки по языку Javascript можно использовать источники, расположенные по адресам <http://learn.javascript.ru> и <http://javascript.ru>. Для справки по фреймворку jQuery можно использовать источники, расположенные по адресам <http://jquery.com> и <http://jquery-docs.ru>.
3. Протестировать созданный документ.
  - Содержание отчета (отчет в электронном виде):
  - отчет сохранить в файле с именем АВТ-000 Иванов (лр4).doc;
  - титульный лист;
  - цель работы;
  - задание;
  - порядок выполнения лабораторной работы
  - разметка html-документа с исходным кодом скрипта;
  - скриншот html-документа;
  - выводы по работе.

*Теоретические сведения*

Теоретические материалы доступны по адресу <http://learn.javascript.ru>.

**Лабораторная работа №5. CGI-скрипт.**

*Цель работы:*

Получить практические навыки в написании и отладке CGI-скрипта на языке программирования, имеющем средства для работы с интерфейсом CGI.

*Задание:*

Во всех вариантах заданий необходимо разработать CGI-скрипт, реализующий некоторый тест, и два счетчика выполнения теста — общий счетчик, значение которого хранится на серверной стороне, и счетчик конкретного посетителя, значение которого хранится на клиентской стороне в виде cookie.

Тест должен содержать не менее трех вопросов с не менее чем тремя вариантами ответа на каждый вопрос.

Данные, введенные пользователем, пересылаются на серверную сторону, обрабатываются CGI-скриптом, который «на лету» формирует документ с результатами прохождения теста и новыми значениями счетчиков прохождения теста.

CGI-скрипт следует написать так, чтобы он мог принимать данные, присланные как методом GET, так и методом POST.

*Вариант 1:*

Проверка знаний правил дорожного движения.

*Вариант 2:*

Проверка знания таблицы умножения.

*Вариант 3:*

Психологический тест.

*Вариант 4:*

Проверка знания языка разметки гипертекста HTML.

*Вариант 5:*

Проверка знания каскадных таблиц стилей CSS.

*Вариант 6:*

Проверка словарного запаса иностранного языка.

*Вариант 7:*

Проверка знания языка программирования JavaScript.

*Вариант 8:*

Проверка знания директив web-сервера Apache.

*Вариант 9:*

Проверка знания языка программирования C++.

*Вариант 10:*

Проверка знания языка программирования PHP.

*Порядок выполнения лабораторной работы:*

1. Для выполнения лабораторной работы установить и настроить web-сервер или воспользоваться программным комплексом Denwer (<http://denwer.ru>).
2. Создать html-документ с формой.
3. Написать CGI-скрипт в соответствии с заданным вариантом.
4. Протестировать созданный CGI-скрипт (при тестировании использовать методы передачи данных GET и POST).

*Содержание отчета (отчет в электронном виде):*

- отчет сохранить в файле с именем АВТ-000 Иванов (лр5).doc;
- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- разметка html-документа;
- исходный код скрипта;
- скриншоты html-документа с исходной формой и документом, сформированным CGI-скриптом;
- выводы по работе.

*Теоретические сведения*

*CGI (Common Gateway Interface) – общий шлюзовой интерфейс*

Один из способов формирования динамических html-документов (документов, создаваемых программно на серверной стороне «на лету») заключается в использовании CGI-скриптов.

CGI — это интерфейс, используемый для связи внешней программы, работающей на серверной стороне, с web-сервером.

Интерфейс CGI разработан таким образом, что для написания серверного CGI-скрипта можно использовать любой язык программирования, имеющий средства для работы со стандартными устройствами ввода/вывода.

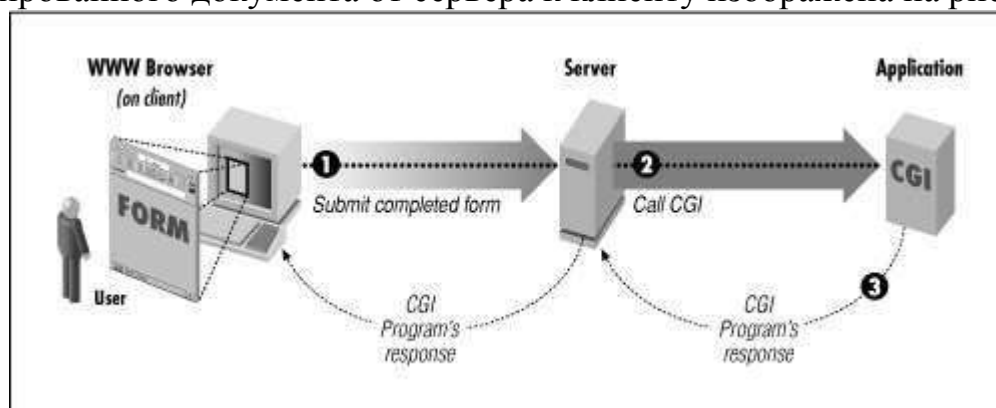
CGI-скрипт, как правило, помещается в каталог cgi (или cgi-bin) web-сервера, но это требование необязательно, так как CGI-скрипт может располагаться в любом каталоге, но при этом большинство web-серверов требуют дополнительной настройки.

CGI-скрипт, использующий CGI-интерфейс, получает информацию от клиента, обрабатывает ее, и возвращает результат (динамически сформированный html-документ, гиперссылку на существующий html-документ, графическое изображение и т.д.) Так как CGI-скрипт — это программа, она должна быть оттранслирована для той операционной системы, под управлением которой работает web-сервер.

На стороне клиента отображается форма, размеченная тегом <form>, содержащая некоторые поля для ввода данных и кнопку для отсылки данных. После заполнения полей и нажатия кнопки данные в запросе клиента пересылаются на сторону сервера, где web-сервер передает присланные данные CGI-скрипту, используя CGI.

После обработки полученных данных CGI-скрипт создает документ и передает его web-серверу, который в ответе сервера возвращает документ на сторону клиента.

Передача информации от клиента к серверу и передача сформированного документа от сервера к клиенту изображена на рисунке.



1. — клиент формирует запрос, включая в него данные, внесенные в поля формы, запрос отсылается web-серверу.
2. — web-сервер, используя CGI, передает присланные в запросе данные CGI-скрипту.
3. — CGI-скрипт на основе данных формирует документ, возвращает его web-серверу, который, в свою очередь, формирует ответ сервера, включая в него документ, созданный CGI-скриптом, и возвращает ответ клиенту.

Для создания формы используется тег <form>.

```
<form action=URL method=GET | POST>  
...  
</form>
```

Атрибут `action` определяет url GCI-скрипта, обрабатывающего присланные данные.

Атрибут `method` определяет метод передачи данных. По умолчанию используется метод `get`.

#### *Метод GET*

Метод `GET` предполагает передачу данных GCI-скрипту через переменные среды (`environment variables`), устанавливаемые на стороне сервера.

Для передачи данных, присланных методом `GET`, используется переменная `QUERY_STRING`. Значением переменной `QUERY_STRING` будет строка, содержащая данные в формате `name1=value1&name2=value2&...&nameN=valueN`, где `name` — это имя поля формы, `value` — значение, определенное пользователем для поля формы.

#### *Метод POST*

При использовании метода `POST` GCI-скрипт получает присланные данные через стандартный поток ввода.

Объем переданных данных (в байтах) можно получить через переменную окружения `CONTENT_LENGTH`.

#### *Формирование возвращаемого документа*

Вне зависимости от метода передачи данных, `GET` или `POST`, результат своей работы GCI-скрипт должен направить в стандартный поток вывода.

При формировании возвращаемого документа GCI-скрипт должен предварить документ хотя бы одним заголовком ответа сервера, определяющим `media`-тип возвращаемого документа, заголовком `Content-type`. Заголовок должен быть отделен от собственно возвращаемого документа пустой строкой, как того требует структура ответа сервера.

Чаще всего GCI-скрипт используется для создания `html`-документов на основе данных, полученных от клиента. В этом случае заголовок ответа сервера должен определять `media`-тип возвращаемого документа как текст в формате `html` (`Content-type: text/html`), за которым необходимо вывести пустую строку, отделяющую заголовок от `html`-документа.

Web-сервер возвращает результат, сформированный GCI-скриптом, клиенту, возможно дополняя его статусной строкой и другими заголовками ответа сервера.

GCI-скрипт может сформировать полный ответ (со всеми заголовками ответа сервера). В этом случае web-сервер ничего не изменяет в результате работы GCI-скрипта, только пересылает его клиенту «как есть».

Пример: на стороне клиента в поля формы вносятся имя и возраст, в зависимости от возраста возвращаются разные приветствия (рассматриваются два варианта: для методов GET и POST).

### *Метод GET*

HTML-документ, содержащий форму:

```
<html>
<form          action=http://localhost/cgi/hello.exe
method=get>
  <p>ИМЯ<input type=text name=name>
  <p>ВОЗРАСТ<input type=text name=age>
  <p><input type=submit>
</form>
```

### CGI-приложение (файл hello.cpp)

```
void main()
{
  int age;
  char *name=new char[256];
  char *query_string=new char[256];
  query_string=getenv("QUERY_STRING");

  //query_string="name=Maria&age=18"
  //из строки извлекаются подстроки "Maria" и "18"
  //и присваиваются переменным name и age
  соответственно

  cout<<"Content-type: text/html\n\n";
  cout<<"<html>";
  if(age<=16) cout<<"Привет, ";
  if(age>16) cout<<"Здравствуйте, ";
  cout<<name<<"</html>";
}
```

### *Метод POST*

HTML-документ, содержащий форму:

```
<html>
<form          action=http://localhost/cgi/hello.exe
method=post>
  <p>ИМЯ<input type=text name=name>
  <p>ВОЗРАСТ<input type=text name=age>
  <p><input type=submit>
</form>
```

### CGI-приложение (файл hello.cpp)

```
void main()
{
  int age;
```



```

char *name=new char[256];
int length=atoi(getenv("CONTENT_LENGTH"));
char * string=new char[length+1];
cin>>string;

//string="name=Maria&age=18"
//из строки извлекаются подстроки "Maria" и "18"
//и присваиваются переменным name и age
соответственно

cout<<"Content-type: text/html\n\n";
cout<<"<html>";
if(age<=16) cout<<"Привет, ";
if(age>16) cout<<"Здравствуйте, ";
cout<<name<<"</html>";
}

```

### *Работа с cookie*

Cookie устанавливаются на клиентской стороне web-сервером с помощью заголовка ответа сервера Set-Cookie.

Set-Cookie: name=value

В случае, если на клиентской стороне cookie уже установлены, информация о cookie пересылается на серверную сторону в запросе браузера-клиента в заголовке Cookie. Присланные cookie на серверной стороне присваиваются переменной окружения HTTP\_COOKIE.

### **Лабораторная работа №6. PHP-скрипт.**

#### *Цель работы:*

Получить практические навыки в написании и отладке PHP-скрипта.

#### *Задание:*

Во всех вариантах заданий необходимо разработать PHP-скрипт, реализующий некоторый тест и счетчик выполнения теста.

Тест должен содержать не менее десяти вопросов с не менее чем тремя вариантами ответа на каждый вопрос. На некоторые вопросы может предлагаться несколько правильных вариантов ответов. Вопросы должны быть разделены на две темы.

Результаты теста должны отображаться в браузере и сохраняться в файле, доступном по ссылке на странице с результатами теста. Кроме результатов на странице и в файле должны быть указаны дата и время прохождения теста.

#### *Вариант 1:*

Проверка знаний правил дорожного движения.

#### *Вариант 2:*

Проверка знания таблицы умножения.

*Вариант 3:*

Психологический тест.

*Вариант 4:*

Проверка знания языка разметки гипертекста HTML.

*Вариант 5:*

Проверка знания каскадных таблиц стилей CSS.

*Вариант 6:*

Проверка словарного запаса иностранного языка.

*Вариант 7:*

Проверка знания языка программирования JavaScript.

*Вариант 8:*

Проверка знания директив web-сервера Apache.

*Вариант 9:*

Проверка знания языка программирования C++.

*Вариант 10:*

Проверка знания языка программирования PHP.

*Порядок выполнения лабораторной работы:*

1. Для выполнения лабораторной работы установить и настроить web-сервер Apache и интерпретатор PHP (интерпретатор PHP установить как модуль web-сервера Apache).
2. Создать html-документ с формой.
3. Написать PHP-скрипт в соответствии с заданным вариантом.
4. Протестировать созданный PHP-скрипт.

*Содержание отчета (отчет в электронном виде):*

- отчет сохранить в файле с именем АВТ-000 Иванов (лрб).doc;
- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- разметка html-документа;
- исходный код скрипта;
- скриншоты html-документа с исходной формой и документом, сформированным PHP-скриптом;
- файл с результатами тестирования;
- выводы по работе.

*Теоретические сведения*

*Установка интерпретатора PHP как модуля web-сервера Apache*

Интерпретатор PHP может быть установлен для работы в двух режимах: как модуль web-сервера Apache или как обработчик CGI-скриптов.

Для установки интерпретатора PHP как модуля web-сервера Apache достаточно распаковать zip-архив с дистрибутивом, например, на диск C:\php и создать копию файла php.ini-production с именем php.ini в той же папке.

В файле `php.ini` можно выполнить настройки путем изменения параметров соответствующих директив.

Директива `error_reporting` задает уровень протоколирования ошибки. Параметр директивы может быть либо числом, либо именованной константой. Параметр `E_ALL` позволяет отображать предупреждения и ошибки всех уровней.

```
error_reporting = E_ALL
```

Директива `extension` позволяет загрузить необходимые динамические расширения.

```
extension=php_gd2.dll ;для работы с графической библиотек
```

```
extension=php_mysql.dll ;для работы с СУБД MySQL
```

Директива `display_errors` позволяет выводить сообщения об ошибках на экран вместе с остальным выводом, либо скрывать сообщения об ошибках от пользователя. Для отладки скриптов рекомендуется использовать директиву `display_errors` с параметром `On`.

```
display_errors = On
```

После отладки скриптов предупреждения и сообщения об ошибках можно скрывать от пользователя, выводя их в файл, расположенный на стороне сервера. Директива `error_log` задает расположение файла с предупреждениями и сообщениями об ошибках.

```
display_errors = Off  
error_log = c:\php\phperror.log
```

Директива `short_open_tag` определяет сокращенную или полную форму записи тега для вставки `php`-скрипта в `html`-разметку.

```
short_open_tag = Off;<?php ... ?> и <script> ... </script>
```

или

```
short_open_tag = On ;дополнительно <? ... ?>
```

Для настройки `web`-сервера `Apache` в основной конфигурационный файл `httpd.conf` следует добавить директивы `LoadModule`, `AddHandler` и `PHPIniDir`.

```
LoadModule php5_module "c:/php/php5apache2_2.dll"  
AddHandler application/x-httpd-php .php  
PHPIniDir "c:/php"
```

*Установка интерпретатора PHP как обработчика CGI-скриптов*

Для установки интерпретатора `PHP` как обработчика `CGI`-скриптов настройки файла `php.ini` выполняются также, как было описано выше, для настройки `web`-сервера `Apache` в основной конфигурационный файл `httpd.conf` следует добавить директивы `ScriptAlias`, `AddType` и `Action`.

```
ScriptAlias /php/ "c:/php/"  
AddType application/x-httpd-php .php  
Action application/x-httpd-php "/php/php-cgi.exe"
```

Теоретические материалы также доступны по адресу <http://www.php.ru/learnphp>, <http://phpclub.ru/manrus>.

### **Лабораторная работа №7. Графическая библиотека PHP GD**

#### *Цель работы:*

Получить практические навыки в использовании графической библиотеки PHP GRAPHICS DRAW (GD).

#### *Задание:*

Во всех вариантах заданий необходимо разработать PHP-скрипт, использующий возможности графической библиотеки PHP GD.

#### *Вариант 1:*

Форма для задания в аналитическом виде функции одного аргумента, начального и конечного значений аргумента и вывода графика функции (для оперирования аналитическим выражением функции использовать функцию `create_function()`).

#### *Вариант 2:*

Форма для голосования с выводом результатов в виде круговой диаграммы с указанием долей в процентах.

#### *Вариант 3:*

Лист календаря с текущим месяцем (с указанием дней недели). Для листа календаря использовать готовые изображения для каждого из месяцев.

#### *Вариант 4:*

Форма для голосования с выводом результатов в виде столбчатой 3d диаграммы с указанием долей в процентах.

#### *Вариант 5:*

Логотип PHP.

#### *Вариант 6:*

Форма для голосования с выводом результатов в виде столбчатой гистограммы с указанием долей в процентах.

#### *Вариант 7:*

Галерея изображений. В каталоге хранится набор файлов с полноразмерными изображениями любого формата, сформировать документ с миниатюрами-гиперссылками изображений (preview) формата jpeg.

#### *Вариант 8:*

Графический счетчик посещения сайта. Фон для значения счетчика выбирается случайным образом из набора графических файлов.

#### *Вариант 9:*

Аналоговые часы с фиксированным текущим значением временем.

#### *Вариант 10:*

Форма для голосования с выводом результатов в виде кольцевой диаграммы с указанием долей в процентах.

#### *Порядок выполнения лабораторной работы:*

1. Для выполнения лабораторной работы установить программный комплекс Denwer.

2. Написать PHP-скрипт в соответствии с заданным вариантом.
3. Протестировать созданный PHP-скрипт.

*Содержание отчета (отчет в электронном виде):*

- отчет сохранить в файле с именем АВТ-000 Иванов (лр7).doc;
- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- исходный код скрипта;
- скриншот изображения, сформированного PHP-скриптом;
- выводы по работе.

*Теоретические сведения*

*Графическая библиотека PHP GD*

Дополнительные возможности PHP приобретает за счет использования расширений, позволяющих решать стоящие перед web-разработчиком задачи. Одним из таких расширений является графическая библиотека GD, предназначенная для динамической работы с растровыми изображениями.

Библиотека поддерживает практически все существующие форматы графики для использования в WWW: PNG, JPEG, GIF, ICO и различные методы работы с графическими файлами (применение фильтров, текст, изменение размера и прочее), позволяет создавать изображения, состоящие из линий, дуг, текста (включая программный выбор шрифтов) и других изображений, а также использовать различные цвета.

### **Лабораторная работа №8. Технология AJAX**

*Цель работы:*

Получить практические навыки в использовании технологии AJAX.

*Задание (типовое):*

Во всех вариантах заданий необходимо разработать скрипт, использующий возможности технологии AJAX.

Создать документ с двумя раскрывающимися списками. Перечень вариантов в первом раскрывающемся списке разметить с помощью языка разметки гипертекста HTML. Выбор варианта в первом раскрывающемся списке определяет тот или иной набор вариантов для выбора во втором раскрывающемся списке. Варианты для выбора во втором раскрывающемся списке должны быть сформированы с использованием технологии AJAX. Итоговые результаты выбора в обоих раскрывающихся списках обрабатываются серверным скриптом и отображаются на клиентской стороне.

При выполнении лабораторной работы возможно применение функций библиотеки jQuery для использования возможностей технологии AJAX.

*Порядок выполнения лабораторной работы:*

1. Разметить html-документ.
2. Написать скрипт с использованием технологии AJAX.
3. Протестировать созданный скрипт.

Содержание отчета (отчет в электронном виде):

- отчет сохранить в файле с именем АВТ-000 Иванов (лр8).doc;
- титульный лист;
- цель работы;
- задание;
- порядок выполнения лабораторной работы
- исходный код скрипта;
- выводы по работе.

*Теоретические сведения*

AJAX (Asynchronous JavaScript and XML) — новый подход к построению интерактивных пользовательских интерфейсов web-приложений, заключающийся в «фоновом» обмене данными браузера с web-сервером

В результате при обновлении данных документ не перезагружается полностью и web-приложения могут быть сделаны более быстрыми и удобными.

AJAX не самостоятельная технология, а концепция использования нескольких смежных технологий: технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью и DHTML для динамического изменения содержимого документа.

AJAX — аббревиатура, обозначающая подход к созданию web-приложений с помощью следующих технологий: стандартизированное представление силами HTML и CSS, динамическое отображение и взаимодействие с пользователем с помощью DOM, обмен и обработка данных в виде XML, асинхронные запросы с помощью объекта XMLHttpRequest, использование JavaScript.

Впервые термин AJAX появился в феврале 2005 года, когда пришлось как-то назвать новый набор технологий, предлагаемый клиенту.

Плюсы технологии: экономия трафика (использование AJAX позволяет значительно сократить трафик при работе с web-приложением, благодаря тому, что часто вместо загрузки всей страницы достаточно загрузить только небольшую изменившуюся часть), уменьшение нагрузки на сервер (AJAX позволяет несколько снизить нагрузку на сервер, к примеру, в Gmail, когда помечаются прочитанные письма, серверу достаточно внести изменения в базу данных и отправить клиентскому скрипту сообщение об успешном выполнении операции, вместо необходимости повторно создавать страницу и отсылать ее клиенту), увеличение реакции интерфейса (поскольку нужно загрузить только изменившуюся часть, то пользователь видит результат своих действий быстрее).

Минусы технологии: интеграция со стандартными инструментами браузера (динамически создаваемые страницы не регистрируются браузером

в истории посещения страниц, поэтому не работает кнопка «Назад», предоставляющая пользователям возможность вернуться к просмотренным ранее страницам), другой недостаток изменения контента страницы при постоянном URL заключается в невозможности сохранения закладки на желаемый материал. Частично решить эти проблемы можно с помощью динамического изменения идентификатора фрагмента (части URL после #), что позволяют многие браузеры, динамически загружаемое содержание недоступно поисковикам (поисковые машины не могут выполнять JavaScript, поэтому разработчики должны позаботиться об альтернативных способах доступа к содержимому сайта), старые методы учета статистики сайтов становятся неактуальными (многие сервисы статистики ведут учет просмотров новых страниц сайта, для сайтов страницы которых широко используют AJAX, такая статистика теряет актуальность).

В стандартном web-приложении обработкой всей информации занимается сервер, тогда как браузер отвечает только за взаимодействие с пользователем, передачу запросов и вывод поступившего HTML

В AJAX-приложении между пользователем и сервером появляется еще один посредник — движок AJAX. Он определяет, какие запросы можно обработать «на месте», а за какими необходимо обращаться на сервер.

Если раньше на каждый запрос сервер выдавал новую страницу, то теперь он отправляет лишь те данные, которые нужны клиенту, а HTML-разметку из них прямо в браузере формирует движок AJAX.

Асинхронность проявляется в том, что далеко не каждый щелчок пользователя доходит до сервера, причем обратное тоже справедливо — далеко не каждая реакция сервера обусловлена запросом пользователя.

Большую часть запросов формирует движок AJAX, причем его можно написать так, что он будет загружать информацию превентивно, предугадывая действия пользователя.

Качественная нагрузка на сервер меняется — если раньше запросов было мало, но каждый из них требовал значительных ресурсов (серверу нужно получить информацию из БД, сформировать из нее web-страницу и отдать браузеру), то теперь задача сервера упрощается (формировать web-страницы не нужно, объем передаваемых данных меньше), но запросов обрабатывать приходится больше.

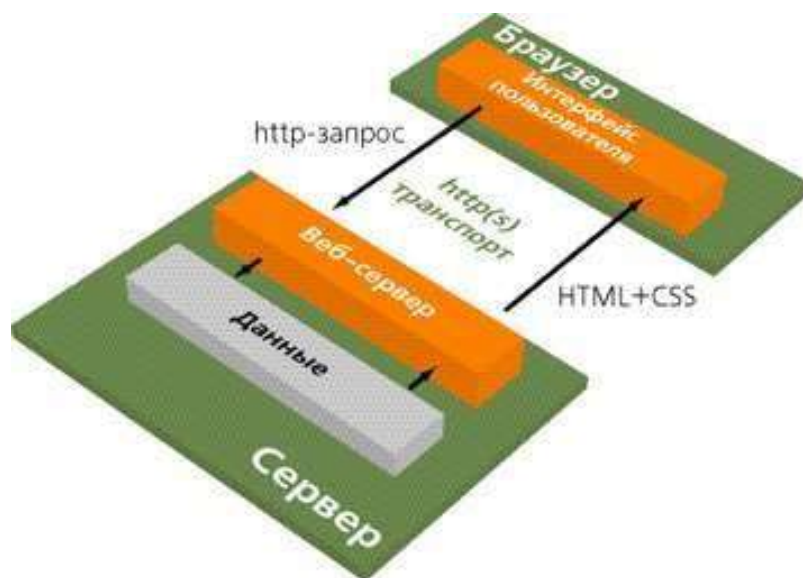


Рисунок 48 Классическое web-приложение

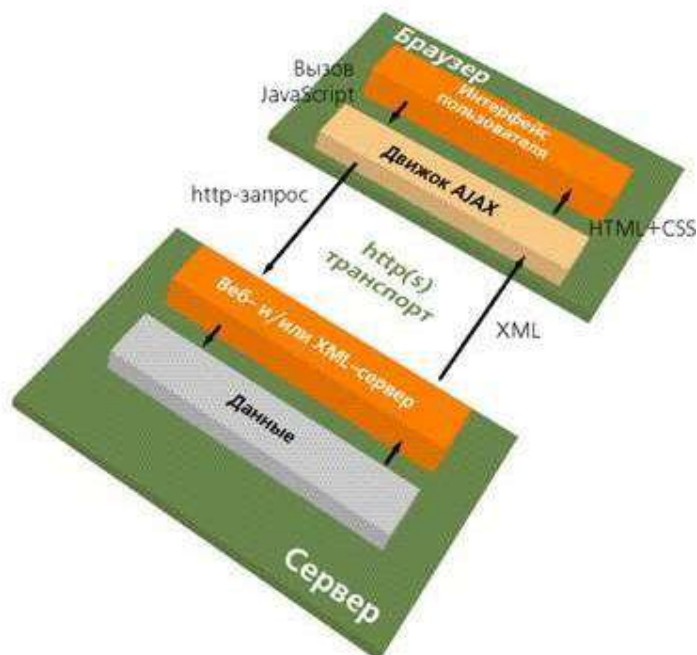


Рисунок 49 Web-приложение, использующее технологию AJAX

## Задание на курсовое проектирование

### Цель работы:

- Разработать структуру web-узла и реализовать динамический web-узел.
- Web-узел должен состоять из не менее чем 10 страниц.
- Разработать удобную систему навигации по web-узлу.
- Реализовать хранение информационного наполнения некоторых страниц web-узла в базе данных, предусмотреть возможность



удаленного редактирования информационного наполнения страниц web-узла.

- Реализовать разграничение категорий пользователей узла (минимум две категории) с регистрацией пользователей, хранением регистрационной информации в базе данных, авторизацией и организацией доступа пользователей к web-узлу в соответствии с определенными правами.

#### **Исходные данные:**

Язык разметки гипертекста HTML5, каскадные таблицы стилей CSS3, клиентский скриптовый язык, серверный скриптовый язык, программный комплекс Denwer (web-сервер Apache, интерпретатор PHP, СУБД MySQL), браузеры.

#### **Содержание пояснительной записки:**

- титульный лист;
- задание на курсовую работу;
- содержание;
- введение;
- постановка задачи;
- обзор современного состояния web-технологий;
- структура web-узла;
- дизайн web-узла;
- скрипты, разработанные при создании web-узла;
- заключение;
- список использованных источников (оформленный в соответствии с ГОСТ);
- приложения (исходные коды, в случае большого объема основные, на усмотрение автора).

#### **Задание на самостоятельную работу студента**

##### **Цель работы:**

- Разработать структуру web-узла и реализовать динамический web-узел.
- Информационное наполнение web-узла — по выбору студента (выбор студента согласовывается с преподавателем).
- Web-узел должен состоять из не менее чем 5 страниц.
- Разработать удобную систему навигации по web-узлу.
- Реализовать хранение информационного наполнения некоторых страниц web-узла в базе данных, предусмотреть возможность удаленного редактирования информационного наполнения страниц web-узла.

##### **Исходные данные:**

Язык разметки гипертекста HTML5, каскадные таблицы стилей CSS3, клиентский скриптовый язык, серверный скриптовый язык, программный

комплекс Denwer (web-сервер Apache, интерпретатор PHP, СУБД MySQL), браузеры.

**Содержание пояснительной записки:**

- титульный лист;
- задание на СРС;
- содержание;
- введение;
- обзор современного состояния web-технологий;
- описание структуры созданного web-узла;
- описание дизайна созданного web-узла;
- описание скриптов, разработанных при создании web-узла;
- список использованных источников (оформленный в соответствии с ГОСТ);
- приложения (исходные коды).

**Темы для выполнения СРС и курсового проектирования**

1. Web-сайт "Виды компьютерных преступлений и методы защиты от них"
2. Web-сайт "Синтезирование речи"
3. Web-сайт "Бронирование билетов в кинотеатре"
4. Web-сайт "Интеллектуальный анализ данных"
5. Web-сайт аэропорта
6. Web-сайт "Система обмена сообщениями"
7. Web-сайт гостиницы
8. Web-сайт "Система комплексного бронирования путешествий" (часть 1)
9. Web-сайт боулинга
10. Web-сайт музыкальной группы
11. Web-сайт "Студия звукозаписи"
12. Web-сайт "Мониторы"
13. Web-сайт "Система комплексного бронирования путешествий" (часть 2)
14. Web-сайт "Художественная галерея"
15. Web-сайт "Нейрокомпьютеры"
16. Web-сайт "Концерты музыкальных исполнителей"
17. Web-сайт "Мой ЮУрГУ" (часть 2)
18. Web-сайт "Фреймворки JavaScript"
19. Web-сайт "HTML-редакторы"
20. Web-сайт "Web-серверы"
21. Web-сайт "Протокол IPv6"
22. Web-сайт " "Протокол IPv4""
23. Web-сайт "Продажа недвижимости"
24. Web-сайт "Системы счисления"
25. Web-сайт "Кодирование символов"

26. Web-сайт "Мой НГТУ" (часть 1)
27. Web-сайт "Продажа протезов конечностей"
28. Web-сайт "Магазин косметики "THEFACESHOP"
29. Web-сайт по продаже букетов
30. Web-сайт "Японский язык"
31. Web-сайт "Интернет – всемирная паутина"
32. Web-сайт "Компьютерная вирусология"
33. Web-сайт "Ногтевой сервис"
34. Web-сайт "Гитарное сообщество"
35. Web-сайт "Сообщество футбольных фанатов"
36. Web-сайт "Сообщество аквариумистов"
37. Web-сайт "Информационная разведка"
38. Интернет-магазин одежды (часть 2)
39. Web-сайт "Школа бега"
40. Мобильное приложение на Framework7
41. Web-сайт "Сервис фитнес-инструкторов"
42. Интернет-магазин одежды (часть 1)
43. Web-сайт "Велосипеды"
44. Web-сайт "Сообщество любителей творчества художника Коровина К.А."
45. Web-сайт "Змейка" (часть 1)
46. Web-сайт "Змейка" (часть 2)
47. Web-сайт "Принтеры"
48. Web-сайт "Интерактивный каталог книг"
49. Web-сайт компании "ТехМаксимум"
50. Web-сайт "Поисковые системы"
51. Web-сайт "Декларативное программирование"
52. Web-сайт "Tabletop RPG "World of Darkness"
53. WSM (work site marketing) — система страхования на рабочем месте
54. Web-сайт "Облачные вычисления"
55. Web-сайт "Параллельное программирование"
56. Web-сайт "Web 2.0"

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

№	Вид литературы	Наименование ресурса в электронной форме	Библиографическое описание
1	Основная литература	Электронно-библиотечная система издательства Лань	Никулина, М. В. Прикладное программирование : учебное пособие / М. В. Никулина. — Нижний Новгород : ВГУВТ, 2016. — 60 с. — Текст : электронный // Лань : электронно-библиотечная система. <a href="https://e.lanbook.com/book/97173">https://e.lanbook.com/book/97173</a>
2	Основная литература	Электронно-библиотечная система издательства Лань	Пай, П. Реактивное программирование на C++ / П. Пай, П. Абрахам ; перевод с английского В. Ю. Винника. — Москва : ДМК Пресс, 2019. — 324 с. — ISBN 978-5-97060-778-7. — Текст : электронный // <a href="https://e.lanbook.com/book/131698">https://e.lanbook.com/book/131698</a>
3	Дополнительная литература	Электронно-библиотечная система издательства Лань	Сайбель, П. Практическое использование Common Lisp / П. Сайбель. — Москва : ДМК Пресс, 2015. — 488 с. — ISBN 978-5-94074-627-0. — Текст : электронный // Лань : электронно-библиотечная система. <a href="https://e.lanbook.com/book/5868">https://e.lanbook.com/book/5868</a>