МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра прикладної математики

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт
з кредитного модуля

«Програмування та підтримка Web-застосувань»

Затверджено на засіданні
кафедри прикладної
математики

Протокол № 9

від « 11 » червня 2014 р.

Завідувач кафедри
прикладної математики

_____ О.А.Молчанов

«____»_____ 2014 р.

Київ – 2014

Лабораторная работа №1.

« Разработка иерархической структуры разделов Web-узла (карты Web-узла)»

Введение

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.

1. Цель работы

Овладение навыками проектирования веб-сайта.

2. Задание на лабораторную работу

1. Согласовать с преподавателем тему сайта. (см. Приложение 1)
2. Разработать модель и структуру сайта.
3. Разработать макеты размещения текстовых и графических материалов с помощью графического редактора (MS Visio, MS Paint brush). См. Приложение 3.
4. Разработать цветовую схему сайта.
5. Сделать отчет, согласно требованиям.

3. Требования к работе

Количество уровней иерархической структуры разделов (переходов по разделам) Web-узла не должно превышать 5. Количество ссылок на каждом уровне не должно превышать 5, за исключением корневого раздела Web-узла. Должен быть определен состав Web-страниц для каждого раздела.

4. Требования к отчету

Отчет должен содержать следующую информацию:

1. Титульный лист (см. Приложение 2)
2. Цель работы.
3. Задание .
4. Выбранную структуру сайта с описанием.
5. Снимки макетов страниц, сделанных с помощью графического редактора.
6. Выводы.

5. Теоретические сведения

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 структу-

ра, 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

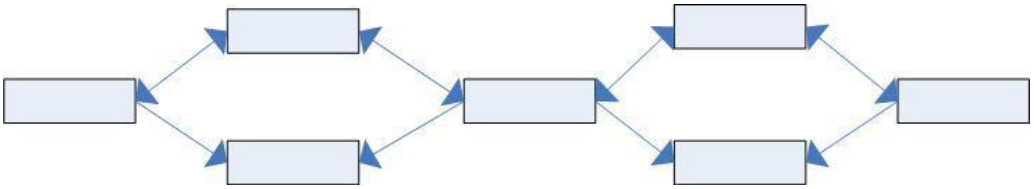
Она способствует упорядоченному продвижению по основной части информации.

Такая форма хороша для презентаций. Для такой формы организации в силу предсказуемости событий возможно осуществить предварительную загрузку (preload) или предварительная выборка (prefeatch) 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.

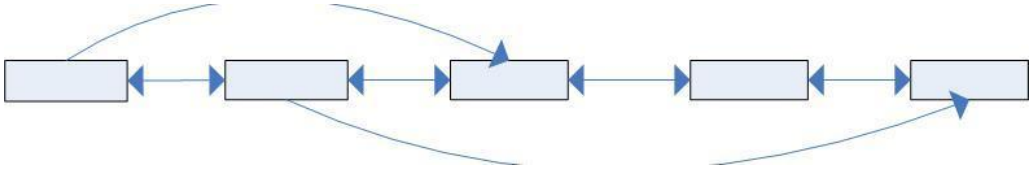
Линейная форма организации с альтернативами имитирует интерактивность, предоставляя два или более вариантов перехода со страницы, которые, в конечном счете, оканчиваются указанием пользователю вернуться на другую страницу в последовательности.



Применением такой формы является сайты –вопросники, с ответами, требующими ответа «да-нет» и передвижения на следующую страницу основывается на ответе. В этом случае создается иллюзия интерактивности.

Линейная форма со свободой выбора

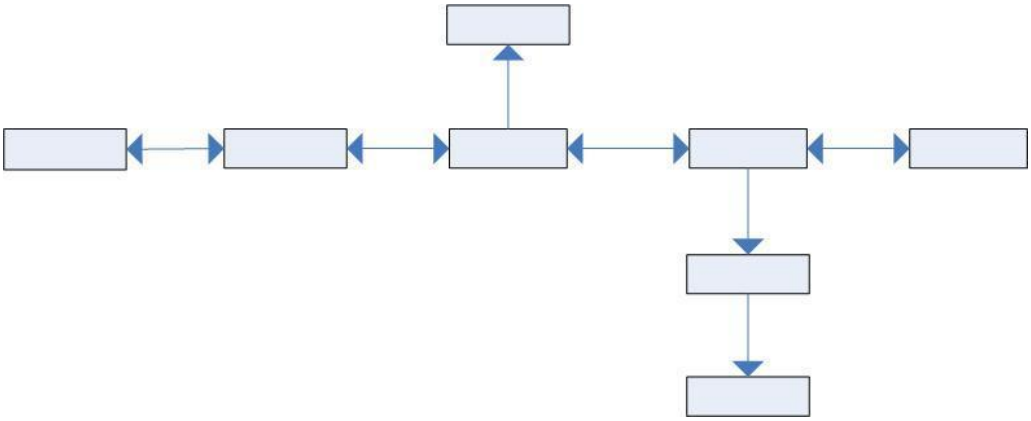
Такая форма хорошо работает, когда необходимо сохранить общее направление, но при этом нужно добавить лечение вариации, такие как пропуск страниц.



Такую форму называют еще линейной структурой с переходами вперед. Например, презентации велосипедов или машин. Например некоторые общие страницы могут быть пропущены и сразу осуществляться переход на страницы с описанием «горных» велосипедов или какого-нибудь класса машин.

Линейная форма с боковыми ответвлениями

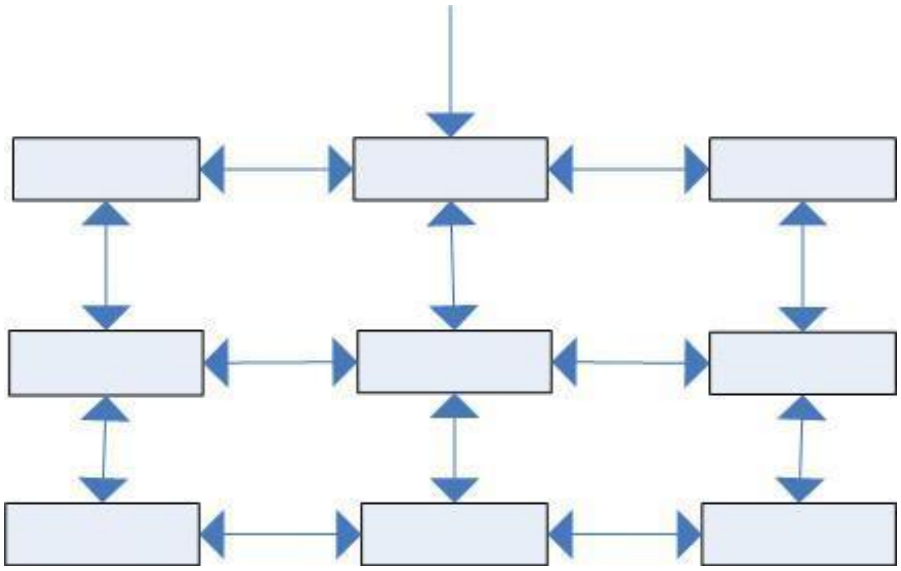
Такая форма позволяет контролировать отклонения от основного направления, но такая структура все же принуждает его вернуться к основному пути, сохраняя первоначальное движение.



Боковое ответвление в линейной последовательности подобно боковой врезке в журнальной статье. Оно позволяет не уводить пользователя далеко от основной темы, и при этом небольшой объем дополнительной информации расширяет кругозор. Хотя, если много разветвлений, такая структура превращается в дерево.

Решетка

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.

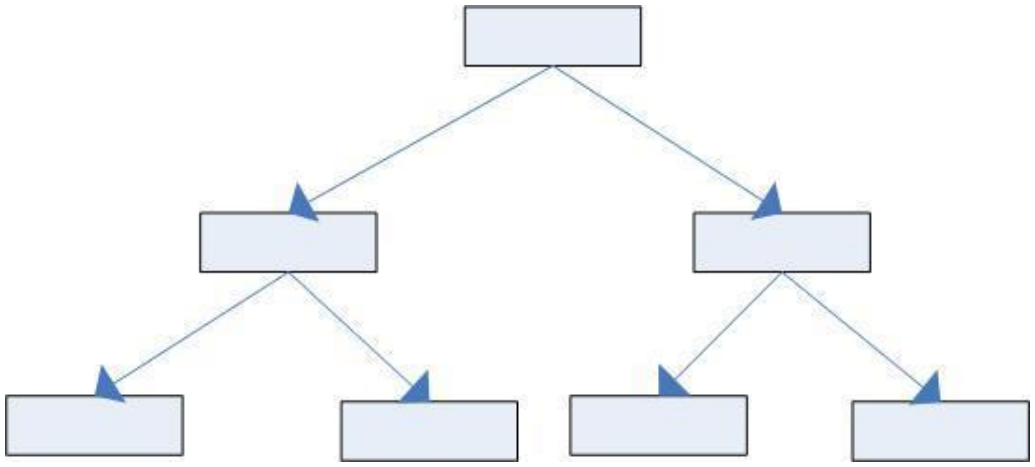
Иерархия

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

Иерархические формы делятся на:

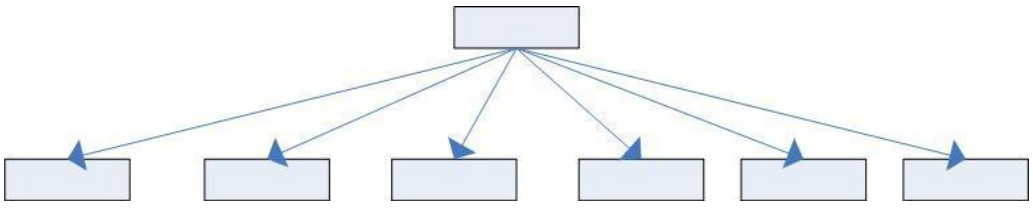
1. Узкие деревья

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



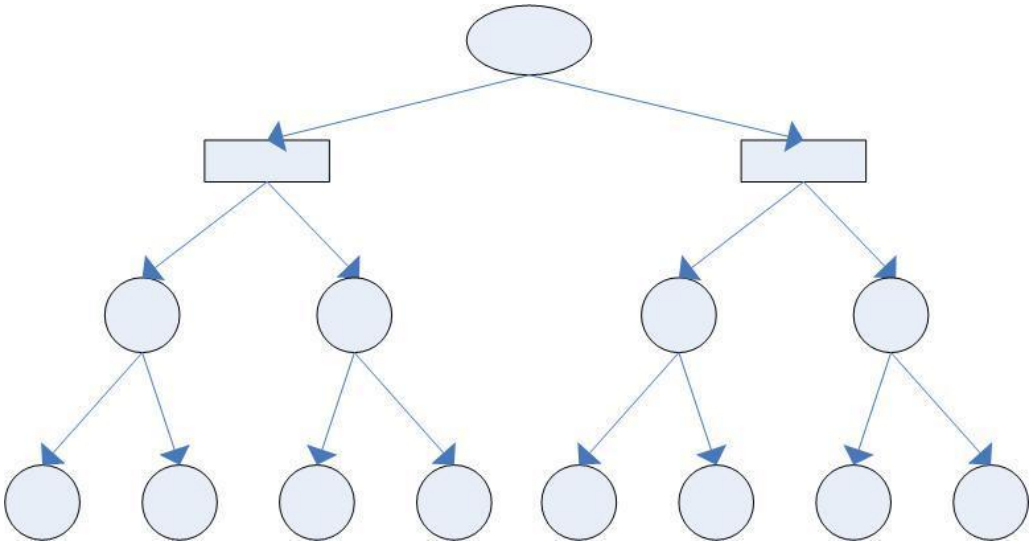
2. Широкие деревья

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



3. Запутанные деревья

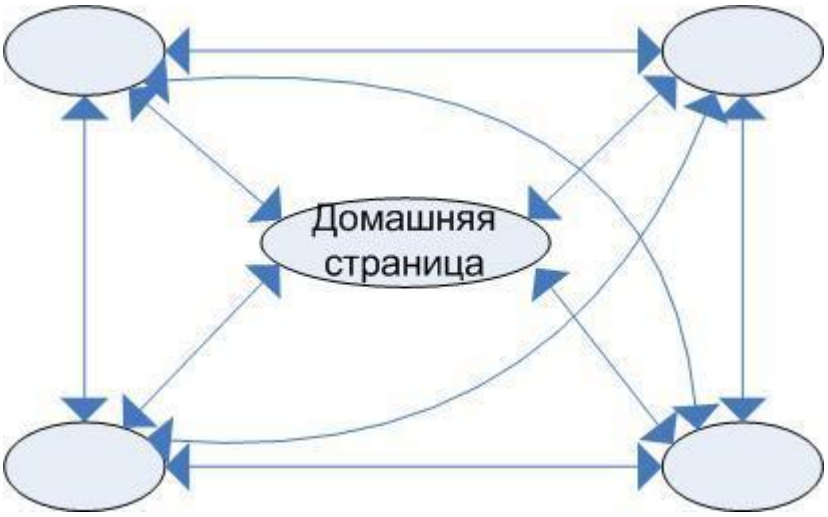
Стандартное дерево используется редко.



С домашней страницы перепрыгиваем в разделы, но в разделах существуют обратные и перекрестные 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.

4. Полное связывание

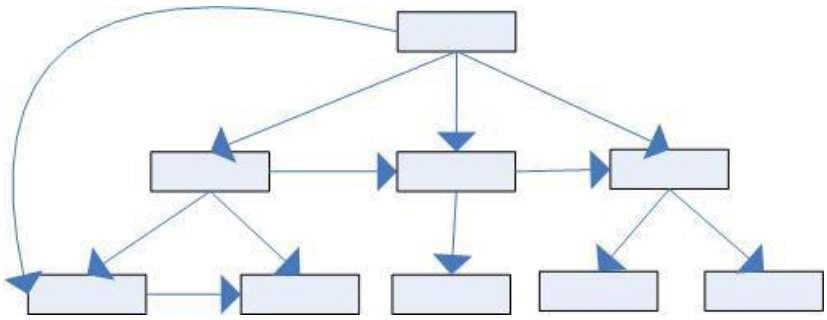
В этом сайте каждая страница связана ссылкой с каждой страницей этого сайта. Количество ссылок = (количество страниц)*(количество страниц – 1). Такой сайт не является лучшим выбором. Большинство сайтов склонны использовать частичное связывание.



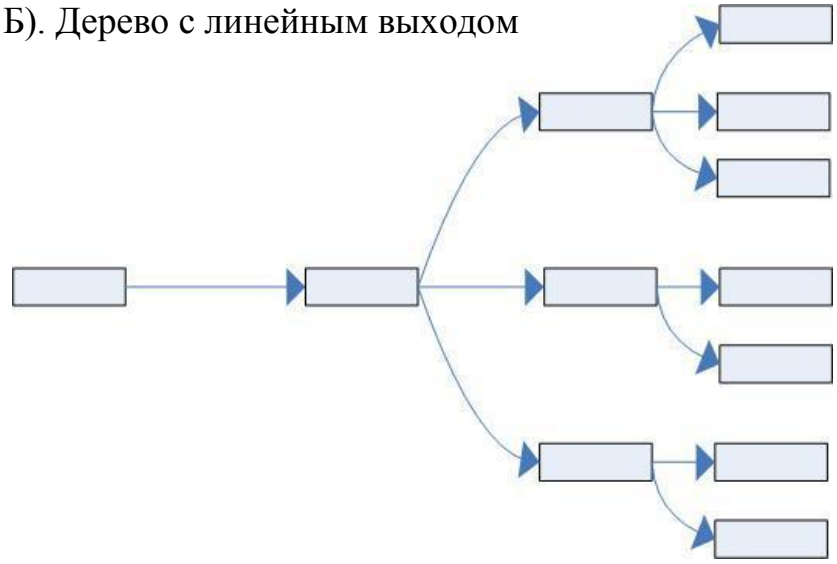
5. Смешанные формы или смешанная иерархия

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



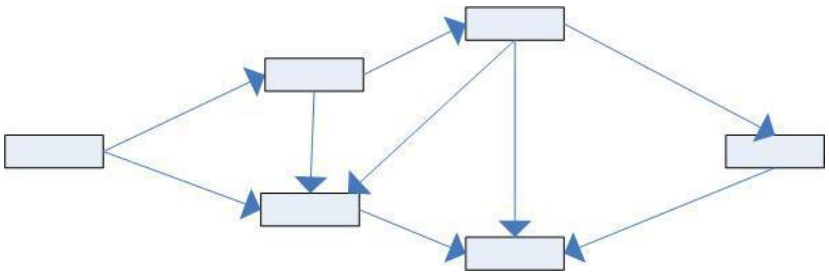
6. Модель "ступица и спицы"

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 содержимого, 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.



Лабораторная работа №2

« Принципы текстовой разметки и язык HTML 4»

Введение

Язык разметки гипертекста

HTML (HyperText Markup Language) - это язык разметки гипертекста. **HTML - представляет собой систему разметки веб-страниц и определяет, где и как на web-странице сайта, отображать отдельные элементы текст, рисунки, ссылки и т.д.** Т.е. при необходимости изменить, например текст (размер, цвет, наклон и т.д.) или вставить картинку в нужном месте добавляются специальные символы называемые ***тегами***.

Теги не отображаются браузерами. Вы будете видеть только результат, т.е. измененный текст. ***Теги всегда заключаются в угловые скобки.*** Например, видимая запись на web-странице: **текст**, означает, что текст, находящийся между тегами и будет отображаться **жирным начертанием**. Аналогично с помощью тегов создаются списки, таблицы, добавляются изображения и другие элементы веб-страницы.

По сути дела **код веб-страницы** и есть набор **Тегов** в сочетании с обычным текстом, картинками и другими файлами. Иногда говорят **Исходный код**.

Исходный код самой веб-страницы является обычным текстовым документом, который можно написать в обычном **Блокноте,** входящем в состав любой **Windows** -подобной операционной системы и состоит всего из двух основных компонентов, которые перемежаются между собой:

- текст, который и надо показать в окне браузера;
- теги, они добавляют рисунки, устанавливают ссылки, меняют вид текста и делают еще множество других полезных вещей.

Этот текстовый документ сохраняют с расширением **.html** или **.htm** и получают готовую веб-страницу.

Таким образом, освоение языка HTML сводится к изучению основных тегов (их около двух десятков) и правил их использования этих тегов.

Для элементов разметки существуют правила их использования, которые установлены спецификацией HTML. Этот документ регламентирует порядок применения тегов, их вложенность и принцип работы. Знание спецификации и ее соблюдение помогает избежать типичных ошибок и позволяет создать универсальный код, одинаково хорошо работающий на различных устройствах и в разных браузерах.

Полезный совет: рекомендуется пользоваться продвинутой и к тому же бесплатной программой **Notepad++.**Функции программы те же, но все более наглядней и удобнее например есть подсветка кода, т.е. теги, параметры тегов и текст отображаются разными цветами, очень удобно, написал тег неправильно сразу изменился цвет очень наглядно и удобно, короче почувствуйте разницу :) .

Полезный совет: В настоящее время существует довольно много визуальных редакторов для создания сайтов, перетаскивая мышкой какой-то элемент, Вы сразу получаете готовый код web-страницы. *Удобно Но… если вдруг, что то на Вашей веб-странице «разъедется» или просто перестанет работать, согласитесь не зная исходного кода Вы не сможете это исправить. Так что знание кода – это Ваш фундамент, и его надо знать. Поэтому не ленитесь – набирайте код (хотя бы вначале) ручками.*

Браузеры

Сам HTML-файл размещается на сервере в интернете и для его просмотра необходима специальная программа — *браузер*, например *Internet Explorer (IE),Mozilla Firefox, Opera* . В задачу браузера входит подключение к удаленному серверу, получение HTML кода web-страницы и его (кода) интерпретация (отображение на экране монитора) согласно спецификации. Иными словами, *браузер преобразует HTML-документ в веб-страницу, которую мы и наблюдаем, когда «ходим» по Интернету.*

Заметим так же, что браузеры вполне корректно работают и на локальном компьютере (не подключенном к Интернету), это позволяет легко проверять и отлаживать работу созданных Вами HTML-документов (сайтов). Для этих целей используется специальное программное обеспечение - Локальный сервер. Например, *Российский пакет Локального сервера - Денвер,* но об этом в следующих лабораторных работах.

1. **Цель работы**

- Приобрести начальные навыки создания простейших Internet-документов.
- Научиться выполнять форматирование созданных Web-страниц.

2. **Задание на лабораторную работу**

1. Выполнить текстовое наполнение сайта (контент), исходя из выбранной модели сайта. Для набора текста использовать Notepad или Notepad++.
2. Определить гипертекстовые ссылки внешние и между страницами.
3. Определить стиль заголовков и разместить на страницах сайта.

4. Реализовать цветовую схему для страниц (фон, разделительные линии , кроме графики).

3. Требования к работе

Набрать текст текстовым редактором. Расставить при форматировании необходимые теги, атрибуты, заголовки, ссылки. Все это должно быть оформлено в выбранной Вами цветовой гамме. Проверить любым браузером.

4. Требования к отчету

Отчет должен содержать следующую информацию:

1. Титульный лист (см. Приложение 2)
2. Цель работы.
3. Задание .
4. Описание сайта (содержимого).
5. Размеченный с помощью HTML текст страниц сайта.
6. Скриншоты страниц сайта в браузере.

5. Теоретические сведения

Общие положения языка HTML

Язык **HTML** (HyperText Markup Language, язык разметки гипертекста) — это язык, на котором создаются Web-страницы.

Web-страницы (HTML-документы) могут просматриваться различными типами Web-броузеров. Когда документ создан с использованием HTML, Web-броузер может интерпретировать HTML для выделения различных элементов документа и первичной их обработки. Использование HTML позволяет форматировать документы для их представления с использованием шрифтов, линий и других графических элементов на любой системе, их просматривающей.

Web-страницы могут быть созданы с помощью

1) обычного текстового редактора;

2) редактора, способного сохранять в формате HTML;

3) специализированного редактора;

4) специализированной системы.

HTML-документы сохраняются на диске как обычные текстовые документы в формате ASCII. Для распознавания Web-страниц по их именам общепринято обозначать такие файлы использованием расширений .HTML.

Кроме полезного текста в HTML-документах используются специальные *управляющие последовательности символов — теги.*

Структура тега имеет следующий вид:

Парный тег:

<ТЕГ атрибут1=значение1...>

текст

другие конструкции

</ТЕГ>

Одиночный тег:

<ТЕГ атрибут1=значение1...>

текст

другие конструкции

Чаще всего теги используются попарно, окружая размеченные фрагменты текста. Такие теги называются **контейнерами**. Закрывающий тег отличается от начального только присутствием символа "/", добавляемого перед именем тега.

При интерпретации тегов браузер не делает различия между строчными и прописными буквами. Поэтому сами теги можно набирать на любом регистре.

Атрибуты (параметры) для тегов часто является необязательной величиной и их можно пропускать.

Структура HTML-документа

Когда Web-браузер получает документ, он определяет, как документ должен быть интерпретирован. Самый *первый тег*, который встречается в документе, должен быть тегом **<HTML>.** Данный тег сообщает Web-браузеру, что документ написан с использованием HTML.

Минимальный HTML-документ имеет структуру:

<HTML>

...тело документа...

</HTML>

Теги **<HTML> и </HTML>** заключают внутри себя все элементы HTML-кода, указывая, что используется язык HTML.

Тем не менее, принято выделять *заголовочную часть и тело* Web-документа.

Теги **<HEAD> и </HEAD>** обозначают *заголовочную часть* Web-документа. Как правило, заголовочная часть содержит название документа, метатеги с информацией для индексирования и некоторые общие установки для данного документа.

Тег заголовочной части документа должен быть использован сразу после тега <HTML> и более нигде в теле документа.

Теги **<BODY> и </BODY>** обрамляют **тело** (основную часть) документа. Здесь размещается основная смысловая текстовая и графическая информация. Тело документа отображается в окне браузера.

Разделение документа на заголовочную часть и тело имеет лишь смысловую нагрузку. Текст, приведенный в любой из этих частей, на экране выглядит совершенно одинаково.

В общем виде Web-документ имеет следующую структуру:

<HTML>

<HEAD>

<TITLE>

… текст для строки заголовка броузера

</TITLE>

</HEAD>

<BODY>

… тело документа (в окне броузера)

</BODY>

</HTML>

Внутри контейнера <HEAD> может использоваться тег **<TITLE>**, как показано выше. Большинство Web-браузеров отображают содержимое этого тега *в строке заголовка окна* Web-браузера, содержащего документ, и в файле закладок, если он поддерживается Web-браузером.

Как любой язык, HTML позволяет вставлять в тело документа пояснительный текст (*комментарии*), который сохраняется при передаче документа по сети, но *не отображается* браузером.

Синтаксис комментария следующий:

<!-- Это комментарий -->

Комментарии могут встречаться в любом месте документа.

Форматирование текста

При выводе на экран текста браузер игнорирует дополнительные пробелы, символы табуляции и возврата каретки, пустые строки. Их можно использовать для того, чтобы сделать текст HTML-документа легко читаемым.

Для правильного отображения Web-страницы в окне браузера следует использовать специальные теги для структурирования и форматирования текста.

Цвета в HTML-документе

Существует **две формы задания цвета**:

символьная - указывается название одного из предопределенных цветов, например, RED;

цифровая - комбинация RGB: #RRGGBB, например,для красного - #FF0000.

Символьные идентификаторы и RGB-комбинации основных цветов приведены ниже, в таблице 1.

Таблица 1. Допустимые цвета в HTML.

Цвет	Символьное обозначение	Цифровое обозначение
Черный	BLACK	#000000
Темно-синий	NAVY	#000080
Голубой	BLUE	#0000FF
Зеленый	GREEN	#008000
Сине-зеленый	TEAL	#008080
Ярко-зеленый	LIME	#00FF00
Ярко-голубой	AQUA	#00FFFF
Темно-красный	MAROON	#800000
Темно-сиреневый	PURPLE	#800080
Оливковый	OLIVE	#808000
Серый	GRAY	#808080
Серебряный	SILVER	#C0C0C0
Красный	RED	#FF0000
Сиреневый	FUCHSIA	#FF00FF
Желтый	YELLOW	#FFFF00
Белый	WHITE	#FFFFFF

Язык HTML определяет для тега **BODY** следующие атрибуты цветов:

BGCOLOR - цвет фона для тела документа,

TEXT - цвет, используемый при выводе на экран текста из данного документа,

LINK - цвет, который будет использоваться при выводе на экран текста из еще не выбранных вами гипертекстовых связей,

VLINK - цвет, который будет использоваться при выводе на экран текста из уже проверенных вами гипертекстовых связей,

ALINK - цвет, которым будут выделяться в тексте гипертекстовые связки в тот момент, когда пользователь щелкает по ним клавишей мыши.

Атрибут **COLOR**, указывающий на цвет, может также использоваться в тегах:

<FONT>, <HR>.

Элементы форматирования на уровне блоков

К тегам блочного уровня относят:

Тег абзаца (параграфа) разделяет два абзаца пустой строкой.

<P ALIGN=LEFT|CENTER|RIGHT|JUSTIFY NOWRAP> текст </P>

Может не иметь пары </P>.

Атрибут **ALIGN** задает выравнивание информации.

Применение атрибута **NOWRAP** дает возможность писать текст без переноса слов.

Для центрирования текста или графики можно использовать также контейнер **<CENTER>...</CENTER>**

Теги заголовков (от 1-го до 6-го уровней) используются для выделения структурных частей текста.

Каждый стиль заголовка имеет свой размер. Тег **<H1>** имеет наибольший размер.

<H1|H2|H3|H4|H5|H6 ALIGN=...> текст </H1|H2|H3|H4|H5|H6>

Тег горизонтальной линейки предназначен для вычерчивания горизонтальной линии**.**

<HR ALIGN=... SIZE=... WIDTH=... NOSHADE>,

где атрибут **SIZE** задает толщину линии в пикселах,

WIDTH —ширину в пикселах или процентах от ширины окна броузера,

NOSHADE позволяет представить линию без тени в виде простой темной полоски.

Тег конца строки вызывает переход не новую строчку без разрыва абзаца.

**
**

Тег <WBR> — определяет место возможного (рекомендуемого) переноса (разрыва) строки.

Контейнер <NOBR> …**</NOBR>** заключает в себе текст, который не должен разбиваться на строки, даже если она выходит за границы экрана. Вместо этого браузер позволит горизонтально прокручивать текст.

Контейнер <PRE>…**</PRE>** используется для отображения на экране символов табуляции, возврата каретки, дополнительных пробелов, сохраняет предварительно выполненное форматирование текста. При отображении такого текста используется моноширинный шрифт. Внутри контейнера могут использоваться другие теги форматирования.

Контейнер <BLOCKQUOTE> предназначен для обозначения в документе **цитаты** из другого источника. Текст, обозначенный этим тегом, отступает от левого края документа на 8 пробелов.

Теги, задающие шрифт

<FONT SIZE=… **COLOR**=... **FACE**=...> текст **</FONT>**

SIZE — устанавливает **размер шрифта**, который будет использоваться текстом, содержащимся в пределах элемента FONT. Можно задать абсолютный размер шрифта, указав целое число от 1 до 7. Для шрифта можно также указывать **относительный размер**, присваивая атрибуту целое число со знаком (например, это может быть SIZE="+1" или SIZE="-2").

COLOR — указывает **цвет**, которым будет выделен данный фрагмент текста. Цвета задаются в виде RGB-значения с шестнадцатеричной нотацией, либо выбирается символьное значение одного из стандартных цветов.

FACE — задает **гарнитуру шрифта**, например FACE=ARIAL.

<TT> текст **</TT>** — телетайпный текст (моноширинный).

<I> текст **</I>** — стиль с наклонным шрифтом (**курсив**).

<B> текст **</B>** — стиль с **жирным** шрифтом.

<U> текст **</ U>** — стиль с **подчеркиванием** текста.

<BIG> текст **</ BIG>**— печать текста шрифтом **увеличенного размера** (большего, чем окружающий текст).

<SMALL> текст **</ SMALL>** — печать текста шрифтом **уменьшенного размера** (меньшего, чем окружающий текст).

<SUB> текст **</ SUB>** — печать текста со сдвигом вниз (**нижний индекс** или подстрочный).

<SUP> текст **</ SUP>**— печать текста со сдвигом вверх (**верхний индекс** или надстрочный).

<STRIKE> текст **</ STRIKE>**или **<S> …</S>** — стиль с перечеркиванием текста.

Специальные теги HTML

Тег <ADDRESS> используется для выделения автора документа и его адреса (например, e-mail).

<ADDRESS> Адрес-автора </ADDRESS>

Некоторые символы являются **управляющими символами** в HTML и добавляются в текст только при помощи ESC-последовательностей:

- левая угловая скобка "<" - <
- правая угловая скобка ">" - >
- амперсанд "&" - &
- двойные кавычки """ - "

Существует большое количество ESC-последовательностей для обозначения специальных символов, например "©" для обозначения знака © и "®" для значка ®. Одной из особенностей является замена символов во 2-ой части символьной таблицы (после 127-ого символа) на escape-последовательности для передачи текстовых файлов с национальными языками по 7-битным каналам. ESC-последовательности **чувствительны к регистру**: НЕЛЬЗЯ использовать < вместо <.

Полезный совет: Более подробно рекомендуется изучить лекции № 2 (*Принципы гипертекстовой разметки и язык HTML*) и № 3 (*Контейнеры тела документа*).

Внимание студентов 2-го курса!

Продолжаем наполнять сайт.

Третья работа по *веб-технологиям основана на материале, изложенном в лекциях Россошинского Д.А. Для данной работы - это четвертая и пятая лекции.*

Основные пункты, которые должны быть реализованы:

- графика и все, что с нею связано(отображение, размещение относительно текста, размер);

- знать все графические форматы;

- реализовать анимацию;

- работать с гиперссылками;

- включить в сайт изображение-карту (map);

- списки и таблицы;

- использовать таблицы в дизайне страниц (сделать таким образом одну из страниц);

- создать в ячейках таблицы графические гипертекстовые переходы.

Еще: *знать ответы на все вопросы, которые приводятся в конце лекций. Это входит в общую оценку работы.*

Напоминаю: *реализованный сайт должен содержать только требования по 4 и 5 лекциях.*

Лабораторная работа №4

Формы в HTML-документе

Цель работы

Изучение возможностей, предоставляемых языком HTML 5, для разработки форм, обеспечивающих взаимодействие между клиентом и сервером.

Порядок выполнения работы

Создать страницу с элементами формы. При этом использовать, по возможности, все описанные в методическом указании и лекциях № 6, 7,8, элементы управления, а именно:

- Теги декларации формы
- Элементы создаваемые тегом input
- Простые и графические кнопки
- Списки выбора
- Текстовые области
- Приемы для выравнивания элементов формы
- Методы передачи данных GET и POST
- Страница, должна содержать форму с современными элементами, определяемыми стандартом на HTML 5 - метки (label)
- Указание элементов, обязательных для заполнения
- Использование элементов выбора даты, выбора значений (слов или фраз, что даст возможность перехода к нужным разделам сайта с применением JavaScript, который будет рассказан позже) или адресов и т.п. из заданного списка

- Ввод электронной почты, ввод логина и пароля, элементы, характерных для организации чата или размещение отзывов на содержание или дополнений по содержанию, определения рейтинга.
- Все надписи на элементах д.б. выполнены кириллицей (на укр., рус. страницах сайта), в т.ч. и на кнопке отправки формы. Надписи на элементах управления латинскими буквами допускаются только на англоязычных страницах, если такие страницы присутствуют на сайте.

Внимание! Из лекций 7 и 8 использовать все, что связано с формами, в частности. Дополнительные атрибуты в тэге input. JavaScript будет изучаться позже, здесь сделать заготовки-окна.

Методические указания

Форма — это инструмент, с помощью которого HTML-документ может послать некоторую информацию в некоторую заранее определенную точку внешнего мира, где информация будет некоторым образом обработана.

Некоторые WWW-браузеры позволяют пользователю, заполнив специальную форму, возвращающую полученное значение, выполнять некоторые действия на вашем WWW-сервере. Когда форма интерпретируется Web-браузером, создается специальные экранные элементы, такие, как поля ввода, checkboxes, radiobuttons, выпадающие меню, скроллируемые списки, кнопки. Когда пользователь заполняет форму и нажимает кнопку "Подтверждение" (SUBMIT - специальный тип кнопки, который задается при описании документа), информация, введена пользователем в форму, посылается HTTP-серверу для обработки и передаче другим программам, работающим под сервером, в соответствии с CGI (Common Gateway Interface) интерфейсом.
Формы передают информацию программам-обработчикам в виде пар [имя переменной]=[значение переменной].
Форма открывается тэгом <FORM> и заканчивается меткой </FORM>.

HTML-документ может содержать в себе несколько форм, однако, формы не должны находиться одна внутри другой.

Тэг <FORM> может содержать три атрибута, один из которых является обязательным:

ACTION: Обязательный атрибут. Определяет, где находитсяобработчик формы.

METHOD: Определяет, каким образом (иначе говоря, с помощью какого метода протокола передачи гипертекстов) данные из формы будут переданы обработчику. Допустимые значения: METHOD=POST и METHOD=GET. Если значение атрибута не установлено, по умолчанию предполагается METHOD=GET.

GET: методом "get" HTTP браузер берёт значение action, добавляет “?” к нему, затем присоединяет набор данных формы, кодированный с использованием типа содержимого "application/x-www-form-urlencoded".

Затем перенаправляет всё по гиперссылке на этот URL. В этом сценарии данные формы ограничены кодами ASCII (нельзя использовать спецсимволы) и имеют весьма жесткие ограничения на объем вводимой информации.

POST: методом "post" HTTP браузер проводит транзакцию HTTP "post" (в теле HTTP-запроса), используя значение атрибута action и сообщение, созданное в соответствии с типом содержимого, определённым атрибутом enctype.

ENCTYPE: Определяет, каким образом данные из формы будут закодированы для передачи обработчику. Если значение атрибута не установлено, по умолчанию предполагается ENCTYPE=application/x-www-form-urlencoded. "Кнопка", чтобы запустить процесс передачи данных из формы на сервер, создается с помощью тэга.

<INPUT TYPE=submit>

Встретив такую строчку внутри формы, браузер нарисует на экране кнопку с надписью Submit, при нажатии на которую все имеющиеся в форме данные будут переданы обработчику, определенному в метке. Надпись на кнопке можно задать любую путем введения атрибута VALUE="[Надпись]".

Пример:

<INPUT TYPE=submit VALUE="Отправить!">

Надпись, нанесенную на кнопку, можно при необходимости передать обработчику путем введения в определение кнопки атрибута NAME=[имя].

Пример:

<INPUT TYPE=submit NAME=button VALUE="Отправить!">

При нажатии на такую кнопку обработчик вместе со всеми остальными данными получит и переменную button со значением Отправить! (т.е. button=Отправить!, это можно видеть в адресной строке).

В форме может быть несколько кнопок типа submit с различными

именами и/или значениями. Обработчик, таким образом, может действовать

по-разному в зависимости от того, какую именно кнопку submit нажал

пользователь.

Существуют и другие типы элементов <INPUT>. Каждый элемент

<INPUT> должен включать атрибут NAME=[имя], определяющий имя

переменной, которая будет передана обработчику. Имя должно задаваться

только латинскими буквами. Большинство элементов <INPUT> должны

включать атрибут VALUE="[значение]", определяющий значение, которое

будет передано обработчику под этим именем.

Основные типы элементов <INPUT>:

TYPE=text

Определяет окно для ввода строки текста. Может содержать дополнительные атрибуты SIZE=[число] (ширина поля для ввода, в символах)и MAXLENGTH=[число] (максимально допустимая длина вводимой строки в

символах).

Пример:

<INPUT TYPE=text SIZE=30 NAME=student VALUE="Вася">

Определяет ширину поля в 30 символов, для ввода текста. По умолчанию в окне находится текст, который пользователь может редактировать. Отредактированный (или неотредактированный) текст передается обработчику в переменной student (student=содержимое_поля). Необходимо попробовать отредактировать поле.

TYPE=password. Определяет окно для ввода пароля. Абсолютно аналогичен типу text, только вместо символов вводимого текста показывает на экране звездочки (*), чтобы посторонний не мог прочесть.

Пример:

<INPUT TYPE=password NAME=pswd SIZE=20 MAXLENGTH=10>

Определяет окно шириной 20 символов для ввода пароля. Максимально

допустимая длина пароля — 10 символов. Введенный пароль передается

обработчику в переменной pswd (pswd=содержимое_поля). Необходимо

попытаться ввести информацию в поле.

TYPE=radio. Определяет переключатель. Может содержать дополнительный атрибут checked (показывает, что кнопка помечена). В группе переключателей с одинаковыми именами может быть только один.

Пример:

<INPUT TYPE=radio NAME=modem VALUE="9600" checked> 9600

бит/с

<INPUT TYPE=radio NAME=modem VALUE="14400"> 14400 бит/с

<INPUT TYPE=radio NAME=modem VALUE="28800"> 28800 бит/с

Определяет группу из трех переключателей, подписанных 9600 бит/с, 14400

бит/с и 28800 бит/с. Первоначально помечена первая из кнопок. Если пользователь не отметит другую кнопку, обработчику будет передана переменная modem со значением 9600 (modem=9600). Если пользователь отметит вторую

кнопку, обработчику будет передана переменная modem со значением 14400 (modem=14400).

TYPE=checkbox

Определяет квадрат, в котором можно сделать пометку. Может содержать дополнительный атрибут checked (показывает, что квадрат помечен). В отличие от переключателей, в группе квадратов с одинаковыми именами может быть несколько помеченных квадратов.

Пример:

<INPUT TYPE=checkbox NAME=comp VALUE="PC">

Персональные компьютеры

<INPUT TYPE=checkbox NAME=comp VALUE="WS" checked>

Рабочие станции

<INPUT TYPE=checkbox NAME=comp VALUE="LAN"> Серверы

локальных сетей

<INPUT TYPE=checkbox NAME=comp VALUE="IS" checked>

Серверы Интернет

Определяет группу из четырех квадратов. Первоначально помечены второй и четвертый квадраты. Если пользователь не произведет изменений, обработчику будут передана одна переменная comp с двумя значениями

(comp=WS и comp=IS).

TYPE=hidden - определяет скрытый элемент данных, который не виден пользователю при заполнении формы и передается обработчику без изменений. Такой элемент иногда полезно иметь в форме, в него можно спрятать от пользователя служебные данные.

Пример:

<INPUT TYPE=hidden NAME=id VALUE="1">

Определяет скрытую переменную индексную id, которая передается обработчику со значением 1.

TYPE=reset - определяет кнопку, при нажатии на которую форма возвращается в исходное состояние (обнуляется). Поскольку при использовании этой кнопки данные обработчику не передаются, кнопка типа reset может и не иметь атрибута name.

Пример:

<INPUT TYPE=reset VALUE="Очистить поля формы">

Определяет кнопку "Очистить поля" формы, при нажатии на которую форма возвращается в исходное состояние.

Элемент <SELECT>:

Меню <SELECT> из n элементов выглядит примерно так:

<SELECT NAME="[имя]">

<OPTION VALUE="[значение 1]">[текст 1]

<OPTION VALUE="[значение 2]">[текст 2]

...

<OPTION VALUE="[значение n]">[текст n]

</SELECT>

Метка <SELECT> содержит обязательный атрибут NAME, определяющий имя переменной.

Метка <SELECT> может также содержать атрибут MULTIPLE, присутствие которого показывает, что из меню можно выбрать несколько элементов. Большинство браузеров показывают меню <SELECT MULTIPLE> в виде окна, в котором находятся элементы меню (высоту окна в строках можно задать атрибутом SIZE=[число]). Для выбора нескольких значений одновременно удерживают кнопку "SHIFT" и выбирают значения мышкой.

Пример:

<SELECT MULTIPLE SIZE=3 NAME="[имя]">

<OPTION VALUE="[значение 1]">[текст 1]

<OPTION VALUE="[значение 2]">[текст 2]

<OPTION VALUE="...">[...]

<OPTION VALUE="[значение n]">[текст n]

</SELECT>

Метка <OPTION> определяет элемент меню. Обязательный атрибут VALUE устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Метка <OPTION> может включать атрибут selected, показывающий, что данный элемент отмечен по умолчанию.

Пример:

<SELECT NAME="selection">

<OPTION VALUE="option1">Вариант 1

<OPTION VALUE="option2" selected>Вариант 2

<OPTION VALUE="option3">Вариант 3

</SELECT>

Такой фрагмент определяет меню из трех элементов: Вариант 1, Вариант 2 и Вариант 3. По умолчанию выбран элемент Вариант 2.

Обработчику будет передана переменная selection (selection=...) значение

которой может быть option1 (по умолчанию), option2 или option3.

Элемент <TEXTAREA>:

Пример:

<TEXTAREA NAME=address ROWS=5 COLS=50>

Поле для ввода большого текста, разбитого на абзацы.

</TEXTAREA>

Все атрибуты обязательны. Атрибут NAME определяет имя, под которым

содержимое окна будет передано обработчику (в примере — address). Атрибут

ROWS устанавливает высоту окна в строках (в примере — 5). Атрибут COLS

устанавливает ширину окна в символах (в примере — 50). Текст, размещенный между метками <TEXTAREA> и </TEXTAREA>, представляет собой содержимое окна по умолчанию. Пользователь может его отредактировать или просто стереть.

Вопросы для контроля

- Для чего предназначен контейнер form?
- Какие элементы создаются в формах применением тега **input?**
- Какие значения может принимать атрибут type в теге input?
- Что означает “submit” как значение атрибута type в теге input?
- Что означает “radio” как значение атрибута type в теге input?
- В каком виде данные передаются на сервер из формы?
- Какие возможности для создания элементов формы дает контейнер button?
- Какм создать графическую кнопку?
- Дайте определение списка выбора с 2-мя элементами со значениями – “Error” и “Warning”!
- Что обеспечивает применение атрибута multiple в списках?
- В чем отличие текстовых областей от текстовых полей?
- Какие 2 способа применяются для выравнивания элементов форм?
- В каком виде передаются данные на сервер при использовании **метода GET**?
- В каком виде передаются данные на сервер при использовании **метода POST**?
- Что определяет атрибут type со значением "search" или "date" в теге INPUT?
- Для чего используется тег **label**?

Лабораторная работа №5

Оформление страницы средствами HTML5 и CSS3

Цель работы

Изучение возможностей, предоставляемых языком HTML 5, для разработки структуры страницы, применения векторной графики и средств для определения стиля с помощью CSS3. Проверка соответствия документа стандартам HTML 4 и HTML 5.

Порядок выполнения работы

Создать страницы с применением элементов HEADER, FOOTER, ARTICLE, SECTION для верстки с полной или частичной заменой table и div. Создать логотип в каком-либо месте главной страницы или задать цвет фона страницы в форме логотипа (полупрозрачного) или надпись (заголовок) в формате SVG + эксперименты с градиентами (показать варианты с линейными и радиальными градиентами применительно к рисункам или тексту и выбрать некоторый окончательно). Не требуется создавать сложный логотип, например это может аббревиатура названия сайта, - одна или несколько букв или несложный значок. Выполнить верификацию документа, по крайней мере одной странички, на соответствие стандарту HTML 4 и HTML 5. Подобрать нужный валидатор и показать результаты в виде скриншотов и пояснений к ним. В работе не требуется строгое следование какому-то одному стандарту, но необходимо прокомментировать результаты верификации.

Работа базируется на материалах лекций 7, 8, 12. Некоторая информация по валидаторам содержится в лекции 3. В качестве справочников можно использовать следующие ссылки:

http://prosetech.com/html5/
http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html#simple_objects
https://developer.mozilla.org/en-US/docs/Web/SVG/Element/

ВНИМАНИЕ!

*Зная Ваши неограниченные способности в использовании **copy-paste** из интернета, убедительная просьба: писать комментарии в тексте страницы, где Вы выполняете требования к данной работе.*

Вопросы для контроля

- Для каких типов графики используются форматы SVG и CANVAS?
- Каким образом выполняется размещение SVG-графики?
- Из каких элементов строится код SVG?
- Что из себя представляет элемент PATH?
- Для чего и где используются команды M и L?
- Что применяют чтобы раскрасить рисунки, созданные в формате SVG?
- Для чего используются контейнеры **linegradient** и **radialgradient** и где они размещаются?
- Какие 2 составляющие требуются для создания визуального элемента при помощи формата CANVAS?
- Какие четыре элемента применют для описания фигур используя CANVAS ?
- Какие методы скрипта используются для рисования линий и рисования дуг?
- Что может обеспечить сервис **Goog1e Docs?**

Вопросы для самостоятельной проработки

- Для чего в SVG-изображении может быть применен элемент textPath?
- Для чего в SVG-графике используется элемент polygon?

Лабораторная работа №6

Программирование скриптов на стороне клиента средствами ЯП JavaScript

Цель работы

Освоение элементов языка JavaScript для программирования скриптов Веб-приложения на стороне браузера. Размещение проекта на сервере.

Порядок выполнения работы

Работа разделена на две части:

1. Размещение проекта на сервере
2. Разработка страницы администратора сайта (с использованием скриптов на JavaScript).

Размещение проекта на сервере

Требуется установить локальный сервер Денвер (Apache) с официального сайта разработчиков. Разместить на сервере, выполненный ранее в рамках лабораторных работ 1-5, проект. Ссылка на сайт разработчиков Денвера и инструкции по размещению сайта приведены в Приложении. Примечание: можно не тестировать SSL-соединение сервера, он и так обеспечит просмотр страниц разработанного сайта.

Разработка страницы администратора сайта

Должна быть разработана страничка администратора сайта (в дальнейшем САС) для обеспечения взаимодействия с информационной частью Веб-приложения. В данной лабораторной работе разрабатывается отладочный автономный вариант странички с использованием массивов и объектов JavaScript. На САС необходимо разместить:

1. Формы для регистрации и авторизации пользователя. В этом случае пользователю можно будет размещать собственный комментарий к содержанию сайта, его дополнять, задавать вопросы и получать ответы (необходимо предусмотреть для этого соответствующую форму).
2. Таблицу, секцию (раздел) или текстовое поле (на усмотрение разработчиков) для отображения истории того, что вводили зарегистрированные пользователи – комментарии, дополнения и пр. Предусмотреть навигацию по данным, которая обеспечивала бы "листание" вперед, назад или сразу на несколько пунктов вперед или назад и т.п. Отображение данных должно

обеспечиваться в том порядке, как они вводились (упорядочение по дате ввода и времени).

3. Формы для авторизации администратора и обеспечения возможности просмотра данных (комментариев и пр.), введенных пользователями сайта и данных о пользователях - их логинов, паролей, адресов эл.почты и пр. данных, введенных при регистрации. Необходимо обеспечить администратора сайта элементами управления для возможности вычеркивания соответствующей записи, введенной некоторым пользователем и занесения его в список пользователей, которым запрещено в дальнейшем регистрироваться и размещать свои комментарии на сайте. Чтобы затруднить возможность повторной регистрации взять за основу - сравнение введенного адреса эл.почты, для этого соответствующее поле сделать обязательным для заполнения (все другие варианты, используемые в настоящее время, не дают 100% вероятности идентификации компьютера в сети).

В следующей лабораторной работе формы, указанные в пп.1, 2 должны быть размещены на страницах пользователе сайта, а формы пп.3 остаться на САС. Для поддержки информационной части САС можно применить распределение данных по следующим массивам (в дальнейшем для этого будут использоваться файлы и возможности PHP):

1. Массив для хранения записей о регистрации пользователей (логины, пароли, эл.адрес, данные о пользователе),
2. Массив блокированных пользователей (те-же данные, что и в предыдущем случае),
3. Упорядоченный по дате и времени введения информации, массив с данными (текстом) разрешенными для отображения (в массиве помимо текста и дат хранятся логины пользователей),
4. То-же, что и предыдущее, но в массиве хранятся данные, которые были введены, но еще должны быть просмотрены администратором сайта и затем включены в упорядоченный массив 3, или записи должны быть удалены, а пользователи, их сформировавшие, блокированы для дальнейшего ввода своих комментариев.

Все данные в массивах должны быть представлены объектами JavaScript.

Работа базируется на материалах лекций 14, 15, 16 и предыдущих темах, связанных с размещением форм на HTML-страницах.

Во **Вопросы для самоконтроля**

- Что из себя представляет язык ECMAScript?
- Перечислите четыре способа функционального применения JavaScript!
- В чем суть схемы включения скрипта - URL?

- Как скрипт включается в документ для обработки события?
- Для чего и как может использоваться контейнер <SCRIPT>?
- Для чего в теге SCRIPT используется атрибут **async**?
- Для чего в теге SCRIPT используется атрибут **src**?
- Как в скриптах выглядит создание переменных?
- Могут ли переменные создаваться без var-оператора?
- На что ссылается и для каких целей может использоваться оператор **this**?
- Назначение оператора **with**?
- Как выполняются операторы === и !==?
- В чем отличие выполнения операторов && и &?
- Как выглядит и для чего применяется оператор "Итерации по свойствам объекта"?
- Как записываются строки-литералы в JavaScript?
- Как создаются в скриптах строки-объекты?
- Могут ли к обычным строкам-литералам применяться методы строк-объектов?
- На какие 2 категории разделяются методы работы со строками?
- Назовите по меньшей мере 5 свойств-конструкторов глобального объекта!
- Что из себя представляет свойство **Math** глобального объекта?
- Назовите варианты создания объектов полбьзователя!
- Приведите пример объекта, создаваемого конструктором **Object** и добавления ему свойства-метода (функции)!
- Приведите пример создания объекта используя литеральную нотацию!
- Приведите пример создания объекта используя конструкцию **function** и добавления объекту некоторого набора свойств!
- Как в языке JavaScript указывается объект=прототип?
- Перечислите варианты создания массивов в скриптах на JavaScript!
- Разработайте конструктор массива, в котором используются 3 параметра, которые определяют - число элементов, строка, определяющая тип элементов (строковые, булевские, числа, даты, объекты), значение, используемое для формирования элементов; конструктор должен проверять соответствие указанного типа типу 3-го параметра и если соответствия нет - вернуть значение **null**!
- Какие методы объекта **document** используются для динамического формирования контента?
- Дайте определение DOM!
- Как выполнить в JavaScript контроль уровня реализации?
- Что включает уровень спецификаций 2 DOM?
- Что находится в вершине иерархии объектной модели документов?
- В каком интерфейсе перечислены атрибуты и методы nodeType, lastChild, replaceChild()?
- Что позволяют выполнить методы - **getElementsByName, getElementById, getElementsByTagName?**
- Что обеспечивают следующие свойства - - **images, links, forms, anchors**, содержащиеся в объекте **document**?
- Какой самый простой способ изменения содержимого документа?
- Как можно изменить значение атрибутов в тегах документа?
- Эквивалентны ли следующие выражения?
- Что необходимо сделать для добавления нового элемента в документ?
- Что необходимо сделать для удаления элемента из документа?
- Напишите пример переопределения обработчика некоторого события для элемента button!
- Какой синтаксис используется в JavaScript для изменения стиля элемента с использованием DOM?

Вопросы для самостоятельной проработки

- Самостоятельно рассмотреть примеры использования операторов **try/catch/throw, а также continue** и **break** с меткой или без нее!

Вопросы для самостоятельной проработки

- Уточните 3-й и 4-й параметры метода open объекта window!

Вопросы для самостоятельной проработки

- Что из себя представляет, для чего используется и как задается понятие "пространство имен"?
- Что означает запись document.implementation.hasFeature?

Приложение

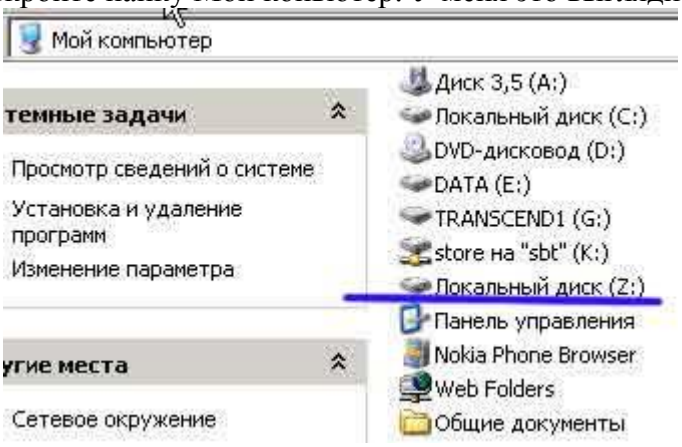
Основы работы с Denver (Apache+PHP)

Денвер построен на Apache плюс широкий набор возможностей среди них поддержка , PHP, MySQL , Perl , Эмулятор sendmail и SMTP-сервера , (для эмуляции почтового сервера) , система управления MySQL - phpMyAdmin и многое другое:http://www.denwer.ru/base.html

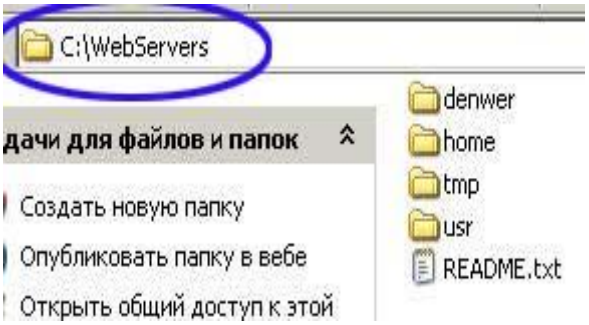
"Прописываем" новый сайт на Локальном сервере

http://www.luksweb.ru/view_post.php?id=195

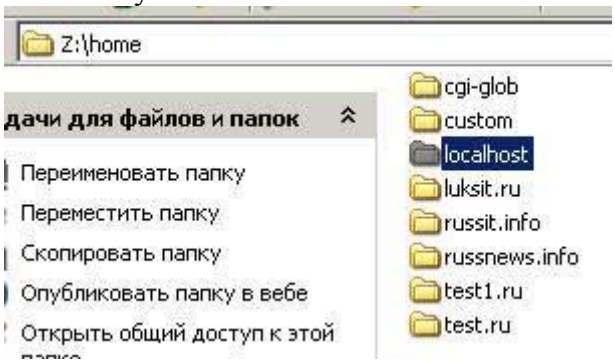
И так, после установки Локального сервера (пакета Денвер) на домашний компьютер у Вас в системе появляется дополнительный виртуальный диск, обычно это диск Z, но у Вас этот диск может иметь другую букву, но это не важно, принцип работы с Денвером от этого не меняется. Откройте папку Мой копьютер. У меня это выглядит вот так:



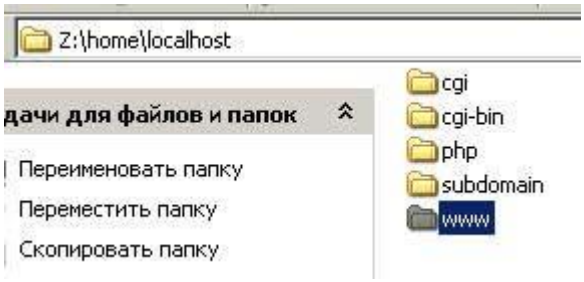
Физически диск Z, расположен на системном диске C, здесь появляется папка WebServers (рисунок ниже):



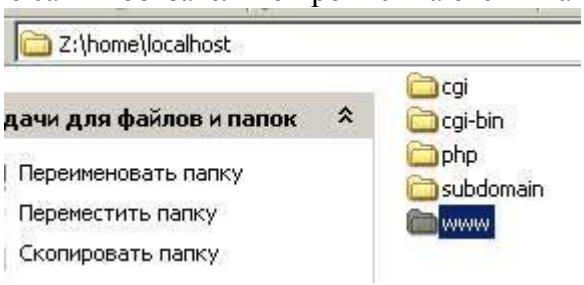
Как Вы будете заходить в рабочую папку своего сервера (через диск Z или через C/WebServers) абсолютно все равно, как Вам удобнее, так и работайте. Итак, заходим в папку WebServers и открываем папку home:



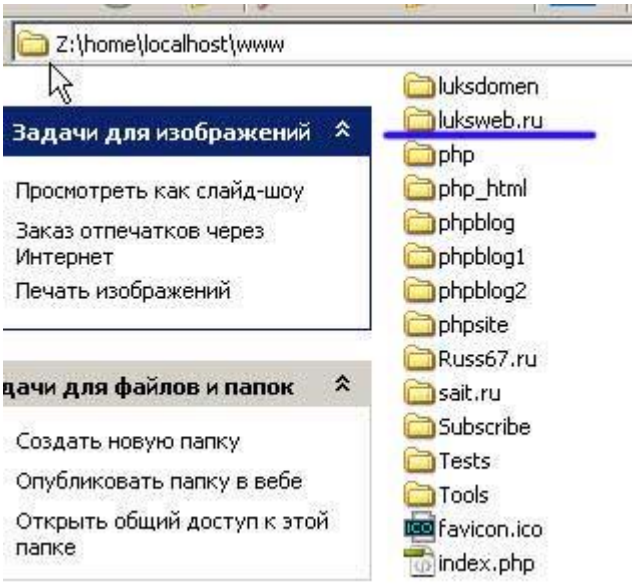
Прописать новый, сайт Вы можете и здесь (в папке home), например здесь, как Вы видите у меня прописаны сайты luksit.ru, russit.info, russnews.info, но все таки удобнее это сделать будет в папке localhost. Почему объясню чуть позже. Открываем папку localhost и видим:



Внимание! Все новые сайты обязательно прописываются в папке www



Открываем папку www и создаем новую, отдельную папку для сайта, которые хотите прописать в Денвере. Например для моего сайта **http://www.LuksWeb.ru** , я создал папку luksweb.ru (рисунок ниже):



Имя для этой папки, Вы можете придумать любое, но логичнее будет назвать эту папку по имени сайта, у меня это luksweb.ru (можно и без .ru). **Это и будет корневая папка сайта на Локальном сервере.** Сюда Вы будете скаладывать все папки и файлы Вашего сайта.

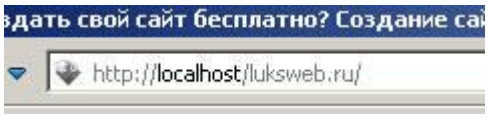
Как Вы помните, самый первый файл, который открывается по умолчанию на всех серверах (как Локальных, так и Реальных в Интернете) - это файл index.html или index.php. В зависимости по какой технологии сделан Ваш сайт. Естестенно этот файл должен лежать в корневой папке.

Когда все это сделано, необходимо запустить Локальный сервер или Перезапустить, если до этого Вы уже запустили сервер. Делается это для того, чтобы новый сайт (папка которую Вы только что создали) прописался на сервере.

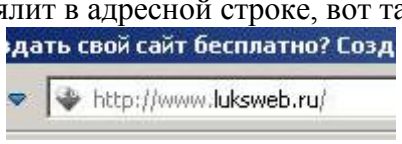
Находите на рабочем столе вот такие ярлыки (созданные при установке локального сервера):

и нажимаете ярлык Start Denwer для запуска сервера или Restart Denwer для перезапуска сервера. Запуск/перезапуск Локального сервера занимает 10 - 15 секунд.

После этого открываете Ваш любимый браузер и в адресной строке прописываете адрес сайта, с которым хотите работать, например вот так:



У Вас соотвественно будет прописан свой сайт. А теперь обратите внимание, что реальный адрес в интернете, выглялит в адресной строке, вот так:



Заметили разницу? Правильно! Сайт прописанный на Локальном сервере имеет адрес такого вида: http://localhost/luksweb.ru/ и лежит на Вашем компьютере.

Сайт имеющий адрес вида: http://www.luksweb.ru/ находится на реальном хостинге в Интернете.

Очень удобно. Иногда возникает необходимость работать одновременно с двумя серверами и видеть свой сайт и на Локальном сервере и в Интернете.

Именно поэтому я рекомендую Вам прописывать на локальном сервере (Денвере) все сайты в папке localhost. Тогда сразу наглядно видно, где открыт сайт: В Интернете или на Вашем домашнем компьютере.

Вот в общем то и все, мы с Вами прописали новый сайт в Денвере или как говорят на Локальном сервере. Таким же образом прописываются все остальные сайты: создаете новую папку, ложите в эту папку новые файлы сайта и перезагружаете Локальный сервер.

Еще совет, после любого внесения изменений в файлы Ваших сайтов, не забывайте сохранять эти файлы (Ctrl + S) и обновлять в окне браузера F5.

После окончания работы с Локальным сервером, обязательно остановите Локальный сервер (нажать ярлык Stop Denwer).

Лабораторная работа №7

Серверная часть Веб-приложения. Основы программирования в PHP.

Цель работы

Овладение знаниями и навыками программирования серверной части веб-приложения с применением языка программирования PHP.

Порядок выполнения работы

В работе необходимо выполнить:

1. Разместить все необходимые элементы управления - регистрации, авторизации, ввода "комментария" и отображения истории "комментариев" на основных страницах сайта, доступных пользователям сайта (в предыдущей работе эти элементы управления для целей отладки размещались на странице администратора).
2. Удалить все перечисленные выше элементы управления со страницы администратора, оставив на ней только те, которые необходимы для - ввода пароля администратора, просмотра сведений о пользователях (с возможной блокировкой некоторых из них по результатам анализа их комментариев), элементы для просмотра новых введенных комментариев и просмотра всех доступных для просмотра на страницах сайта комментариев.
3. Заменить ранее разработанный локальный отладочный вариант работы с данными на основе массивов JavaScript на вариант работы с файлами с применением языка программирования PHP.

Требования к организации работы с файлами

В работе необходимо обеспечить следующие требования:

1. Отображаемые для просмотра на страницах сайта комментарии должны быть упорядочены по времени и дате ввода. В соответствующем элементе управления должны отображаться последовательности записей, которые включают - "дата и время ввода комментария, логин пользователя и сам комментарий". Отдельные записи необходимо разделять, например строкой символов почеркивания или каким-либо другим способом.
2. Доступ к файлам со стороны пользователей сайта и администратора сайта необходимо синхронизировать таким образом, чтобы избежать возможных сбоев или эффекта "клинча" (см. Приложение).

Требования к демонстрации результатов разработки

Необходимо представить работу с сайтом в рамках единовременного сеанса как со стороны администратора сайта так и со стороны пользователя. Так-же необходимо вести и отладку, чтобы выявить возможные "локи" по доступу к файлам.

Теоретический материал

Работа базируется на материалах лекций 17-20 и предыдущих темах, связанных с размещением форм на HTML-страницах и использованием DOM. Достаточно полно освещена работа с PHP и в частности с файлами на Интернет-ресурсах http://www.php.su/functions/?cat=filesystem и http://www.php.net/manual/ru/index.php.

Приложение

Синхронизация доступа к файлу (функция flock)

flock

(PHP 3 >= 3.0.7, PHP 4, PHP 5)
flock -- Портируемое рекомендательное запирание файлов

Функцию необходимо применить в реализации курсовых проектов или лабораторных, когда к файлу возможен одновременный доступ многих пользователей! В этом случае будут надежно обрабатываться как операции чтения из файла, так и записи в него, а также соблюдаться необходимый порядок размещения записей.

Описание

bool **flock** (resource handle, int operation [, int &wouldblock])

PHP поддерживает портируемый механизм запирания файлов целиком, который имеет рекомендательный характер (это означает, что все обращающиеся к файлу программы должны использовать такой же способ запирания файла, иначе запирание не сработает).

Замечание: flock() является обязательным под Windows.

flock() применяется к *handle*, который должен быть указателем на открытый файл. Параметр *operation* может принимать следующие значения:

- Чтобы установить общее запирание (чтение), установите *operation* в значение **LOCK_SH** (или 1, в случае версии PHP ниже 4.0.1). В этом случае запись в файл из других сценариев блокируется, но возможно чтение из файла.
- Чтобы установить эксклюзивное запирание (запись), установите *operation* в значение **LOCK_EX** (или 2, в случае версии PHP ниже 4.0.1), в этом случае все остальные сценарии (кроме вашего исполняемого) не смогут получить доступ к файлу, пока не будет снято запирание или не завершится ваш сценарий.

- Чтобы отпереть файл (после общего или эксклюзивного запирания), установите *operation* в значение **LOCK_UN** (или 3, в случае версии PHP ниже 4.0.1).
- Если вы не хотите, чтобы **flock()** блокировал сценарий при запирании файла, добавьте **LOCK_NB** (или 4, при использовании версии PHP ниже 4.0.1) к параметру *operation*. В этом случае сценарий не блокируется (пока файл заперт другим сценарием), если файл заперт другим сценарием и функция вернет **false.**

Если третий параметр установлен в **true**, то нужно не только установить запирание на файл, но и блокировать выполнение сценария, пока файл не станет доступным.

flock() позволяет вам реализовывать простую модель чтения/записи, которая может быть использована практически на любой платформе (включая большинство проивзодных от Unix платформ, и даже Windows). Необязательный третий аргумент устанавливается в **TRUE**, если запирание также блокирует (код ошибки EWOULDBLOCK). Блокировка снимается при помощи этой же функции или функции **fclose() (**которая также автоматически вызывается при завершении выполнения скрипта).

Возвращает **TRUE** в случае успешного завершения или **FALSE** в случае возникновения ошибки.

Пример 1. Пример использования функции flock()

```
<?php
$fp = fopen("/tmp/lock.txt", "w+");
if (flock($fp, LOCK_EX+LOCK_NB)) { // выполнить эксклюзивное запирание
   fwrite($fp, "Что-нибудь пишем\n");
   flock($fp, LOCK_UN); // отпираем файл
} else {
   echo "Не могу запереть файл !";
}
fclose($fp);
?>
```

Замечание: Из-за того, что функции **flock()** необходим указатель на файл, - может понадобиться воспользоваться специальным запирающим файлом для того, чтобы ограничить доступ к файлу, который вы намерены очищать путём открытия его в режиме записи (используя "w" или "w+" в качестве аргумента функции **fopen()**).

Внимание - flock() не будет работать на NFS и многих других сетевых файловых системах. Обратитесь к документации вашей операционной системы для получения дополнительной информации.

В некоторых операционных системах **flock()** реализован на уровне процессов. При использовании многопоточных серверных API, таких как ISAPI, вы не можете полагаться на **flock()** для защиты ваших файлов от других PHP-скриптов, которые работают в параллельном потоке на том же сервере!

flock() не поддерживается на старых файловых системах вроде *FAT* и его производных, так что всегда будет возвращать **FALSE** в этом окружении (это особенно касается пользователей Windows 98).

Лабораторная работа №8

Библиотека jQuery и Ajax-технология в разработке Веб-приложения

Цель работы

Необходимо изучить приемы использования библиотеки jQuery и технологию Ajax с применением jQuery

Порядок выполнения работы

В работе необходимо выполнить:

1. Скачать библиотеку jQuery в директорий своего сайта на Денвере
2. Ознакомиться с функциями библиотеки в категориях управления объектами, обработки событий, добавления эффектов и поддержки Ajax
3. Выполнить определение некоторых обработчиков событий (1-2 обработчика), назначенных статически (атрибутами on…) динамически с помощью jQuery
4. Добавить некоторые эффекты (анимацию или что-либо подобное) с помощью функций jQuery (1-2 эффекта) или вместо этого пункта обеспечить контроль некоторых элементов форм с помощью функций jQuery (1-2 элемента любой из используемых на сайте форм)
5. Реализовать обмен данными между формами и сервером (по желанию студента - используются все формы или только некоторые из них)

Основные требования к программной реализации

В виду ограниченного времени, оставшегося до завершения семестра, лаб.работа №8 оценивается следующим образом - выполнение всех пунктов 1-4 оценивается исходя из обычной оценки, установленной для лаб.работ. Выполнение пп.5 - оценивается дополнительными баллами - 3-4 балла.

Требования к демонстрации результатов разработки

Особые требования не выдвигаются. Сайт размещается на Денвере. Демонстрируется достигнутая функциональность и дается подробное объяснение преподавателю по поводу использованных функций jQuery (это касается результатов выполнения пп.1-4). По поводу пп.5 дается объяснение за счет чего "реализуется Ajax" как со стороны клиента, так и со стороны сервера.

Теоретический материал

Работа базируется на материалах лекций 21-24, самостоятельного изучения материалов из Интернет, в т.ч. сведений по работе с XMLHttpRequest, xajax и/или Smarty.

Приложение 1. ТЕМЫ САЙТОВ для разработки студентами.

1. Архитектура древней Греции. (Акрополь, Парфенон, Большой театр Москва, Ватикан Италия,элементы архитектуры, фотографии,)
2. Русские художники, входившие в объединение передвижников. (Время возникновения объединения, фамилии художников, выставки, картины).
3. Французские художники-импрессионисты. (Время возникновения объединения, фамилии художников, выставки, картины).
4. «Серебряный век» русской поэзии. (1880-ые – 1917 годы. Символисты: А.Белый, к. Бальмонт, В.Брюсов, ранний А. Блок. Акмеисты: Н.Гумилев, ранняя А.Ахматова, О.Мандельштам. Футуристы: В.Маяковский, В.Хлебников, ранний Б.Пастернак. Характеристики творчества. Примеры. Фото).
5. История исследования космоса и развитие космонавтики. (Отец ракетостроения Циолковский К.Э., принцип работы ракетного двигателя, покорители космоса, Байконур, первый шаг на Луне, рекорды космонавтики).
6. Басни Эзопа. (Жанр литературы – басня. Биография родоначальника басни – Эзопа. Примеры. Продолжатель – баснописец Крылов).
7. Рождество. (История рождества, персонажи, празднование в разных странах, рождество в искусстве : фильмы, картины, балет Чайковского П.И. «Щелкунчик», опера Римского-Корсакова Н.А. «Снегурочка»).
8. Великие информатики. (Касперский Е.В., Дейкстра Эдсгер Вибе, Лебедев С.А., Линус Бенедикт Торвальдс : биографии, достижения, фото).
9. Внутреннее устройство ПК. (Имеется в виду архитектура системного блока. Охарактеризовать все узлы).
10. Виртуальная экскурсия по Байкалу. (Географическое положение, характеристики, жители Байкала, фотографии в разные времена года).
11. Австралия.
12. Гражданская война в США. (Время возникновения войны, причины, важнейшие сражения, руководители, результат).
13. Город моего детства.
14. Известные имена. Николай Иванович Пирогов.
15. Известные имена. Михайло Ломоносов.
16. Архитектура России 18-19 веков. (Известные памятники: Зимний дворец, Петергоф, Эрмитаж. Архитектор Варфоломей Растрелли. Фото.)

17. Живопись. (Основные техники: темпера,акварель, пастель, гуашь, масло, графика, мозаика, фреска. Виды живописи по назначению: монументальная, станковая и т.д. Примеры).
18. Автомобили 20-х-30-х годов. Ретро. http://avtovstileretro.ru/1920-1930g/
19. Заповедники средней полосы Украины. (Черноморский биосферный заповедник. Природный заповедник «Горганы» .Заповедник «Каменные могилы» Донецкой области. Каневский заповедник.)
20. Музыкальные инструменты средневековья. (Духовые, ударные, смычковые струнные, струнные щипковые: описание, фото.)
21. Культура каллиграфии Китая и Японии. (Стили письма. Связь каллиграфии с живописью. Инструменты. Фото.)
22. Развитие экстремальных видов спорта (дайвинг, парашютизм, альпинизм, стрит-рейсинг, дельтапланеризм, паркур.)
23. Международный музыкальный фестиваль Koktebel Jazz Festival (Джаз Коктебель). (История, основатели-идеологи, участники. Фото.).
24. Всеукраинский фестиваль современной песни и популярной музыки «Червона Рута». (История, основатели, исполнители. Фото.)
25. Рок-фестивали в Украине.
26. Музеи и галереи Киева. (Национальный заповедник "София Киевская". Этнографический музей «Пирогово». Национальный художественный музей Украины. Музей русского искусства. Национальный музей истории Украины. Музей Великой Отечественной войны. Музей Одной улицы. Музей Михаила Булгакова.)

27. Стихотворный размер (метр): ямб, хорей, анапест, дактиль, верлибр. (Поэты и примеры творчества.)

28. Творчество литовского художника и композитора Микалоюса Чюрлениса. (Биография, музыка, картины, фото.)
29. Жанры литературы Японии. (История возникновения, жанры, поэты).
30. Военная авиация. (Истребители, бомбардировщики, разведчики, штурмовики, транспортные, морские.)

Приложение 2 . Титульный лист

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут”

Кафедра прикладної математики

Лабораторна робота №__

з дисципліни «**Програмування та підтримка Web-застосувань»**

Виконав:

Студент групи _____ _____ _____

(шифр групи) (прізвище, ім’я, по батькові) (підпис)

Перевірив:

Викладач _____Бали _____ _____

(вчені ступінь та звання, прізвище, ініціали) (підпис)

Приложение 3. Пример структуры сайта.

Простейший пример структуры сайта по теме: поэт Ф.И.Тютчев

Файл poet.htm
Заглавие *документа* *(гиперссылка на Заглавие Текста2)* ()
Таблица: Портрет____ Ф.И.Тютчева Текст1 (файл photo1.jpg) (файл article) *(гиперссылка на п-т Мерзлякова*
Заглавие 2-го уровня *(гиперссылка на Текст3)* Текст2 (файл bio)
Горизонтальная черта Текст4 (файл groza)

Файл photo2.gif
Портрет А.Ф.Мерзлякова (файл photo2)

Файл about.htm
Текст3 *(гиперссылка на Текст4)*