Ермоленко А.В.

Web-программирование

Лекции и лабораторные работы

Направление «Математика и компьютерные науки»

Всего часов – 108 ч:

Самостоятельная работа – 53 ч;

Лекции – 16 ч;

Лаб. занятия – 34 ч

Контрольная работа – 1

Итоговый контроль: зачет

Оглавление

Введение

Лекции и лабораторные работы разработаны для студентов направления «Математика и компьютерные науки» на основе многолетнего преподавания автором дисциплины «Web-программирование» в Сыктывкарском государственном университете. В рамках курса рассматриваются только технологии, работающие на стороне клиента, а именно – HTML, CSS, JavaScript. Серверная часть в рамках курса не рассматривается.

Курс рассчитан на 108 часов, из них

Самостоятельная работа – 53 ч;

Лекции – 16 ч;

Лабораторные занятия – 34 ч

В рамках курса предусмотрена 1 контрольная работа.

Итоговый форма контроля – зачет.

Продолжение курса видится автором в виде дальнейшего изучения баз данных и программирования на php, а также в рамках курсового и дипломного проектирования.

Содержание курса направлено на понимание сути web-технологий. Поэтому приведены только основные конструкции и примеры, призванные показать принципы взаимодействия. Для более детального изучения рекомендуется изучить материалы, представленные, например, на сайтах:

- http://htmlbook.ru;
- http://javascript.ru;
- https://htmlweb.ru.

Тем не менее, пособие призвано создать полноценную картину о web-программировании, поэтому содержит примеры и необходимые материалы.

Для удобства приведены вопросы к зачету по дисциплине «Web-программирование».

Вопросы к зачету

1. Определение языков разметки. HTML, версии.
2. Структура Web-страницы (обычная, с фреймовой структурой).
3. HTML. Форматирование текста, изменение шрифта, заголовки, списки.
4. HTML. Вставка рисунков и таблиц.
5. Верстка страниц при помощи таблиц.
6. HTML. Гиперссылки, примеры.
7. HTML. Карты изображений.
8. HTML. Фреймы. Пример.
9. HTML. Формы. Способы передачи данных на сервер. Элементы формы.
10. Определение, назначение, версии каскадных таблиц стилей (CSS).
11. Синтаксис CSS.
12. Верстка страниц при помощи CSS.
13. Статические и динамические фильтры.
14. Управление положением на странице (свойства left, top, z-index, position, visibility, overflow).
15. JavaScript, назначение, размещение, основные операторы.
16. Классы языка JavaScript.
17. Класс Data. Пример использования.
18. Класс String. Пример использования.
19. Работа с математическими формулами в JavaScript.
20. Обращение к элементам формы из JavaScript.
21. Обработка событий при помощи JavaScript.
22. Объектная модель DHTML.
23. Объект window.
24. Объект document.
25. Объекты history, location, screen, navigator.
26. Объект event. Обработка событий.
27. Функции и свойства смены содержимого.

28. Модель DOM. Уровни. Структура документа.
29. DOM. Навигация по дереву документов. Создание узлов.
30. DOM. Редактирование дерева элементов.
31. DOM. Работа с массивами элементов. Пример.
32. Библиотека jQuery. Обращение к элементам.
33. Создание элементов DOM с помощью jQuery. Пример реализации раскрывающегося списка.

Для самостоятельного изучения

34. Системы управление контентом (CMS). Назначение, функции.
35. Классификация CMS.
36. Схема функционирования CMS. Проблемы установки и использования CMS.
37. Установка CMS WordPress.
38. Настройка WordPress. Плагины, шаблоны.
39. Способы программирования на php в WordPress.

1. HTML

Лекция 1. Основы HTML

1.1. История языка HTML. Создание web-страниц

HTML (*HyperText Markup Language* — «язык гипертекстовой разметки») — стандартный язык разметки документов во Всемирной паутине.

Версии языка HTML

- HTML 0.9
- HTML 2.0, одобренный как стандарт 22 сентября 1995 года;
- HTML 3.2— 14 января 1997 года;
- HTML 4.0 — 18 декабря 1997 года;
- HTML 4.01 — 24 декабря 1999 года;
- HTML 5— 28 октября 2014 года
- HTML 5.1 начал разрабатываться примерно 19 декабря 2012 года.

Для создания web-страниц нам понадобиться обычный текстовый редактор, например, **Блокнот** или **NotePad++**. Отличие в том, что создаваемым файлам дается расширение *html*.

Рассмотрим создание страницы, выводящей фразу «Сыктывкарский государственный университет». Для этого в блокноте набираем:

```
<HTML>
<HEAD>
<TITLE>СыктГУ</TITLE>
</HEAD>
<BODY>
Сыктывкарский государственный университет
</BODY>
</HTML>
```

Затем необходимо сохранить этот документ в виде файла с расширением html. Далее выполняем двойной щелчок по файлу, после чего запускается браузер для просмотра результата.

1.2. Структура html-документа

HTML — элемент верхнего уровня — ограничивает начало и конец документа. Элемент HTML является элементом верхнего уровня в языке HTML. Использование элемента HTML не сказывается на выводе документа. Если элемент HTML используется, начальный и конечный теги HTML должны идти вокруг *всего* документа, сразу после объявления DOCTYPE.

<HTML>

заголовок и тело документа

</HTML>

HEAD (ЗАГОЛОВОК) — заголовок документа

Основная структура HTML-документа должна состоять из заглавия и тела. Явное вложение заголовка в элемент HEAD не обязательно. Использование элемента HEAD не сказывается на вывод документа. Атрибуты отсутствуют. В документе должен присутствовать только один элемент HEAD, и он должен появиться до элемента BODY .

Синтаксис использования

<HEAD>

элемент TITLE

</HEAD>

Заметим, что начальный и конечный теги могут отсутствовать.

Тег <HEAD> должен содержать один TITLE элемент и возможно следующие элементы

- ISINDEX:
- META;
- LINK;
- STYLE;
- SCRIPT.

BODY (ТЕЛО) — тело документа. Основная структура документа HTML всегда состоит из заголовка и тела. Нет необходимости явно помещать тело в элемент BODY, однако делая так, можно специфицировать атри-

буты, влияющие на представление документа в целом (например, установить фоновое изображение или цвет).

Если элемент BODY не содержит атрибуты, использование его не дает явного эффекта в непосредственном отображении документа. Элемент HTML может быть задан яво или подразумеваться. Только один элемент BODY разрешен в документе, и он должен находится после элементаHEAD (который тоже может быть как подразумеваемым, так и явным).

Основной синтаксис

<BODY>*тело документа*</BODY>

Возможные атрибуты тега BODY

Имя атрибута	**Возможные значения**	**Смысл**
BGCOLOR	*Цвет*	Фоновый цвет документа
TEXT	*Цвет*	Цвет для текста документа
LINK	*Цвет*	Цвет для *непосещенной* гипертекстовой связи
VLINK	*Цвет*	Цвет для *посещенной* гипертекстовой связи
ALINK	*Цвет*	Цвет для *активной* гипертекстовой связи; используется для выделения текста , когда пользователь нажал на связь
BACKGROUND	*URL*	URL фонового образа

1.3. Теги управления форматированием

Абзац формируется при помощи тега … . Изначально предполагалось использовать вместе с тегами параметры[1]. Параметры тега – способы выравнивания (left, right, center, justify). Например, если необходимо сделать абзац с выравниванием по ширине, то надо написать:

Содержание абзаца

Заголовки предназначены, во-первых, показать важность раздела, к которому относятся, а во-вторых, с помощью различных заголовков легко регулировать размер текста. Чем выше уровень заголовка, тем больше размер шрифта. Самым верхним уровнем является уровень 1 (<h1>), а самым нижним — уровень 6 (<h6>).

Также следует отметить, что использование заголовков приветствуется поисковыми системами.

Пример использования заголовков.

```
<body>
 <h1>Заголовок первого уровня</h1>
 <h2>Заголовок второго уровня</h2>
 <h3>Заголовок третьего уровня</h3>
 <h4>Заголовок четвертого уровня</h4>
 <h5>Заголовок пятого уровня</h5>
 <h6>Заголовок шестого уровня</h6>
</body>
```

Для **полужирного начертания** применяется два тега: <b> и .

```
<b>Жирное начертание шрифта</b>
<strong>Сильное выделение текста</strong>
```

Для **курсивного начертания** используются два тега: <i> и .

```
<i>Курсивное начертание шрифта</i>
<em>Выделение текста</em>
```

[1] В настоящее время параметры тегов также поддерживаются, но предпочтение отдается использованию стилей.

Следует отметить, что теги <b> и , также как <i> и хотя и похожи по своему действию, являются не совсем эквивалентными и заменяемыми. Первый тег <b> — является тегом физической разметки и устанавливает жирное начертание текста, а тег — тегом логической разметки и выделяет помеченный текст.

Маркированный список формируется с помощью контейнера<ul>, а каждый пункт списка начинается с тега <li>, как показано ниже.

```
<ul>
 <li>Значение №1</li>
 <li> Значение №2</li>
 <li> Значение №3</li>
</ul>
```

В списке непременно должен присутствовать закрывающий тег , иначе возникнет ошибка. Закрывающий тег хотя и не обязателен, но советуем всегда его добавлять, чтобы четко разделять элементы списка.

Нумерованные списки формируются так:

```
<ol>
 <li> Значение №1</li>
 <li> Значение №2</li>
 <li> Значение №3 </li>
</ol>
```

Бегущая строка задается тегом <marquee>.

1.4. Таблицы

Таблицы могут использоваться не только для более наглядного предоставления информации, но и для верстки веб-страниц. Таблица с невидимой границей представляет собой словно модульную сетку, в блоках которой удобно размещать элементы веб-страницы. Однако в настоящее время верстка страниц в основном осуществляется с помощью слоев.

Рассмотрим пример добавления таблицы, показанной ниже.

1	2
3	4

Для этого надо написать следующий код:

```
<table border="1">
 <tr>
  <td> 1</td>
  <td> 2</td>
 </tr>
 <tr>
  <td> 3</td>
  <td> 4</td>
 </tr>
</table>
```

Для объединения двух и более ячеек в одну используются атрибуты colspan и rowspan тега <td>.

Рассмотрим пример добавления таблицы с объединенными ячейками как показано ниже.

1	
3	4

Для этого надо написать следующий код:

```
<table border="1">
 <tr>
  <td colspan=2> 1</td>
 </tr>
 <tr>
  <td> 3</td>
  <td> 4</td>
 </tr>
</table>
```

Лекция 2. Ссылки, рисунки, фреймы

2.1. Ссылки

Ссылки являются основой гипертекстовых документов и позволяют переходить с одной веб-страницы на другую. Особенность их состоит в том, что сама ссылка может вести не только на HTML-файлы, но и на файл любого типа, причем этот файл может размещаться совсем на другом сайте

Общий синтаксис создания ссылок следующий:

```
<a href="URL">ключевая фраза</a>
```

Атрибут **href** определяет URL (Universal Resource Locator, универсальный указатель ресурса). Текст, расположенный между тегами <a> и , по умолчанию становится синего цвета и подчеркивается.

Адрес ссылки может быть абсолютным или относительным. Абсолютные адреса должны начинаться с указания протокола (обычно http://) и содержать имя сайта. Относительные ссылки ведут отсчет от корня сайта или текущего документа.

С тегом <a> можно использовать параметры target и title.

target указывает окно, в котором нужно открыть ссылку. По умолчанию, при переходе по ссылке документ открывается в текущем окне или фрейме. Синтаксис следующий:

```
<a target="имя окна">...</a>
```

В качестве значения используется имя окна или фрейма, заданное атрибутом name. Если установлено несуществующее имя, то будет открыто новое окно. В качестве зарезервированных имен применяются следующие.

- _blank — загружает страницу в новое окно браузера.
- _self — загружает страницу в текущее.
- _parent — загружает страницу во фрейм-родитель, если фреймов нет, то это значение работает как _self.
- _top — отменяет все фреймы и загружает страницу в полном окне браузера, если фреймов нет, то это значение работает как _self.

Параметр **title** добавляет поясняющий текст к ссылке в виде всплывающей подсказки, которая отображается при наведении мыши на ссылку. Пример:

университет

Для создания **ссылки на адрес электронной почты** в параметре href надо указать mailto:адрес электронной почты. Например,

администратор

Кроме внешних ссылок можно использовать внутренние ссылки. Для этого вначале необходимо сделать метку в соответствующем месте страницы и дать ей имя при помощи атрибута name тега <a> . Для создания метки используется символ решетки (#).

Пример. Создадим в конце web-страницы ссылку, которая введет в начало документа. Для этого в начале документа, задаем метку:

```
<body>
<a name="ttt"></a>
Затем в конце страницы задаем ссылку:
<a href="#ttt">В начало</a>
 </body>
```

Также можно сделать ссылку на метку другого документа. Приме:

```
<a href="text.html#ttt">Как написано на сайте…</a>
```

2.2. Рисунки

Для добавления изображения используется тег <img>, атрибут src которого определяет адрес графического файла. Общий синтаксис добавления изображения будет следующий.

```
<img src="URL рисунка" >
```

Полезными являются атрибуты alt, width, height. Атрибут **alt** предназначен для создания альтернативного текста. При помощи атрибутов **width** и **height** задается размер рисунка. Следует отметить, что заданный размер не влияет на скорость загрузки.

2.3. Карты изображения

Карты изображения позволяют привязывать ссылки к разным областям одного изображения. По сути это графические изображения с размеченными областями. Для создания карты изображения нужно дать команду

<img src="map.gif" usemap="#map1">

Далее необходимо определить области на карте следующим образом:

<map name="map1">

<area shape="форма" coords="список координат" href="ссылка">

…

</map>

Возможны следующие варианты форм:

Форма	Название	Задаваемые координаты
rect	прямоугольник	координаты двух вершин
poly	многоугольник	координаты всех вершин
circle	окружность	координаты центра и радиус

Пример. <area shape="poly" coords="282,2, 222,1, 203,6, 210,27, 212,33, 282,31, 301,33, 301,28" href="1.html">

2.4. Фреймы

Окно браузера может быть разделено на фреймы, т.е. на самостоятельные области. В каждую из этих областей можно загружать свои html-страницы. Рассмотрим следующий пример (строки для удобства комментирования пронумерованы):

1) <HTML>
2) <HEAD>
3) <TITLE>ФРЕЙМЫ</TITLE>
4) </HEAD>
5) <FRAMESET ROWS="45%,45%,*">
6) <FRAME SRC="B611.HTM">
7) <FRAME SRC="B612.HTM">
8) <FRAME SRC="B613.HTM">

9) </FRAMESET>

10) </HTML>

В 5-й строке вместо тега <body> используется тег FRAMESET. Параметр ROWS указывает на то, что страница должна быть разбита на горизонтальные фреймы. Если же надо было разбить на горизонтальные фреймы, то надо указать COLS. В 5-й же строке указано, что фреймов должно быть 3 – на 1-й и 2-й отводится 45% высоты экрана, на третий – все, что осталось.

В 6-й, 7-й, 8-й строках непосредственно подключается содержимое фреймов, которое берется из файлов B611.HTM, B612.HTM, B613.HTM.

В настоящее время фреймы практически не используются, так как более удобным является использование слоев. Тем не менее активно используются плавающие фреймы, для задания которых используется тэг <**IFRAME**>. Плавающие фреймы в поддерживаются не всеми браузерами. Плавающий фрейм можно расположить в любом месте страницы, для этого надо указать код

<iframe src="111.html" name="iframe1" width="300" height="300" align="left"></iframe>

Здесь будет подключен файл 111.html.

2. CSS

Лекция 3. Каскадные таблицы стилей

3.1. Синтаксис CSS

Определение. Таблицы стилей (или каскадные таблицы стилей, CSS) – это описание правил, задающих параметры представления отдельных элементов на языке HTML

CSS появились одновременно с HTML 4.0 (Dynamic HTML). Сам термин «каскадные таблицы стилей» был предложен в 1994 году. Все объявления CSS называются *селекторами*, записываются в фигурных скобках.

Примеры использования синтаксиса.

1. Для того чтобы все заголовки первого уровня (h1) были зеленого цвета с размером шрифта 15 пикселей, надо записать

h1 {color: green; size: 15px;}

Свойства разделяются точкой с запятой

2. Одно свойство можно присвоить нескольким тэгам

h1, h2, h3 {color: green; size: 15px;}

3. Свойство наследования. Если по h3 текст должен быть зеленым, то и между em он тоже будет желтым.

<h3> Часть 4.Очень большая и хорошая</h3>

4. Контекстные селекторы. Если необходимо, чтобы текст между тегами em, которые между h3, был зеленым, а в других случаях каким-то другим, то надо задать

h3 em {color:green}

5. Использование классов.

Определили h1:

h1 {color: green; size: 15px;}

Чтобы некоторые заголовки были красными, необходимо задать

h2.red {color:red}

Использование:

<h2 class=red>Это красный заголовок</h2>

Замечание. В таблице стилей можно указать просто без указания родительского тэга:

.red {color:red}

6. Можно использовать маску * для задания всех селекторов

7. Еще один символ маски >. Задает свойства для определенных подэлементов

ol>li {color:green}

8. Для определения элемента в DHTML используется id. Поэтому можно задавать свойства через идентификаторы

#cont {color:yellow}

И когда встретится идентификатор cont, то текст этого элемента будет желтым.

9. Для создания прямоугольной области используется тег .

10. Для создания строчной области используется тег .

11. Использование псевдоклассов. Динамические псевдо-классы: :hover, :active, и :focus.

Пример использования псевдо-классов. Указав следующие описание стилей:

a:hover { background: yellow }

получим, что при наведении курсора на ссылку, область под ней станет желтой.

3.2. Размещение каскадных таблиц

3.2.1. Свойства можно определять непосредственно с тегом. Например, вставка следующего тега сделает текст красным на синем фоне

красный на синем

3.2.2. Таблицу можно разместить между тегами <style> и </style>. Например,

```
<head>
<style>
a { color: #33ADDB;
        background-color: inherit;}
a:hover {     color: #575757;
        background-color: inherit;}
h1 {   font: bold 1.8em Arial, Sans-Serif;
        letter-spacing: -1px;
        margin: 0;
        padding: 0;}
h1 a { text-decoration: none;}
</style>
</head>
```

И теперь гиперссылки и заголовки первого уровня будет определяться в соответствии с указанными правилами.

3.2.3. Таблицу стилей можно разместить в отдельном файле. А подключение затем выполняется так:

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

3.3. Верстка

Наиболее популярным является деление макетов по ширине и количеству колонок. Выделяют следующие типы макетов[2], связанных с шириной:

- фиксированные;
- резиновые;
- эластичные;
- адаптивные;
- комбинированные.

Фиксированный макет располагается по центру окна браузера, а его ширина ограничивается заданными размерами в пикселах.

При создании **резинового макета** задается в процентах ширина колонок таким образом, что макет занимает всю свободную ширину окна браузера.

Эластичный макет по своему виду может не отличаться от фиксированного или резинового макета. Размер элементов задаётся в em, привязанных к размеру шрифта. Значение em можно использовать не для всех элементов, оставляя ширину некоторых фиксированной.

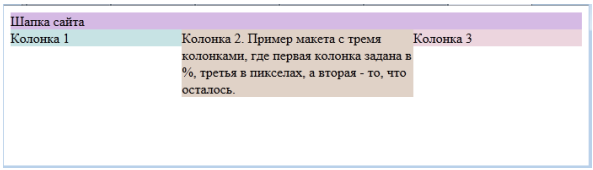
Адаптивный макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину макета, число колонок, размеры изображений и текста. Для этого заготавливается несколько стилевых правил или файлов под разный диапазон разрешений, выбор правил происходит через скрипты или CSS3, которые и определяют нужную для этого информацию о пользователе.

[2] Более подробно о верстке можно прочесть на сайте http://htmlbook.ru.

Комбинированный макет предполагает использование разной ширины для отдельных частей страницы, например, шапку и подвал делают резиновыми, а контент фиксированным.

Пример макета с тремя колонками, где первая колонка задана в %, третья в пикселах, а вторая – то, что осталось.

```
<html>
 <head>
<style>
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; width: 30%; }
.col2 { background: #E0D2C7; left: 30%; right: 200px; }
.col3 { background: #ECD5DE; right: 0; width: 200px; }
</style>
  <title>Три колонки</title>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col2">Колонка 2. Пример макета с тремя колонками, где первая колонка задана в %, третья в пикселах, а вторая - то, что осталось.</div>
   <div class="col3">Колонка 3</div>
  </div>
</body>
</html>
```



3.4. Параметры CSS, управляющие положением на странице

position: absolute | fixed | relative | static | inherit

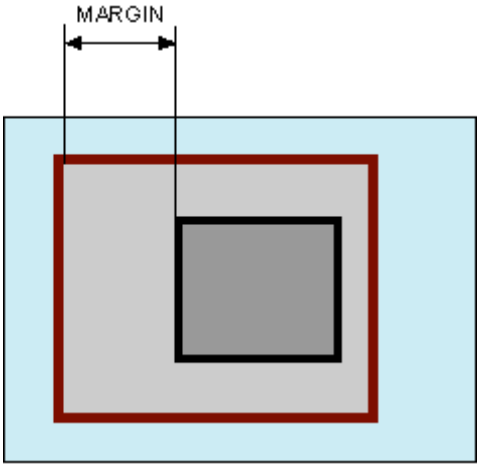
Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице. **absolute** указывает, что используются абсолютные координаты. **relative** указывает, что используются относительные координаты. **inherit** наследует значение родителя.

left задает положение относительно левого края контейнера. **top** задает положение относительно верхнего края контейнера. Задаются в процентах или пикселях.

z-index указывает на то, какой элемент должен располагаться выше при перекрытии. Измеряется в единицах.

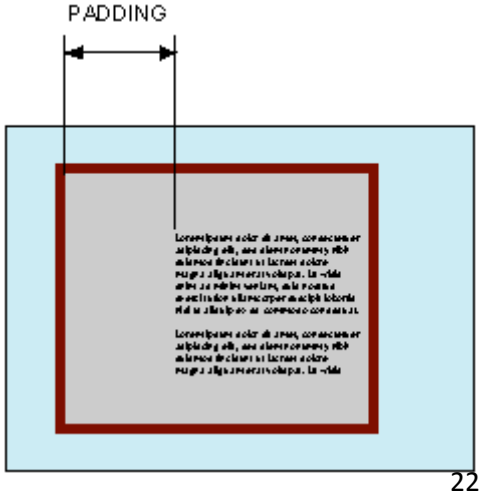
visibility определяет видимость элемента. Принимает значения visible (видимый), hidden (скрытый), inherit (наследуется от родительского элемента).

margin устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента.



Разрешается использовать одно, два, три или четыре значения, разделяя их между собой пробелом. Если задано одно значение, то будут установлены четыре одинаковых отступа. Если два, то первое значение – отступ сверху и снизу, второе – слева и справа. Если определены три значения, то первое значение задает отступ от верхнего края, второе — одновременно от левого и правого края, а третье — от нижнего края. Четыре параметра определяют отступ от верхнего, правого, нижнего и левого края.

padding устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое. Также как и у margin возможны несколько вариантов значений.



3. JavaScript

Лекция 4. Основные сведения о JavaScript

4.1. Знакомство с JavaScript

JavaScript – объектно-ориентированный сценарный язык программирования.

Области применения:

- динамическое создание web-страницы;
- проверка достоверности полей формы до передачи на сервер;
- вывод сообщений для пользователей;
- учет особенностей браузера и монитора пользователя для корректного отображения;
- составной элемент технологии AJAX;
- счетчики посещаемости.

Сценарий JavaScript встраивается в HTML-документ с помощью тега **<script>**. При этом скрипт может располагаться в произвольном месте страницы.

Пример. Выводит слово «Привет» в теле документа.

```
<html>
<head>
</head>
<body>
<script>
document.write("Привет!");
</script>
</body>
</html>
```

Здесь объект document – это HTML-документ, загруженный в окно браузера. Метод write записывает в тело HTML-документа строку "Привет!".

Переменные.

Можно объявлять с помощью ключевого слова var, а можно использовать без объявления.

```
var th=12
```

age=12
var dd=”qwe”

4.2. Операторы языка JavaScript

В данном подразделе рассматриваются основные операторы языка JavaScript. При этом полный список операторов не ограничивается приведенными.

4.2.1. *Операторы присваивания*

=	Присваивание
+=	Сложение или слияние строк (x=x+2 аналогично x+=2)
–=	Вычитание (x=x-3; аналогично x-=3)
*=	Умножение
/=	Деление

4.2.2. *Операторы сравнения*

>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно

4.2.3. *Унарные операторы*

++	Увеличение значения переменной на единицу. Может применяться как префикс и как суффикс

--	Уменьшение значения переменной. Может применяться как префикс и как суффикс

Чтобы увидеть разницу между использованием унарных операций как префикс и суффикс, изучите работу двух кодов:

```
a=1                          a=1
document.write(a++)          document.write(++a)
```

В первом случае будет выведено значение 1, во втором – 2.

4.2.5. *Условные операторы*

if-else

Пример. Поиск максимального значения из двух чисел

```
if(a<b) {max=b} else {max=b}
```

Последний пример можно реализовать при помощи такой более компактной конструкции:

```
max=(a<b) ?b:a
```

switch - case

Определяет действия в зависимости от значения переменной

```
switch(variable) {
  case value_1: {
    //блок операторов_1
    break;
  }
  case value_2: {
    //блок операторов_2
    break;
  }
  case value_n: {
    //блок операторов_n
    break;
  }
  default: {
    //блок операторов по умолчанию
  }
}
```

4.2.6. *Операторы цикла*

for

Пример вывода пирамидки из горизонтальных линий.

Для этого достаточно набрать код:

```
for(i=1;i<=10;i++)

document.write("<hr width=\""+i*10+"%\">")
```

Таким образом, первый аргумент – это начальное значение, второй – конечное, третий – изменение. В примере использована конструкция вида \", которая называется экранирование. Цель – поместить внутрь строковой переменной кавычки для вывода на экран.

FOR-IN позволяет получить полный список свойств, методов, событий определенного объекта. Например.

```
for(props in document)

document.write(props+"<br>");
```

WHILE

Пример. Вывести таблицу умножения для введенного числа.

```
a=prompt("Введите число",6)
i=1
while(i<=10)
{document.write(a+"x"+i+"="+a*i+"<br>")
i++}
```

Кроме этих операторов в организации цикла могут участвовать еще два оператора: **break** (выход из цикла) и **continue** (переход на следующий шаг).

4.2.7. *Функции*

Функции определяются так:

```
function F_name(аргументы)
{тело функции}
```

Замечания

1. Операторы JavaScript напоминают общеизвестные операторы языка C++.
2. После оператора надо ставить точку с запятой, но если в строке используется только одна команда, то точку с запятой можно не ставить.
3. Для объединения группы операторов используются фигурные скобки.
4. Однострочные комментарии задаются при помощи двух слешей //
5. Многострочные комментарии при помощи конструкции /* в начале и */ в конце.

4.3. Объекты JavaScript

4.3.1. Объект **Array**

Для создания массива можно воспользоваться одним из трех способов:

abc=new Array()

abc=new Array(10)

abc=new Array("Синий","Красный","Зеленый")

объект Array имеет одно свойство **length** – число элементов массива.

Методы объекта Array

- join( ). Слияние элементов массива в строку. Через параметр передается разделитель элементов. По умолчанию разделителем служит запятая.
- reverse( ). Меняет порядок элементов массива на обратный.

Пример. Создание аналога gif-рисунков.

```
<html>
<head>
<title>Создаем анимацию
</title>
<script>
i=0
function ShowNextImage()
{
        iii.src=i + ".jpg"
        i++
        setTimeout ("ShowNextImage()",1000)
        if ((i>=4)) i=0
}
```

```
</script>
</head>
<body>
 <img src="0.jpg" name="iii">
 <script>
 ShowNextImage()
</script>
</body>
</html>
```

В данном примере заранее заготовлены рисунки с названиями 0.jpg … 4.jpg. Если бы мы хотели использовать произвольные названия, то вместо команды

```
iii.src=i + ".jpg"
```

следовало бы использовать команду

```
iii.src=fname[i]
```

где fname – массив содержащий названия файлов.

4.3.2. Объект **Date**

Предоставляет функции для работы с датой и временем.

Создание объекта

```
newDate= new Date()
```

Методы:

getDate() – возвращает число месяца как целое число.

getDay() – возвращает день недели.

getHours()

getMinutes()

getSeconds()

getYear()

getMonth()

getTime() – возвращает количество миллисекунд прошедших с 0:0:0 1 января 1970 года.

Если вместо **get** указать **set**, то метод устанавливает соответствующее значение в переменной. Например, newDate.getYear(2015) становит значение год, равное 2015.

Следующий пример при обновлении страницы выводит число секунд, которые пользователь был на сайте.

```
<html>
<script>
function person_in()
{
        t1=new Date()
}
function person_out()
{
        t2=new Date()
        t3=t2.getTime()-t1.getTime()
        t3=Math.round(t3/1000)
        alert("Вы были на сайте " + t3+ "c")
}
</script>
<body onLoad="person_in()" onUnLoad="person_out()">
</body>
</html>
```

В данном примере использованы две написанные функции:

- person_in(), которая вызывается при открытии страницы – обработчик onLoad записан в теге body. Данная функция сохраняет значение времени в переменной t1.
- person_out(), которая вызывается при открытии страницы – обработчик onUnLoad. В настоящее время большинство браузеров не поддерживает этот обработчик.

4.3.3. Объект Math

Предназначен для работы с математическими константами и функциями. Приведем некоторые свойства и методы.

Свойства: E, PI, SQRT, SQRT1_2.

Методы: cos(), sin(), exp(), abs() – модуль, round() – округление, ceil() - округление вверх, floor() – округление вниз, pow(число1, число2) – возведение в степень, sqrt() – корень квадратный.

4.3.4. Объект **String**

Предназначен для работы со строковыми значениями.

Создать переменную можно двумя способами:

MyString=”значение”

MyString=new String(”значение”)

Одно свойство – length – длина переменной.

Методы (частично):

big() – возвращает строку с добавлением тегов <big>…</big>;

bold() – возвращает строку с добавлением тегов …;

charAt(позиция) – возвращает символ, стоящий на указанной позиции;

fontcolor(“цвет”) – изменяет цвет;

sub(), sup() – индексы;

substring(позиция1, позиция2) – возвращает подстроку, начинающуюся символом в первой позиции и заканчивающуюся перед второй позицией;

toLowerCase() – преобразует исходную строку в строку со строчными символами;

toUpperCase() – преобразует исходную строку в строку с заглавными символами.

Пример. В текстовом элементе формы по надписи пробегает заглавная буква.

```
<form name="f1">
<input type="text" name="t1" size="100">
</form>
<script>
str="Сыктывкарский государственный университет им. Питирима Сорокина"
i=0
function qwe()
{
        i++
        document.f1.t1.value=str.substring(0,i) + str.charAt(i).toUpperCase() + str.substring(i+1, str.length)
        setTimeout("qwe()",300)
}
qwe()
```

</script>

4.3.5. Некоторые встроенные функции JavaScript

- isNaN(значение) – возвращает истину, если значение не является числом;
- parseFloat(строка) – преобразует строку в число с плавающей точкой;
- parseInt(строка,основание) – преобразует строку в целое число по указанному основанию;
- typeOf(объект) – возвращает тип указанного объекта как строку.

Лекция 5. Формы HTML. Обработка форм с помощью JavaScript

5.1. Основные элементы формы

5.1.1. *Форма* – это элемент HTML, позволяющий передавать информацию на web-сервер, где информация будет обработана. С помощью форм организуются тесты, голосования, опросы. Заметим, что html-формы сами по себе позволяют только организовывать ввод информацию. Для обработки форм необходимо использовать языки программирования, для обработки на стороне клиента можно использовать, например, язык JavaScript, а на стороне сервера – например, PHP, Perl, C.

Форма задается тегами **<form></form>.** Все остальные элементы формы располагаются между этими тегами. Тег <form> может иметь несколько параметров:

- name - имя формы. Необходимо, если на странице несколько форм.
- action - определяет URL-адрес, по которому будет отправлена информация введеная пользователем.
- method - определяет способ отправки информации.
- target - указывает имя окна, в котором будут отображаться результаты обработки отправленной формы.

5.1.2. Рассмотрим элементы, располагающиеся в форме. Начнем с *текстового поля*. Оно задается тегом <**input**>. Пример использования.

Исходный код	Вид в браузере
<form name="forma1"> <input type="text" name="t1" size="20" maxlength="50" value="Введите текст"> </form>	Введите текст

- Параметры тестового поля:
- name - имя элемента,
- type - тип элемента (в данном случае - text),
- size - размер текстового поля в символах, которые одновременно будут видны, при вводе большего количества символов, они будут прокручиваться,
- maxlength - максимальное количество символов, которое можно ввести в поле, если опустить этот параметр, то число символов будет неограниченным,
- value - текст, который будет отображаться (его можно стереть), при отсутствии этого параметра поле будет пустым.
- disabled - блокирует поле от любых изменений,
- readonly - делает поле доступным только для чтения.

5.1.3. *Текстовое поле для ввода пароля*. Представляет собой текстовое поле с тем отличием, что вводимый текст не отображается, вместо него появляются специальные символы, например звездочки. Используется при вводе паролей. От текстового поля отличается только параметром type="password".

5.1.4. *Флажки*. Начнем с примера.

Ваши любимые предметы:

☐ Математика ☐ Литература ☐ Физическая культура

Флажки задаются тегом <input>, причем один тег задает один флажок. Поэтому чтобы задать три флажка, надо три раза писать input. Для задания указанного выбора необходимо задать следующий код:

<form name="forma1">

Ваши любимые предметы:

<input type="checkbox" checked>Математика

<input type="checkbox" > Литература

<input type="checkbox" > Физическая культура

</form>

Рассмотрим параметры тега <input>:

- type - тип элемента (в данном случае - checkbox),
- name - имя элемента, указывает программе обработчику формы, какой пункт выбрал пользователь,
- value - значение элемента, указывает программе обработчику формы значение пункта, который выбрал пользователь,
- checked - им обычно помечают наиболее вероятные для выбора пункты, пользователь щелчком мыши может выбрать другие пункты.

5.1.5. *Переключатели*. В отличии от флажков, здесь можно выбрать только один пункт. В связи с этим значения параметра name должны быть одинаковы для всех элементов группы. Параметр type="radio", все остальные такие же, как у флажков.

Пример.

<form name="forma1"> Укажите ваш пол: <input type="radio" name="sex" value="man" checked>мужской <input type="radio" name="sex" value="woman"> женский </form>	Укажите ваш пол: мужской женский

5.1.6. *Кнопки*. Существует четыре типа кнопок:

- submit - кнопка отправки содержимого формы web-серверу. Ее параметры:
  - type="submit" - тип кнопки,
  - name - имя кнопки,
  - value - надпись на кнопке.
- image - графическая кнопка отправки содержимого формы web-серверу. Для ее использования необходимо подготовить картинку кнопки, а потом использовать ее в виде кнопки. Ее параметры:
  - type="image" - тип графической кнопки,

- name - имя кнопки,
- src - адрес картинки для кнопки.

- reset - кнопка, позволяющая восстановить все значения по умолчанию в форме. Ее параметры:
  - type="reset" - тип кнопки очищения,
  - name - имя кнопки,
  - value - надпись на кнопке.
- button - произвольная кнопка, ее действия назначаются вами с использованием языков программирования, т.е. сама она делать ничего не умеет. Ее параметры:
  - type="button" - тип произвольной кнопки,
  - name - имя кнопки,
  - value - надпись на кнопке.
  - onclick - обработчик события - указывает, что делать при щелчке по кнопке. Вообще, у этого типа кнопок есть и другие события (например, двойной щелчок), но здесь мы не будем их рассматривать.

Кнопки можно задавать и при помощи тегов <**button**> </**button**>. Возможности у таких кнопок несколько, они могут иметь содержимое в виде текста или картинки. Этот тег имеет следующие параметры:

- type - тип кнопки, может принимать значения:
  - reset - кнопка очистки формы,
  - submit - кнопка отправки данных,
  - button - кнопка произвольного действия.
- name - имя кнопки,
- value - надпись на кнопке.

5.1.7. *Поле для файлов*. Поле для ввода имени файла, сопровождаемое кнопкой Browse (Обзор), при щелчке по которой открывается окно просмотра дерева папок компьютера, где можно выбрать нужный файл. Выбранный файл присоединяется к содержимому формы при отправке на сервер.

Пример.

<form name="forma1">

<input type="file" name="load" size="50">

</form>

5.1.8. *Поле для ввода текста*. Для больших текстов, например для почтовых сообщений, удобно использовать именно этот элемент. Он создается тегами <**textarea**> </**textarea**> и имеет следующие параметры:

- name - имя поля,
- cols - ширина поля в символах,
- rows - количество строк текста, видимых на экране,
- wrap - способ переноса слов:
    - off - переноса не происходит,
    - virtual - перенос отображается, но на сервер поступает неделимая строка,
    - physical - перенос и на экране и при поступлении на сервер.
- disabled - неактивное поле,
- readonly - разрешено только чтение.

5.1.9. *Раскрывающиеся списки*. Списки бывают с возможностью выбора одного элемента и с множественным выбором. Задются и те, и другие с помощью тегов <**select**> </**select**>, внутри которых располагаются элементы значений, заданных тегом <**option**>

Параметры этих тегов.

<select>:

- name - имя списка. Каждый выбранный элемент списка при передаче на сервер будет иметь вид: name.value, где значение (value) берется из тега option.
- size - определяет количество видимых элементов в списке: 1 - простой раскрывающийся список, больше 1 - список с полосой прокрутки.
- multiple - разрешает выбор нескольких элементов списка.

<option>:

- selected - им помечают наиболее вероятный для выбора элемент списка, если список со множественным выбором, то можно пометить несколько пунктов.
- value - значение, которое будет отправлено серверу, если пункт выбран.

Пример.

```
<form name="forma1">
Какой язык вы хотите изучать:
<select name="language" size="1">
<option selected value="html">html
<option value="php">php
<option value="java">java
</select><br><br>
Какое время вы готовы на это потратить:<br>
<select name="time" size="3">
<option selected value="1">1 месяц
<option value="2">2 месяца
<option value="3">3 месяца
</select><br><br>
Какие дни недели для занятий вас устроят:<br>
(выбирайте с нажатой клавишей ctrl)<br>
<select name="day" size="7" multiple>
<option selected value="mon">понедельник
<option value="tue">вторник
<option value="wen">среда
<option selected value="thu">четверг
<option value="fri">пятница
<option value="sat">суббота
<option value="san">воскресенье
</select>
</form>
```

Результат

Какой язык вы хотите изучать: html

Какое время вы готовы на это потратить:

1 месяц
2 месяца
3 месяца

Какие дни недели для занятий вас устроят:
(выбирайте с нажатой клавишей ctrl)

понедельник
вторник
среда
четверг
пятница
суббота
воскресенье

Можно использовать теги <**optgroup**> </**optgroup**>, позволяющие группировать элементы списка по каким-либо признакам. Например, мы хотим задать каталог сайтов в виде списка, тогда удобнее разбить его на группы по интересам. Для этого нужно поступить следующим образом

```
<form name="forma1">
Каталог сайтов:<br>
<select name="catalog" size="9">
<optgroup label="Компьютеры">
<option value="1">интернет</option>
<option value="2">мобильники</option>
<option value="3">hardware</option>
</optgroup>
```

Каталог сайтов:

```
<optgroup label="Работа">
<option value="4">вакансии</option>
<option value="5">трудоустройство</option>
<option value="6">резюме</option>
</optgroup>
<optgroup label="Дом">
<option value="7">здоровье</option>
<option value="8">красота</option>
<option value="9">дети</option>
</optgroup>
</select>
</form>
```

Замечание.В данном случае необходимо указывать закрывающие теги </option>.

5.1.10. Надписи. Все элементы формы можно связать с их надписями при помощи элемента <**label**> и его параметра for, значением которого является имя элемента, с которым связываем надпись. Например.

```
<form name="forma1">
<label for="load">Выбирайте файл: </label>
<input type="file" name="load" size="30">
</form>
```

5.2. Обращение к элементам формы

5.2.1. Обращение к значениям текстового поля и текстовой области

Первый способ по именам элементов формы:

document.fname.tname.value

Здесь fname – название формы, ftext – название поля или области.

Кроме этого все формы на странице образуют коллекцию, в свою очередь все элементы формы образуют также коллекцию. Поэтому доступно такое обращение:

document.forms[0].elements[2].value

5.2.2. Проверка checkbox

if (document.fname.check1.checked) {операторы}

5.2.3. Выпадающий список

Сначала определяется выбранный элемент, затем определяется значение:

```
idx=document.fname.sel1.selectedIndex
a=document.fname.sel1.options[idx].text
```

5.2. Обработчики событий

Приведем события, которые можно определить объектам формы и документа.

- **onBlur** происходит, когда поля формы или окно документа теряет фокус.
- **onFocus –** приобретение фокуса.
- **onChange –** изменение значения select.
- **onClick** – щелчок мыши.
- **onMouseOver** – при наведении мыши на объект.
- **onMouseOut** – при отведении мыши с объекта.
- **onLoad** – при загрузке документа.
- **onUnload** – при закрытии страницы.
- **onSelect** – при выделении текста.
- **onSubmit** – при нажатии кнопки Принять.

Лекция 6. Объектная модель Dynamic HTML

Динамический HTML (DHTML) – совокупность приемов, позволяющих динамически изменять оформление и содержание веб-страницы в ответ на действия пользователя. DHTML является продуктом взаимодействия трех технологий: языка HTML, каскадных таблиц стилей (CSS) и языка сценариев (JavaScript). Для предоставления сценариям доступа к HTML и CSS содержимое документа представляется в виде дерева объектов в программной модели.

6.1. Window

Объект window является вершиной иерархии объектной модели. Все остальные объекты являются дочерними или более отдаленными потомками объекта window. Браузер создает, как правило, единственный объект window,

когда открывает документ в окне, однако если документ содержит фреймы (элементы frame и iframe), то дочерние объекты window создаются также для каждого фрейма. Доступ к дочерним окнам возможен через коллекцию frames родительского окна.

Полный список свойств, методов, событий объекта window (как и у других объектов) можно получить так:

```
<script>
sprops="<H2>Свойства объекта window</H2>"
for(props in window)
        sprops+="<b>"+props+"</b><br>";
document.write(sprops);
</script>
```

Свойства объект window:

- parent;
- self;
- top;
- document;
- event;
- history;
- location;
- navigator;
- screen.

Методы:

- alert – вывод сообщения;
- prompt – ввод данных;
- confirm – подтверждение;
- close – закрыть окно;
- open – открыть окно;
- blur – убрать фокус;
- focus – установить фокус;

- navigate – загрузить страницу;
- scroll – развернуть страницу на указанный размер;
- setTimeout() – загрузить функцию через указанное время;
- cleartimeout() – удалить загрузчик функции;
- showModalDialog – открыть модальное окно.

6.2. document

Рассмотрим некоторые свойства, коллекции и методы документа, загруженного в окно браузера.

Свойства

- linkColor – цвет ссылок;
- fgColor – цвет щрифта;
- bgColor – цвет фона.

Коллекции

- all – все теги на странице;
- anchors – все ссылки;
- forms – коллекция форм.

Методы:

clear – очистить документ;

close – закрыть;

open – открыть;

write – записать в документ.

6.3. Объект navigator

Объект navigator содержит информацию о браузере. Свойства:

- userAgent содержит строку, которая передается в HTTP заголовке на веб-сервер с каждым запросом. Эта строка содержит информацию о самом браузере и его версии, операционной системе и ее версии и некоторую дополнительную информацию.
- appName содержит имя.
- appCodeName – "Mozilla" для большинства браузеров

- appVersion, appMinorVersion, platform, systemLanguage и другие содержат информацию о версии браузера и операционной системы, языковые настройки и другое.

6.4. Объект location

Объект location содержит полную информацию об адресе страницы.

Свойства:

- href содержит всю строку URL;
- protocol – протокол передачи данных;
- host – имя сервера + порт (www.ya.ru:80);
- hostname – имя сервера (www.ya.ru);
- port – порт;
- pathname – путь;
- search – строка запроса (?a=html&page=1);
- hash – закладка, ссылка внутри страницы (#label1).

6.5. Объект history

Объект history содержит информацию об адресах, посещенных пользователем с момента открытия браузера.

Методы:

- back() – переход на предыдущую страницу. Эквивалентно нажатию кнопки "Назад" в браузере.
- forward() – переход на следующую страницу. Эквивалентно нажатию кнопки "Вперед" в браузере.
- go(n) – переход на n позиций в индексе

6.6. Объект screen

Объект screen содержит информацию о мониторе: ширина – width, высота – height, глубина цвета – colorDepth.

6.7. Объект event[3]

Событие – это сообщение, отправляемое в ответ на некоторое действие, например, завершение загрузки документа, перемещение или щелчок мыши, нажатие клавиши. Обработчик события – это, как правило, функция, написанная на языке сценариев (например, JavaScript), которая выполняет действия, когда произошло соответствующее событие.

Объект window.event создается браузером автоматически в момент возникновения. Следует подчеркнуть, что модель объекта event в Internet Explorer существенно отличается от используемой в других браузерах. Тем не менее, целый ряд свойств оказывается одинаковым во всех современных браузерах.

- type – тип события без приставки "on". Например, событие onclick имеет тип click.
- button – нажатая кнопка мыши (0 – ни одна, 1 – левая, 2 – правая, 3 – левая и правая, 4 – средняя, 5 – левая и средняя, 6 – правая и средняя, 7 – все три). Это свойство имеет смысл для событий мыши. В случае остальных событий имеет значение 0 независимо от действительного состояния кнопок.
- clientX, clientY – координаты указателя мыши относительно верхнего левого угла клиентской области окна браузера.
- screenX, screenY - координаты указателя мыши относительно верхнего левого угла дисплея.
- keyCode – целочисленный код клавиши, которая вызвала событие клавиатуры. Коды клавиш различаются для различных аппаратных платформ (PC или Macintosh).
- shiftKey, ctrlKey, altKey – истина (true), если во время события нажата также клавиша Shift, Ctrl или Alt, соответственно.
- cancelBubble – запрещает (true) или разрешает (false, по умолчанию) передачу события вверх по иерархии от элемента-источника к родительско-

[3] Подраздел написан по материалам сайта http://www.intuit.ru

му элементу. Каждое событие происходит на единственном элементе-источнике. Если элементу-источнику не сопоставлена функция-обработчик, событие передается вверх по дереву элементов, пока не встретит функцию-обработчик или не достигнет корневого элемента. Если функция-обработчик найдена, но не содержит cancelBubble=true, то передача вверх продолжается, если содержит, то прерывается.

Отметим некоторые свойства, доступные в Internet Explorer:

- элемент-источник события доступен в IE как свойство srcElement, в Mozilla Firefox и других как target.
- события onmouseover и onmouseout в Internet Explorer показывают свойства-объекты откуда (fromElement) и куда (toElement). В Mozilla Firefox эти объекты являются свойством relatedTarget.

6.8 Обращение к свойствам таблицы стилей

Если известен идентификатор объекта (например, id1) и необходимо обратиться к свойству left, то можно обратиться так:

id1.style.left=”100 px”

6.9. Свойства и методы управления содержимым

Свойства:

- innerText – заменяет текст, содержащийся в элементе;
- outerText – заменяет весь элемент;
- innerHTML – заменяет текст и код элемента;
- outerHTML заменяет весь текст и код.

Пример. Имеется заголовок третьего уровня

<h3 id="heading1">Старый заголовок</h3>

Далее в скрипте JavaScript даем командуы

objHead1=document.all["Heading1"]

objHead1.innerText="Новый заголовок "

Методы:

- insertAdjacentText(положение, текст) – вставляет только текст;

- insertAdjacentHTML(положение, текст) – вставляет текст и HTML.

Варианты положения: BeforeBegin, AfterBegin, BeforeEnd, AfterEnd.

Пример. objHead1.incertAdjacentText(“AfterBegin”, “Привет”).

Пример реализации раскрывающихся списков с использованием приведенных свойств и методов.

```
<html>
<head>
<title>
Раскрывающийся список сиспользованием Inner
</title>
<script>
i1t=0
i2t=0
i3t=0
function qwe()
{
 Obj=window.event.srcElement
 if(Obj.id.length==2)
 { if(eval(Obj.id+"t==0"))
   {

Obj.insertAdjacentHTML("BeforeEnd",document.all[Obj.id+"1"].innerHTML)
    eval(Obj.id+"t=1") }
    else
    {
    Obj.children[0].outerHTML=""
    eval(Obj.id+"t=0")}
   }
}
</script>

</head>
Раскрывающийся список с использованием Inner
<div id=i11 style="display:none"><ul><li>1.1<li>1.2<ul>  </div>
<div id=i21 style="display:none"><ul><li>2.1<li>2.2<ul>  </div>
<div id=i31 style="display:none"><ul><li>3.1<li>3.2<ul>  </div>

<ul onClick="qwe()">
<li id=i1>1
<li id=i2>2
<li id=i3>3
</ul>
```

```
</html>
```

Лекция 7. Модель DOM

7.1. Структура документа DOM

DOM – Document Object Model – программный интерфейс XML (и HTML) документов.

Уровни DOM

- Уровень 0. Включает в себя все специфические модели DOM, которые существовали до появления (см., например, лекцию 6) Необходимо обратить внимание, что эти модели формально не являются спецификациями DOM, опубликованными W3C, а скорее являются информацией о том, что существовало до начала процесса стандартизации.
- Уровень 1 (1998). Появление понятия узел.
- Уровень 2 (2000-2004). Поддержка пространства имён.
- Уровень 3 (с 2004).

DOM представляет документ как иерархию узлов (node). На вершине иерархии узел – document, который представляет собой весь документ. В качестве узлов представлено все содержимое документа: HTML-элементы, атрибуты и текст.

Для обсуждения свойств и методов будем использовать следующий фрагмент:

```
<div>
<ul id="components">
<li>HTML</li>
<li>CSS</li>
<li>JavaScript </li>
</ul>
</div>
```

7.2. Навигация по дереву документа

Навигацию можно начать с любого узла, у которого известен идентификатор.

```
var oList=document.getElementById("components")
```

Также в случае корректной страницы можно указать document.documentElement – самый верхний уровень.

Ссылка на родительский элемент:

var oParent=oList.parentNode

Дочерние элементы представляют собой коллекцию, на элемент которой ссылаются так:

var oItem1=oList.childNodes[1]

В данном случае это будет элемент CSS.

На первый и последний дочерний элемент ссылаются так: firstChild, lastChild.

Кроме этого можно сослаться на соседние узлы одного уровня:

var oPreviousSibling=oList.previousSibling

var oNextSibling=oList.nextSibling

7.3. Создание новых узлов

Методы: createElement(), createTextNode().

Рассмотрим, как добавить к нашему списку элемент <li> XML.

Первым делом создаем узел LI. Для этого даем команду

var oItem=document.createElement("LI")

Рекомендуется писать название тегов заглавными буквами.

Далее создаем содержимое узла:

var oText=document.createTextNode("XML")

Обращаю внимание на то, что узлы созданы, но еще не связаны ни между собой, ни с основным списком.

7.4. Редактирование дерева документов

Приведем некоторые методы для редактирования дерева документов.

Вставка: appendChild(), insertBefore().

Копирование: cloneNode().

Замещение: replaceChild().

Удаление: removeChild().

Присоединим узел oText к узлу oItem:

oItem.appendChild(oText)

Теперь в памяти имеется веточка XML. Добавим ее в конец узла oList:

oList.appendChild(oItem)

После последней команды в списке должен появиться элемент XML.

Для вставки в произвольное место можно использовать метод insertBefore(), например, так:

oList.firstChild.nextSibling.insertBefore(oItem)

Для копирования узла используется метод cloneNode(). Пример:

var oClone=oList.cloneNode(true)

Если установлен параметр true, то узел копируется со всеми потомками.

Далее скопированный узел можно добавить к списку:

oList.appendChild(oClone)

Замена последнего элемента:

```
var oItem=document.createElement("LI")
oItem.appendChild(document.createTextNode("JS"))
oList.replaceChild(oItem,oList.lastChild)
```

Пример удаления:

var oRemovedItem=oList.removedChild(oList.lastChild).

7.5. Работа с массивами элементов

Свойство getElementById позволяет обращаться к элементу, у которого известен идентификатор. В случае, когда надо работать с определенными тегами, полезным является свойство getElementsByTagName, которое возвращает все указанные теги в виде массива.

Пример. Выделение красным цветом каждой второй строки таблицы с использованием метода getElementsByTagName.

```
<body>
<table border="1">
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
```

```
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</body>
<script>
        a=document.getElementsByTagName("tr")
        n=a.length
        for(i=0;i<n;i=i+2)
        {a[i].style.color="red"}
</script>
```

Лекция 8. Библиотека jQuery

8.1. Введение в jQuery

jQuery – библиотека JavaScript, которая позволяет получать доступ к любому элементу модели DOM, обращаться к атрибутам и содержимому, а также манипулировать ими. Кроме этого библиотека предоставляет удобный API для работы с технологией AJAX.

Библиотека была создана в 2006 году. В настоящее время практически каждый сайт использует данную библиотеку.

Подключив библиотеку jQuery вместо десятков команд на JavaScript можно написать несколько команд. Команды основаны на селекторах и классах CSS.

Библиотеку jQuery можно скачать с сайта http://jquery.com/. Размер библиотеки в минимальном варианте составляет примерно 60 Кб.

Перед началом работы библиотеку надо подключить:

```
<script src="jquery.js" type="text/javascript"></script>
```

Вся работа с библиотекой ведется с использованием функции $. Общая идея использования: 1) выбирается элемент или группа элементов, 2) выполняются действия над выделенными элементами.

Пример. Выделение красным цветом каждой второй строки таблицы (сравните с аналогичным примером из лекции 7).

```
<script src="jquery.js" type="text/javascript"></script>
<style>
        .rrr{color:red}
</style>
```

```
<body>
<table border="1">
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
<script>
        $("table tr:even").addClass("rrr")
</script>
```

8.2. Обращение к элементам

Рассмотрим на примерах варианты обращения. Обращаю внимание, что все основано на синтаксисе CSS.

1. $(p a) выбирает все ссылки, расположенные в абзацах.

2. $(' #myId ') – выбор элемента с указанным идентификатором.

3. $('myClass') – использование классов.

4. $('body>div') – выбор элементов div, являющихся прямыми потомками элемента body.

5. $(p:even) – четные абзацы.

6. Использование ссылок.

- a[href^=http://] – этому селектору соответствуют все ссылки, которые начинаются с символов http://, на это указывает символ ^/
- a[href$=.pdf] – ссылки заканчиваются на «pdf»
- a[href*=ya.ru] – ссылка содержит упоминание ya.ru в произвольном месте.

7. Имеется возможность выбора определенных элементов при условии, что они содержат другие элементы (в примере ниже двоеточие как в математике означает “такое что”). Например, селектор

li:has(a)

выбирает элементы <li>, которые содержат элемент <a>.

8. Выбор по позиции (некоторые варианты)

:first – первое совпадение на странице;

:last – последнее совпадение на странице;

:first-child – первый дочерний элемент;

:last-child – последний дочерний элемент;

:nth-child(n) – n-й элемент;

:nth-child(even|odd) – четные или нечетные дочерние элементы;

:even – четные элементы;

:odd – нечетные элементы.

9. Выбор элементов на основе характеристик, не предусмотренных спецификацией CSS

:input – выбирает элементы формы;

:selected – выбранные элементы option;

:visible – выбор невидимых элементов.

8.3. Обработчик готовности документа

Первый способ – запуск обработчика, когда загрузилась страница

```
window.onload=function(){
        $("table tr:nth-child(even) ").addClass("even");
};
```

Недостаток данного метода заключается в том, что ничего не будет изменено, пока все не загрузится, включая картинки, т.е. пользователь может усмотреть как все изменяется, т.е. увидеть и исходный вариант, и преобразованный.

Второй способ – дождаться только загрузки структуры документа, а именно:

```
$(documen).ready(function(){
        $("table tr:nth-child(even) ").addClass("even");
});
```

Т.е. читаем буквально – “Запуск функции после загрузки”. У этой конструкции имеется сокращенная форма:

```
$(function(){
        $("table tr:nth-child(even) ").addClass("even");
});
```

8.4. Создание элементов DOM

Создание элемента

$(' Hi ')

Пустой элемент можно создать так:

$(' ')

Можно использовать кавычки, можно апострофы. Но наличие тегов обязательно. Задавать просто текст нельзя. Вспомните (лекция 7), сколько было команд, чтобы создать веточку элементов без jquery. Сама по себе эта команда ничего не выведет, надо эту веточку опять же привязать к родительскому элементу. Делается это так:

$(' Hi ').insertAfter('#a1')

Здесь a1 – это идентификатор объекта, после которого необходимо вставить элемент.

8.5. Работа с полученным набором значений

В терминах jQuery эти наборы называют обернутыми.

8.5.1. Определение размера. Для этого можно использовать свойство length, а можно метод size().

Пример. Замена содержимого с идентификатором a1 на количество элементов класса b2

$("#a1").html($(".b2").size())

8.5.2. Полученный набор значений рассматривается как массив, и с ним мы соответственно и работаем. Получить любой элемент в обернутом наборе можно по индексу. Например, получить второй элемент в наборе всех гиперссылок:

$('a')[1].

Вместо индексов можно использовать метод get(), т.е. последнюю команду можно записать так:

$('a').get(1).

При помощи метода get() можно получить обычный массив JavaScript, содержащий все обернутые элементы. Пример:

var allanchors=$('a').get().

8.5.3. Существуют специальные команды библиотеки, которые позволяют объединять различные полученные наборы, убирать определенные значения по некоторым правилам

8.6. Манипулирование объектами на странице

Команды jQuery позволяют манипулировать свойствами, атрибутами, стилями и содержанием элементов.

Для обращения к свойствам и их значениям используются методы JavaScript, нет методов непосредственно библиотеки.

Пример. Изменяет свойство title у всех элементов класса b2

```
$('.b2').each(function(n){this.title="New "+n})
```

Здесь использована команда **each(функция).** Она выполняет обход всех элементов в наборе и вызывает для них функцию. В качестве параметра функции передается индекс элемента в наборе.

8.7. Обработка событий

8.7.1. Различные браузеры по-своему могут обрабатывать события, jQuery пытается сгладить эти неприятности. Поэтому обращаемся только к методам jQuery, а библиотека уже сама смотрит, что за браузер и применяет то или иное свойство.

- Модель событий jQuery обладает следующими свойствами:
- поддерживает единый метод установки событий;
- позволяет устанавливать несколько обработчиков для события;
- использует стандартные названия типов событий;
- предоставляет единые методы отмены события и блокирования действий по умолчанию.

8.7.2. Подключение обработчиков. Рассмотрим пример функции, которая будет срабатывать при щелчке по любому рисунку:

```
$('img').blind('click',fuction(event){alert('Приветствую!');});
```

Для удаления обработчика используется команда unbind()

8.8. Скрытие и отображение объектов

8.1. Функции **hide**() и **show**(). С ними есть некоторые нюансы (например, show показывает изначальное состояние свойства display, поэтому при загрузке рекомендуют скрыть объект при помощи hide()), но в целом все как в примере.

Пример. $('#b3').hide()

А затем, скажем, по щелчку, $('#b3').show()

8.2. Имеются различные эффекты – раскрывающиеся списки, слайдеры, увеличение рисунков (когда-нибудь надо разобрать подробнее)

8.9. Создания слайдера

Рассмотрим пример создания слайдера – блока с контентом, выползающего по нажатию на конец закладки.

8.9.1. В текст вставляем следующий фрагмент:

```
<div id="container">
<div id="panel">
Содержимое панели
</div>
<p class="button"><a href="#" class="buttontext">Справка</a></p>
</div>
```

9.2. При помощи стилей изначально скрываем панель. Остальные параметры просто описательные – размер, цвет и т.д.

```
<!--Из-за наличия auto заодно и центрует в Mozila, т.к. auto  Указывает, что размер отступов будет автоматически рассчитан браузером.-->
<!--buttontext можно вообще не задавать-->
```

```
<style>
	.rrr	{color:red}
#container {
	margin: 0 auto;
	width: 152px;
	}
#panel {
	background: #1ca8f6;
	height: 230px;
	display: none;
	}
.button {
	width: 152px;
	height: 40px;
	border-top: #333 dotted 1px;
	text-align: center;
	}
.buttontext {
	font-weight: bold;
	font-size: 1.2em;
	text-shadow: 1px 1px 1px #666;
	}

</style>
```

9.3. Описываем функцию

```
<script type="text/javascript">
$(document).ready(function(){
$(".buttontext").click(function(){
$("#panel").slideToggle("normal"); return false;
});
});
</script>
```

Комментарии

- Команда **.slideToggle** позволяет чередовать два простых действия: показать и скрыть. normal – это скорость, означает, что панель выедет за 400 миллисекунд, есть еще fast, slow.
- Команда **return false** означает, что мы не даем перейти по ссылке, в качестве которой используется кнопка-закладка. То есть, она просто работает в качестве удобной ручки, за которую дергают, но не как непосредственно ссылка.

Лабораторные работы

Лабораторная работа № 1. Знакомство с HTML

Ознакомиться с работой следующих тегов и параметров: html, head, title, body (bgcolor, background, text, link), p (align), b, em, u, i, font, ol/ul, li, a.

Обратить внимание на то, что теги бывают парные и непарные, а также на то, что у многих тегов имеются параметры.

Задание 1.1. Создать в Блокноте следующую web-страничку (Название в title указать «Знакомство с HTML»):

Название страницы Эпиграф Кратко о себе несколько строчек. Описание должно содержать использование нескольких шрифтов, несколько слов выделить цветом. Страничка должна содержать все перечисленные теги с разными параметрами. Число Что-то вроде подписи в виде графического файла

Работа с таблицами, рисунками, гиперссылками

Для работы с таблицами используются теги table (border), tr (width, height), td/th (align, valign, width, colspan, rowspan)

Задание 1.2. Добавить к своей странице 2 таблицы с осмысленной информацией. Одна таблица — обычная, вторая должна содержать объединенные ячейки.

Для вставки рисунка используется тег img (alt, align, width, height, vspace, hspace, border).

Задание 1.3. Добавить к своей странице один рисунок три раза с разными размерами. Все используемые рисунки должны храниться в отдельной папке.

Гипертекстовые связи задаются при помощи тега a (href, target, name). Связи бывают трех видов: ссылки на ресурсы Internet, ссылка на сервис электронной почты, ссылки внутри документа.

Задание 1.4. Реализовать все три типа гиперссылок.

Лабораторная работа №2. Карты изображений и фреймы

В настоящее время фреймы на практике практически не используются. Задание дано в основном для закрепления навыков работы с тегами.

Задание 2.1. Создайте рисунок, содержащий расположенные в разных местах названия сайтов. Названия должны располагаться в геометрических фигурах — круге, прямоугольнике, четырехугольнике, пятиугольнике. Организуйте переход на заданные сайты, используя карты изображений.

Задание 2.2.

1. Создайте web-страничку, состоящую из 3-х фреймов, названных условно «left», «right», «bottom». Фрейм «left» должен содержать перечень факультетов университета, при щелчке по названию факультета в фрейме «right» должно появиться описание факультета. Во фрейм «bottom» поместить бегущую строку с приглашением поступать в университет. Бегущая строка должна быть разноцветной. Для выполнения задания используйте пример из лекции.
2. Создайте страничку, состоящую из двух горизонтальных фреймов. В верхний фрейм необходимо поместить рисунок (или фотографию) с несколькими явно выраженными объектами. При щелчке по каждому объекту в нижнем фрейме должно появиться описание указанного объекта.

Лабораторная работа №3. CSS

Работа посвящена отработке навыков по настройке web-страниц и каскадных таблиц. Для работы используйте материалы с лекции. Посмотрите, как определить вид тегов HTML с использованием CSS, также посмотрите, как подключать внешний стилевой файл.

Выполните следующие **задания.**

3.1. 2,5-мерный текст. Общая идея — три раза выводится один и тот же текст с небольшими смещениями, но с тремя разными цветами —

красным, темнокрасным, серебристым. Тогда темнокрасный цвет создает тень, а серебристый цвет — блики. Описания стилей должны содержаться в отдельном файле.

3.2. Создайте стилевой файл для HTML-документа из первого задания. Можно использовать как внешний стилевой файл, так и определение стилей внутри html-файла. По крайней мере опишите в стилевом файле следующие элементы для шрифта: название, размер, межстрочный интервал, размер красной строки. На основании созданного описания для эпиграфа специально задайте зеленый цвет и отсутствие красной строки.

Замечание. Название свойств CSS для выполнения этой части задания я на лекции не давал. Предполагается, что Вы самостоятельно найдете информацию о свойствах в Интернет. Цель этого — понять, что все свойства знать нельзя, но всегда можно найти.

3.3. Подключите статические фильтры для нескольких элементов на странице. Используйте Internet Explorer. Выведите рисунок тремя способами — обычный, перевернутый, инвертированный. Примените к некоторому тексту эффекты glow и shadow

Лабораторная работа № 4. Знакомство с JavaScript

Данная лабораторная работа предполагает знакомство с размещением кода JavaScript в html-документе, методами document.write, циклическими и условными конструкциями.

4.1. Постройте треугольник из линий. Для этого необходимо написать код

```
<script>
for(i=1;i<=10;i++)
document.write(«<hr width=\»"+i*10+»%\»>»)
</script>
```

4.2. Постройте треугольник из убывающих линий, т.е. самая длинная сверху, короткая внизу.

5*1= 5
5*2= 10
5*3= 15
5*4= 20
5*5= 25
5*6= 30
5*7= 35
5*8= 40
5*9= 45
5*10= 50

4.3. Постройте таблицу умножения для вводимого числа на числа от 1 до 10.

4.4. Модифицируйте предыдущий пример, взяв каждое значение в ячейку таблицы.

4.5. Создайте таблицу умножения, как показано на рисунке ниже.

1*1= 1	2*1= 2	3*1= 3	4*1= 4	5*1= 5	6*1= 6	7*1= 7	8*1= 8	9*1= 9	10*1= 10
1*2= 2	2*2= 4	3*2= 6	4*2= 8	5*2= 10	6*2= 12	7*2= 14	8*2= 16	9*2= 18	10*2= 20
1*3= 3	2*3= 6	3*3= 9	4*3= 12	5*3= 15	6*3= 18	7*3= 21	8*3= 24	9*3= 27	10*3= 30
1*4= 4	2*4= 8	3*4= 12	4*4= 16	5*4= 20	6*4= 24	7*4= 28	8*4= 32	9*4= 36	10*4= 40
1*5= 5	2*5= 10	3*5= 15	4*5= 20	5*5= 25	6*5= 30	7*5= 35	8*5= 40	9*5= 45	10*5= 50
1*6= 6	2*6= 12	3*6= 18	4*6= 24	5*6= 30	6*6= 36	7*6= 42	8*6= 48	9*6= 54	10*6= 60
1*7= 7	2*7= 14	3*7= 21	4*7= 28	5*7= 35	6*7= 42	7*7= 49	8*7= 56	9*7= 63	10*7= 70
1*8= 8	2*8= 16	3*8= 24	4*8= 32	5*8= 40	6*8= 48	7*8= 56	8*8= 64	9*8= 72	10*8= 80
1*9= 9	2*9= 18	3*9= 27	4*9= 36	5*9= 45	6*9= 54	7*9= 63	8*9= 72	9*9= 81	10*9= 90
1*10= 10	2*10= 20	3*10= 30	4*10= 40	5*10= 50	6*10= 60	7*10= 70	8*10= 80	9*10= 90	10*10= 100

4.6. Организуйте объединение двух ячеек по горизонтали и двух по вертикали, как это показано на рисунке. (Цель данной части задания исключительно учебная.)

1*1= 1	2*1= 2	3*1= 3	4*1= 4	5*1= 5	6*1= 6	7*1= 7	8*1= 8	9*1= 9	10*1= 10
1*2= 2	2*2= 4	3*2= 6	4*2= 8	5*2= 10	6*2= 12	7*2= 14	8*2= 16	9*2= 18	10*2= 20
1*3= 3	2*3= 6	3*3= 9	4*3= 12	5*3= 15	6*3= 18	7*3= 21	8*3= 24	9*3= 27	10*3= 30
1*4= 4	2*4= 8	3*4= 12	4*4= 16	5*4= 20	6*4= 24	7*4= 28	8*4= 32	9*4= 36	10*4= 40
1*5= 5	2*5= 10	3*5= 15	4*5= 20		6*5= 30	7*5= 35		9*5= 45	10*5= 50
1*6= 6	2*6= 12	3*6= 18	4*6= 24	5*6= 30	6*6= 36	7*6= 42	8*6= 48	9*6= 54	10*6= 60
1*7= 7	2*7= 14	3*7= 21	4*7= 28	5*7= 35	6*7= 42	7*7= 49	8*7= 56	9*7= 63	10*7= 70
1*8= 8	2*8= 16	3*8= 24	4*8= 32	5*8= 40	6*8= 48	7*8= 56	8*8= 64	9*8= 72	10*8= 80
1*9= 9	2*9= 18	3*9= 27	4*9= 36	5*9= 45	6*9= 54	7*9= 63	8*9= 72	9*9= 81	10*9= 90
1*10= 10	2*10= 20	3*10= 30	4*10= 40	5*10= 50	6*10= 60	7*10= 70	8*10= 80	9*10= 90	10*10= 100

Лабораторная работа № 5. Взаимодействие JavaScript и CSS

Общий принцип работы состоит в следующем. Имеется, например, объект с идентификатором id1. Тогда обращение к свойств стиля из JavaScript выполняется так:

id1.style.[конкретное свойство стиля]

При работе со свойствами css в Internet Explorer полезными являются свойства с приставкой pos, например, **posLeft**

posLeft задает левую позицию объекта, причем в тех же единицах измерения, что и свойство left. Это свойство читается/записывается.

Задания

5.1. Создайте скрипт, заставляющий текст бежать бесконечно направо.

Пример кода приведен на рисунке

```
<div id="id1" style="position:absolute;left:10px;
width:100px;height:100px;background:yellow">
Эксперимент</div>

<script>
function beg()
{
  id1.style.posLeft++
  setTimeout("beg()",100)
}
beg()
</script>
```

5.2. Текст должен бегать от левого края до правого и назад.

5.3. Этот же текст должен бегать по окружности.

Здесь можно использовать уравнение окружности в полярных координатах:

x=x0+Rcos(phi)

y=y0+Rsin(phi)

5.4. Движение по «сердцу». Можно использовать формулу:

x=x0+R(1-sin(phi))cos(phi)

y=y0+R(1-sin(phi))sin(phi)

а можно поискать в Интернет более точную формулу.

Лабораторная работа № 6. Работа с формами

6.1. Реализовать произвольную форму с обязательными полями. Например, форму регистрации. Через команду alert вывести значения формы. Использовать элементы text, textarea, checkbox, radio, select, button

6.2. Создать психологический тест по какой-нибудь теме. Для этого необходимо подобрать тест с числовыми значениями. При нажатии на «готово» функция должна проверить, все ли поля заполнены. Если заполнены все поля, то должен быть выдан результат.

Тесты с вопросами вида «Сколько будет 2 умножить на 3?», «Что больше 5 или 6?» не принимаются.

Лабораторная работа № 7. Калькулятор

7.1. Создайте калькулятор, выполняющий 4 арифметические операции и извлекающий корень. Два значения хранить в тестовом поле. Операции проводить при нажатии на кнопки. Результат записывать в третье текстовое поле.

7.2. Организуйте проверку в текстовых полях на предмет того, чтобы в вводились числа, используя функцию isNaN().

7.3. Добавить несколько методов объекта Math.

Лабораторная работа № 8. Работа с рисунками

8.1. Подобие gif-рисунков. Реализовать пример из темы «Array»

8.2. Реализовать чередование рисунков по циклу.

8.3. Изменение рисунков при наведении мыши. Для этого необходимо подготовить 2 рисунка, например, 1.png, 2.png. Затем набрать следующий код:

<img src=”1.png” name=”ris1” onMouseOver=”ris1.src=’2.png’” onMouseOut=”ris1.src=’1.png’” >

Здесь использованы обработчики двух событий: onMouseOver – происходит при наведении мыши, onMouseOut – при отводе мыши с рисунка.

Лабораторная работа № 9. Работа с датой и временем

9.1 Скрипт вычисляет количество секунд, которые пользователь был на странице. Для этого нужно описать две функции – первая срабатывает на со-

бытие onLoad и запоминает время, вторая – на событие onUnload и сравнивает время.

9.2. Загружает фон страницы в зависимости от текущей секунды.

9.3. В виде формы вводится дата и время будущего события, например, Новый год. Скрипт отсчитывает в отдельном текстовом поле, сколько дней, часов, минут, секунд осталось до указанной даты.

Подумайте над тем как поделить количество миллисекунд между двумя датами на дни, часы, минуты, секунды.

Лабораторная работа № 10. Строка

10.1. Заглавная буква пробегает в предложении, записанном в текстовом поле.

10.2. Используя тег , организуйте бегущую строку в произвольном месте страницы.

Лабораторная работа № 11. Фильтры

11.1. Имеется рисунок и текст. При помощи радиопереключателей к ним применяются различные статические фильтры. Рассмотреть поворот на определенный угол, отражение, увеличение, прозрачность. Реализовать возможность просмотра в различных браузерах.

11.2. Вставить рисунок. Рядом разместить выпадающий список с возможными размерами рисунка. При выборе определенного значения изменяетс размер рисунка.

11.3. Имеется рисунок. При щелчке по нему проявляется второй рисунок с каким-то динамическим фильтром. Повторный щелчок восстанавливает первый рисунок

Лабораторная работа № 12. Event

12.1. Изучение свойств мыши. Сравните координаты screenX, clientX, offsetX. Для изучения свойств сделайте форму, в текстовых полях которой выводятся соответствующие координаты при движении мышки.

12.2. Реализуйте движение некого объекта за курсором. При этом объект не должен быть «приклеен» к курсору, а двигаться к нему на небольшой скорости.

12.3. Реализация двух объектов, бегающих за курсором.

Лабораторная работа № 13. Пазлы

13.1. Подготовьте рисунок. Разделите его на 6 равных фрагментов.

13.2. Эти фрагменты должны появиться в произвольном месте страницы.

13.3. Реализовать возможность перетаскивания фрагментов на странице, а именно – при щелчке мышкой по фрагменту, он должен «приклеиться» к курсору, при повторном щелчке «отклеиться».

13.4. Добавить кнопку проверки правильности сбора рисунка.

Лабораторная работа № 14. Секретное окно

14.1. В течение 5 секунд щелкаем мышью по 3-м объектам в определенном порядке, после этого открывается определенная страница.

14.2. Набираем определенную фразу, она нигде не отображается, после этого открывается определенная страница. Использовать события onPress и свойства evet.keyCode для определения ASCII-кода.

Лабораторная работа № 15. DOM. Раскрывающееся меню

15.1. Реализовать пример из лекции, генерирующий список.

15.2. Создать раскрывающееся меню. При повторном щелчке меню должно сворачиваться.

Лабораторные работы № 16-17. jQuery

16.1. Реализуйте слайдер из лекции.

16.2. Найдите интересные примеры с использованием jQuery.

Лабораторные задания одним списком (сокращенные формулировки)

1. Знакомство с HTML

Создать web-страницу (желательно со смыслом), содержащую следующие элементы: рисунок, заголовки различного уровня, текст с использованием нескольких шрифтов, 3 вида гиперссылок, 2 таблицы — обычная и с объединенными полями.

2. Фреймы и карты изображений

2.1. Создайте web-страничку, состоящую из 3-х фреймов, названных условно «left», «right», «bottom». Фрейм «left» должен содержать перечень факультетов университета, при щелчке по названию факультета в фрейме «right» должно появиться описание факультета. Во фрейм «bottom» поместить бегущую строку с приглашением поступать в университет. Бегущая строка должна быть разноцветной. Для выполнения задания используйте пример из лекции.

2.2. Создайте страничку, состоящую из двух горизонтальных фреймов. В верхний фрейм необходимо поместить рисунок (или фотографию) с несколькими явно выраженными объектами. При щелчке по каждому объекту в нижнем фрейме должно появиться описание указанного объекта.

3. CSS

3.1. Выполнить задание «2,5-мерный текст».

3.2. Создайте стилевой файл для страницы из первого задания.

3.3. Создайте сайт со статическими фильтрами.

4. Знакомство с JavaScript

4.1. Выводится таблица умножения для введенного числа на числа от 1 до 10 (пример из лекции).

4.2. Самостоятельно составить таблицу Пифагора. Результат вывести в виде таблицы.

4.3. В полученной таблице объединить какие-нибудь ячейки по вертикали и горизонтали.

5. Взаимодействие JavaScript и CSS

5.1. Организовать движение текста слева-направо.

5.2. Организовать движение по окружности.

6. Работа с формами

6.1. Реализовать произвольную форму с обязательными полями. Через команду alert вывести значения формы. Использовать элементы text, textarea, checkbox, radio, select, button

6.2. Создать психологический тест по какой-нибудь теме.

7. Калькулятор

7.1. Создайте калькулятор, выполняющий 4 арифметические операции и извлекающий корень. Организуйте проверку в текстовых полях на предмет того, чтобы в вводились числа, используя функцию isNaN().

7.2. Добавить несколько методов объекта Math.

8. Работа с рисунками

8.1. Подобие gif-рисунков, т.е. реализовать чередование рисунков.

8.2. Изменение рисунков при наведении мыши.

9. Работа с объектом Date

9.1. Скрипт вычисляет количество секунд, которые пользователь был на странице.

9.2. Загружает фон страницы в зависимости от текущей секунды.

9.3. В виде формы вводится дата и время. Скрипт отсчитывает в отдельном текстовом поле, сколько дней, часов, минут, секунд осталось до указанной даты.

10. Работа с объектом String

Заглавная буква пробегает в предложении, записанном в текстовом поле.

11. Фильтры

11.1. Имеется рисунок и текст. При помощи радиопереключателей к ним применяются различные статические фильтры

11.2. Имеется рисунок. При щелчке по нему проявляется второй рисунок с каким-то динамическим фильтром. Повторный щелчок восстанавливает первый рисунок.

12. Знакомство с объектом Event

12.1. Изучение свойств мыши.

12.2. Реализация двух объектов, бегающих за курсором.

13. «Пазлы»

Реализовать перетаскивание мышью любого объекта на странице.

14. Секретное окно

14.1. В течение 5 секунд щелкаем мышью по 3-м объектам в определенном порядке, после этого открывается определенная страница

14.2. Набираем определенную фразу, она нигде не отображается, после этого открывается определенная страница.

Лабораторная работа № 15. DOM. Раскрывающееся меню

15.1. Реализовать пример из лекции, генерирующий список.

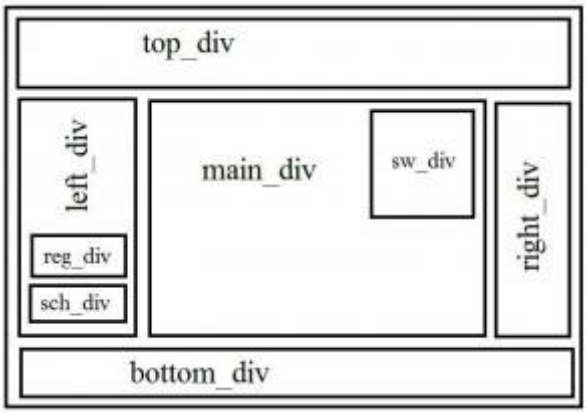
15.2. Создать раскрывающееся меню. При повторном щелчке меню должно сворачиваться.

Лабораторные работы № 16-17. jQuery

16.1. Реализуйте слайдер из лекции.

16.2. Найдите интересные примеры с использованием jQuery.

Контрольная работа



1. Каждый студент создает шаблон в соответствии с утвержденной преподавателем темой. Тема может быть совершенно любая – это и обзор сотовых телефонов, и описание Вашего любимого города. Как правило студент сам предлагает тему. Если студент не может предложить тему, то преподаватель назначает ее.
2. Большое внимание уделяется цветовой гамме. Текст должен быть читаемым.
3. Названия контейнеров должны соответствовать приведенным на рисунке. Расположение контейнеров может быть несколько иным, но все они должны присутствовать. Контейнер *sw_div* должен быть определен со свойством *float*.
4. Стилевой файл должен содержать хотя бы 100 определений.

5. Делать надо под определенное разрешение, но предусмотреть, чтобы при других разрешениях все смотрелось не плохо.
6. Попытаться реализовать под известные браузеры.
7. Страница должна содержать несколько рисунков, удовлетворяющих тематике сайта.
8. Быть готовым к тому, что работа будет отправлена на доработку.
9. Если контрольная работа не выполнена, то зачет получен не будет.
10. По дисциплине “Web-программирование” реализовать данную структуру также и при помощи таблиц без использования контейнеров.