Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Кузбасский государственный технический университет
имени Т.Ф. Горбачева»

А.А. Тайлакова

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ. РАЗРАБОТКА ВЕБ-СТРАНИЦ

Учебное пособие

Рекомендовано учебно-методической комиссией
специальности 230700.62 «Прикладная информатика»
в качестве электронного учебного пособия

Кемерово 2012

Рецензенты:

1. Прокопенко Евгения Викторовна, кандидат физико-математических наук, доцент кафедры прикладных информационных технологий.
2. Соколов Игорь Александрович, кандидат технических наук, доцент, заведующий кафедрой прикладных информационных технологий, председатель учебно-методической комиссии направления 230700.62 «Прикладная информатика».

Тайлакова Анна Александровна. Информационные системы и технологии. Разработка веб-страниц: учебное пособие [Электронный ресурс]: для студентов направления подготовки бакалавров 230700.62 «Прикладная информатика» / А. А. Тайлакова. – Электрон. дан. – Кемерово: КузГТУ, 2012. – 1 электрон. опт. диск (CD-ROM) ; зв. ; цв. ; 12 см. – Систем. требования : ОЗУ 64 Мб ; Windows XP/Vista/7 ; (CD-ROM-дисковод). – Загл. с экрана.

Предназначено для студентов, изучающих дисциплину «Информационные системы и технологии». В электронном учебном пособии изложены в систематизированном виде теоретические основы, обеспечивающие единую базу для изучения дисциплины.

ОГЛАВЛЕНИЕ

1. ТЕМА №1 ОСНОВНЫЕ ПРИНЦИПЫ РАЗРАБОТКИ ВЕБ-СТРАНИЦ

Для начала необходимо разобраться с основными понятиями веб-технологий: веб-сайт и веб-страница. Часто неопытные пользователи их неправомерно смешивают. Веб-страница – это минимальная логическая единица Всемирной паутины, которая представляет собой документ, однозначно идентифицируемый уникальным URL. Веб-сайт – это набор тематически связанных веб-страниц, находящихся на одном сервере и принадлежащий одному владельцу. В частном случае веб-сайт может быть представлен одной единственной веб-страницей. Всемирная паутина является совокупностью всех веб-сайтов.

Основой всей Всемирной паутины является язык разметки гипертекста HTML – Hyper Text Markup Language (Рисунок 1 □ ***Веб-технологии***). Он служит для логической (смысловой) разметки документа (веб-страницы). Иногда его неправомерно используют для управления способом отображения содержимого веб-страниц на экране монитора или при выводе на принтер, что в корне противоречит идеологии, принятой во всемирной паутине.

Для целей управления отображением содержимого веб-страниц предназначены каскадные таблицы стилей (CSS). CSS во многом сходны со стилями, применяемыми в популярном текстовом процессоре Word.

Идеология, подразумевающая использование логической разметки и стилей, является очень удобной, так как позволяет изменить оформление всего сайта путем изменения соответствующего стиля. В противном случае пришлось бы менять все теги, отображение содержимого которых требуется изменить.

Для придания веб-страницам динамизма (выпадающие меню, анимация) используются языки написания скриптов. Стандартным скриптовым языком во всемирной паутине является JavaScript. Ядром языка JavaScript является ECMAScript.

HTML, CSS, JavaScript – являются языками, с помощью которых можно создавать сколь угодно сложные веб-сайты. Но это всего лишь лингвистическое обеспечение, в то время как в браузерах документы представляются в виде набора объектов, множество типов которых является объектной моделью браузера (BOM). Объектная модель браузера уникальна для каждой модели и таким образом возникают проблемы

при создании межбраузерных приложений. Поэтому Веб-консорциум предложил объектную модель документа (DOM), являющуюся стандартным способом представления веб-страниц с помощью набора объектов.

Рисунок 1 □ Веб-технологии

В отличие от объектной модели браузера DOM содержит набор объектов лишь для содержимого документа и не имеет объектов, позволяющих управлять окнами и рамками окон. При написании приложений в целях поддержки межбраузерной переносимости необходимо придерживаться стандартов DOM, а к объектной модели браузера прибегать лишь при крайней необходимости. Такая необходимость может возникнуть, например, при управлении окнами и строкой состояния.

Следует отметить, что не все браузеры в полной мере поддерживают DOM, но, тем не менее, их последние версии обеспечивают такую поддержку в объеме, достаточном для

Как видно из вышесказанного, все веб-технологии тесно взаимосвязаны. Понимание этого факта позволит легче осознать назначение того или иного механизма, применяемого при создании веб-приложений [1].

Для эффективной работы не обойтись без необходимых и привычных инструментов, в том числе и при написании кода HTML. Поэтому для начальной разработки веб-страниц или даже небольшого сайта так называется набор страниц, связанных между собой ссылками и единым оформлением, понадобятся следующие программы:

1. текстовый редактор;
2. браузер для просмотра результатов;
3. валидатор – программа для проверки синтаксиса HTML и выявления ошибок в коде;
4. графический редактор;
5. справочник по тегам HTML.

HTML-документ можно создавать в любом текстовом редакторе, хоть «Блокноте», тем не менее, для этой цели подойдет не всякая программа. Нужна такая, чтобы поддерживала следующие возможности:

- подсветка синтаксиса выделение тегов, текста, ключевых слов и параметров разными цветами. Это облегчает поиск нужного элемента, ускоряет работу разработчика и снижает возникновение ошибок;
- работа с вкладками. Сайт представляет собой набор файлов, которые приходится править по отдельности, для чего нужен редактор, умеющий одновременно работать сразу с несколькими документами. При этом файлы удобно открывать в отдельных вкладках, чтобы быстро переходить к нужному документу;
- проверка текущего документа на ошибки.

Например:

- PSPad (http://www.pspad.com/ru/download.php),
- HtmlReader (http://manticora.ru/download.htm),
- Notepad++ (http://notepad-plus.sourceforge.net/ru/site.htm),
- EditPlus (http://www.editplus.com).

Браузер это программа, предназначенная для просмотра веб-страниц. Каждый браузер имеет свои уникальные особенности, поэтому для проверки универсальности кода требуется просматривать и корректировать код с их учетом. На сегодняшний день наибольшей популярностью пользуются Firefox, Google Chrome, Internet Explorer и Opera.

Графический редактор необходим для обработки изображений и их подготовки для публикации на веб-странице. Самой популярной программой такого рода является Photoshop, ставший стандартом для обработки фотографий и создания графических изображений для сайтов. Но в большинстве случаев мощь Photoshop-а избыточна, и лучше воспользоваться чем-нибудь более простым и проворным. В частности, программа Paint.Net позволяет сделать все необходимые манипуляции с изображениями, вдобавок бесплатна для использования[2].

ТЕМА №2 ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

HTML (HyperText Markup Language, язык разметки гипертекста) это система верстки, которая определяет, как и какие элементы должны располагаться на веб-странице.

2.1. Структура документа

Чтобы браузер при отображении документа понимал, что имеет дело не с простым текстом, а с элементом форматирования, применяются теги. Общий синтаксис написания тегов следующий.

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

Как видно из данного примера, теги бывают двух типов − одиночные и парные (контейнеры). Одиночный тег используется самостоятельно, а парный может включать внутри себя другие теги или текст. У тегов допустимы различные атрибуты, которые разделяются между собой пробелом. Впрочем, есть теги без всяких дополнительных атрибутов. Условно атрибуты можно подразделить на обязательные, они непременно должны присутствовать, и необязательные, их добавление зависит от цели применения тега.

Если открыть любую веб-страницу, то она будет содержать в себе типичные элементы, которые не меняются от вида и направленности сайта.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Пример веб-страницы</title>
 </head>
 <body>
  <h1>Заголовок</h1>
  <!-- Комментарий -->
  <p>Первый абзац.</p>
  <p>Второй абзац.</p>
 </body>
</html>
```

Тег <html> определяет начало HTML-файла, внутри него хранится заголовок (<head>) и тело документа (<body>).

Заголовок документа, как еще называют блок <head>, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением контейнера <title>.

Тело документа <body> предназначено для размещения тегов и содержательной части веб-страницы [2].

2.2. Основные элементы HTML

Заголовки выполняют важную функцию на веб-странице. Поисковые системы добавляют рейтинг тексту, если он находится внутри тега заголовка. Это важно для раскрутки сайта и для его занятия первых строк выдачи результата в поисковой системе по ключевым словам. Чем выше уровень заголовка, тем больше размер шрифта. Самым верхним уровнем является уровень 1 (<h1>), а самым нижним — уровень 6 (<h6>).

```
<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>
```

Параграф. Тег <p> определяет абзац (параграф) текста. Если закрывающего тега нет, считается, что конец абзаца совпадает с началом следующего блочного элемента.

```
<p>Второй абзац.</p>
```

Тег <p> является блочным элементом, поэтому текст всегда начинается с новой строки, абзацы идущие друг за другом разделяются между собой отбивкой (так называется пустое пространство между ними).

Маркированный список определяется тем, что перед каждым элементом списка добавляется небольшой маркер, обычно в виде закрашенного кружка. Сам список формируется с помощью контейнера <ul>, а каждый пункт списка начинается с тега <li>, как показано ниже.

```
<ul>
 <li>Первый пункт</li>
 <li>Второй пункт</li>
 <li>Третий пункт</li>
</ul>
```

Нумерованные списки представляют собой набор элементов с их порядковыми номерами. Вид и тип нумерации зависит от атрибутов те-

га <ol>, который и применяется для создания списка. Каждый пункт нумерованного списка обозначается тегом <li>.

```
<ol>
 <li>Первый пункт</li>
 <li>Второй пункт</li>
 <li>Третий пункт</li>
</ol>
```

Добавление **изображения** происходит в два этапа: вначале готовится графический файл желаемого размера, затем он добавляется на страницу через тег <img>. Сам HTML предназначен только для того, чтобы отобразить требуемую картинку, при этом никак ее не меняя.
При подготовке изображений следует учесть несколько моментов.

Поскольку веб-страница загружается по сети, существенным фактором выступает объем графического файла, встроенного в документ. Чем он меньше, тем быстрее отобразится изображение.

Размер картинки необходимо ограничить по ширине, например, установить не более 800 пикселов. Иначе изображение целиком не поместится в окне браузера, и появятся полосы прокрутки.

```
<img src="URL" alt="альтернативный текст">
```

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных, однако возможности таблиц этим не ограничиваются. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Для добавления таблицы на веб-страницу используется тег <table>. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются соответственно с помощью тегов <tr> и <td>. Таблица должна содержать хотя бы одну ячейку.

```
<table border="1" width="100%" cellpadding="5">
   <tr>
    <th>Ячейка 1</th>
    <th>Ячейка 2</th>
   </tr>
   <tr>
    <td>Ячейка 3</td>
    <td>Ячейка 4</td>
  </tr>
 </table>
```

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. Главное, чтобы к документу, на который делается ссылка, был доступ. Иными словами, если путь к файлу можно указать в адресной строке браузера, и файл при этом будет открыт, то на него можно сделать ссылку.

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. Оба действия выполняются с помощью тега <a>. Общий синтаксис создания ссылок следующий.

```
<a href="URL">текст ссылки</a>
```

Атрибут href определяет URL (Universal Resource Locator, универсальный указатель ресурса), иными словами, адрес документа, на который следует перейти, а содержимое контейнера <a> является ссылкой.

Адрес ссылки может быть как абсолютным, так и относительным. Абсолютные адреса должны начинаться с указания протокола (обычно http://) и содержать имя сайта.

```
<a href="http://htmlbook.ru">Изучение HTML</a>
```

Относительные ссылки ведут отсчет от корня сайта или текущего документа.

```
<a href="Ссылаемый документ.html">Ссылка</a>
<a href="../Ссылаемый документ.html">Ссылка</a>
<a href="Папка/Ссылаемый документ.html">Ссылка</a>
```

Ссылки могут вести на область внутри текущей страницы. Для создания таких ссылок необходимо добавить на страницу якорь. Якорем называется закладка с уникальным именем на определенном месте веб-страницы, предназначенная для создания перехода к ней по ссылке. Якоря удобно применять в документах большого объема, чтобы можно было быстро переходить к нужному разделу.

Для создания якоря следует вначале сделать закладку в соответствующем месте и дать ей имя при помощи атрибута name тега <a>. В качестве значения href для перехода к этому якорю используется имя закладки с символом решетки (#) впереди [2].

```
<body>
  <p><a name="top"></a></p>
```

```
  <p>...</p>
  <p><a href="#top">Наверх</a></p>
 </body>
```

Карта ссылок это изображение, разбитое на определенные зоны, каждая из которых представляет собой гиперссылку. После щелчка кнопкой мыши в пределах зоны браузер открывает страницу, отвечающую этой зоне.Прежде чем создавать карту ссылок, надо в графическом редакторе определить координаты крайних точек зон. Зоны могут быть трех типов:

- прямоугольная зона (rect). В этом случае задаются координаты двух точек левого верхнего угла, правого нижнего угла.
- многоугольник (poly). В этом случае задаются координаты каждой точки многоугольника.
- круг (circle). В этом случае задается координата центра круга и его радиус.

Для создания карты ссылок необходимо вставить нужное изображение с помощью дескриптора <img>, затем, при помощи атрибута USEMAP, присвоить ему имя. После этого можно приступить непосредственно к вводу данных при помощи тэга <map>.

Дескриптор <area> определяет зоны изображения карты ссылок.

Атрибут shape описывает форму зоны карты ссылок.

Атрибут coords назначает координаты конкретной зоны. Количество точек в этом атрибуте зависит от формы зон (прямоугольник, круг, многоугольник).

Атрибут href указывает страницу на которую ведет данная зона карты ссылок [3].

```
<img src="img/map.jpg" width="132" height="121" border="0"
usemap="#australia">

<map name="australia">

<area shape="rect" coords="0,0,68,60"
href="lsn022_1.html" target="_blank"
alt="Северо-восточная часть Австралии">

<area shape="rect" coords="68,0,132,60"
href="lsn022_2.html" target="_blank"
alt="Северо-западная часть Австралии">

<area shape="rect" coords="0,60,68,120"
href="lsn022_3.html" target="_blank"
alt="Юго-восточная часть Австралии">
```

```
<area shape="rect" coords="68,60,132,120"
href="lsn022_4.html" target="_blank"
alt="Юго-западная часть Австралии">

</map>
```

2.3. Элементы формы

Тег <form> устанавливает форму на веб-странице. Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

Документ может содержать любое количество форм, но одновременно на сервер может быть отправлена только одна форма. По этой причине данные форм должны быть независимы друг от друга.

```
<form action="URL">
  ...</form>
```

Тег <input> является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Главным образом <input> предназначен для создания текстовых полей, различных кнопок, переключателей и флажков. Хотя элемент <input> не требуется помещать внутрь контейнера <form>, определяющего форму, но если введенные пользователем данные должны быть отправлены на сервер, где их обрабатывает серверная программа, то указывать <form> обязательно. То же самое обстоит и в случае обработки данных с помощью клиентских приложений, например, скриптов на языке JavaScript.

Основной атрибут тега <input>, определяющий вид элемента — type. Он позволяет задавать следующие элементы формы: текстовое поле (text), поле с паролем (password), переключатель (radio), флажок (checkbox), скрытое поле (hidden), кнопка (button), кнопка для отправки формы (submit), кнопка для очистки формы (reset), поле для отправки файла (file) и кнопка с изображением (image). Для каждого элемента существует свой список атрибутов, которые определяют его вид и характеристики. Кроме того, в HTML5 добавлено еще более десятка новых элементов.

Тег <select> позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором, как показано далее. Конечный вид зависит от использования атрибута size тега <select>, который устанавливает высоту списка. Ширина списка определяется самым широким текстом, указанным в теге <option>, а также может изменяться с помощью стилей. Каждый пункт создается с помощью тега <option>, который должен быть вложен в контейнер <select>. Если планируется отправлять данные списка на сервер, то требуется поместить элемент <select> внутрь формы. Это также необходимо, когда к данным списка идет обращение через скрипты.

```
<select>
  <option>Пункт 1</option>
  <option>Пункт 2</option>
</select>
```

Поле <textarea> представляет собой элемент формы для создания области, в которую можно вводить несколько строк текста. В отличие от тега <input> в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер.

Между тегами <textarea> и </textarea> можно поместить любой текст, который будет отображаться внутри поля [2].

```
<textarea атрибуты>
  текст
</textarea>
```

2.4. Табличная верстка

Чтобы понимать, что можно ожидать от таблиц при вёрстке, следует знать их явные и неявные особенности, которые перечислены далее.

Вложенные таблицы. Одну таблицу допускается помещать внутрь ячейки другой таблицы. Это требуется для представления сложных данных или в том случае, когда одна таблица выступает в роли модульной сетки, а вторая, внутри нее, в роли обычной таблицы.

Размеры таблицы. Размеры таблицы изначально не устанавливаются и вычисляются на основе содержимого ячеек. В итоге суммарная ширина таблицы складывается из следующих параметров:

- ширина содержимого ячеек;
- толщина всех границ между ячеек;
- поля вокруг содержимого, устанавливаемые через атрибут cellpadding;
- расстояние между ячейками, которое определяется значением cellspacing.

Если для таблицы установлена её ширина в процентах или пикселах, то содержимое таблицы подстраивается под указанные размеры. Так, браузер автоматически добавляет переносы строк в текст, чтобы он полностью поместился в ячейку, и при этом ширина таблицы осталась без изменений. Бывает, что ширину содержимого ячейки невозможно изменить, как это, например, происходит с рисунками. В этом случае ширина таблицы увеличивается, несмотря на указанные размеры. Чтобы избежать указанной ситуации применяют несколько средств.

- Не добавляют в ячейку фиксированной ширины те изображения, размер которых превышает ширину ячейки. Способ, конечно, звучит банально, тем не менее, зная особенности ячеек, можно избежать неприятностей с их отображением.
- Для тега <table> используют стилевое свойство table-layout со значением fixed. Применение этого свойства позволяет обрезать рисунок, если он не помещается целиком в ячейку

Порядок ячеек. Основой таблицы выступает строка и ячейка, формирование таблицы происходит слева направо и сверху вниз (Рисунок 2 – Порядок создания ячеек).

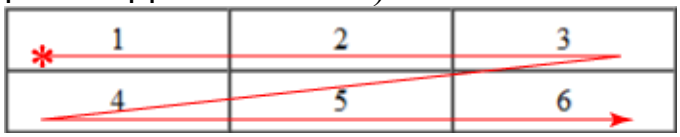


Рисунок 2 – Порядок создания ячеек

Неудобства этой схемы проявляются при активном использовании колонок и большом числе ячеек. Конечно, есть теги <col> и <colgroup>, но их возможности ограничены. Вставка новых ячеек или редактирование существующих может привести к ошибкам отображения таблицы.

Загрузка таблицы. Пока таблица не загрузится полностью, её содержимое не начнёт отображаться. Дело в том, что браузер, прежде чем показать содержимое таблицы, должен вычислить необходимые размеры ячеек, их ширину и высоту. А для этого нужно знать, что в этих

ячейках находится. Поэтому браузер и ожидает, пока загрузится все, что находится в ячейках, и только потом отображает таблицу.

Исходя из этого факта, таблицы не используют для хранения большой информации (от 100 Кб). А чтобы ускорить загрузку табличного макета, его разбивают на отдельные таблицы или используют свойство table-layout, применение которого позволяет несколько повысить скорость отображения содержимого таблицы. В обычной таблице браузер анализирует все ячейки и затем уже изменяет ширину колонок на основе этой информации. Включение table-layout со значением fixed меняет алгоритм расчета — браузер анализирует только первую строку и ширину колонок строит согласно ей. За счёт уменьшения числа вычислений и происходит выигрыш скорости отображения таблицы в целом [2].

ТЕМА №3 КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ CSS

Стилем или CSS (Cascading Style Sheets, каскадные таблицы стилей) называется набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид [2].

3.1 Способы внедрения CSS

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

```
<p style="font-size: 120%; font-family: monospace; color: #cd66cc">Пример текста</p>
```

При использовании **глобальных стилей** свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. Этот способ добавления стиля позволяет хранить стили в одном месте, в данном случае прямо на той же странице с помощью контейнера <style> [1].

```
<head>
  <title>Глобальные стили</title>
  <style type="text/css">
   H1 {
    font-size: 120%;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #333366;
   }
  </style>
 </head>
 <body>
  <h1>Hello, world!</h1>
 </body>
```

При использовании **связанных стилей** описание селекторов и их значений располагается в отдельном файле, как правило, с расширением css, а для связывания документа с этим файлом применяется тег <link>. Данный тег помещается в контейнер <head>[2].

```
<head>
  <title>Стили</title>
  <link rel="stylesheet" type="text/css" href="mysite.css">
  <link rel="stylesheet" type="text/css" href="http://www.htmlbook.ru/main.css">
 </head>
```

3.2 Селекторы

Основным понятием CSS выступает **селектор** − это некоторое имя стиля, для которого добавляются параметры форматирования [1]. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид:

```
Селектор { свойство1: значение; свойство2: значение; }
```

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

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

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега [2]. При использовании селекторов-классов синтаксис для будет следующий:

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

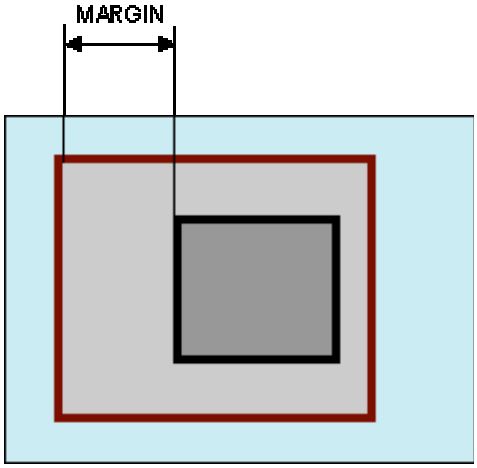
Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определенным классом, к тегу добавляется атрибут class="имя класса"

```
<html>
 <head>
    <title>Классы</title>
  <style type="text/css">
   .cite { /* Абзац с классом cite */
    color: navy; /* Цвет текста */
    margin-left: 20px; /* Отступ слева */
   }
  </style>
 </head>
 <body>
  <p class="cite">Для исключения засветки экрана дисплея световыми
потоками оконные проемы снабжены светорассеивающими шторами.</p>
 </body>
</html>
```

3.3 Основные свойства

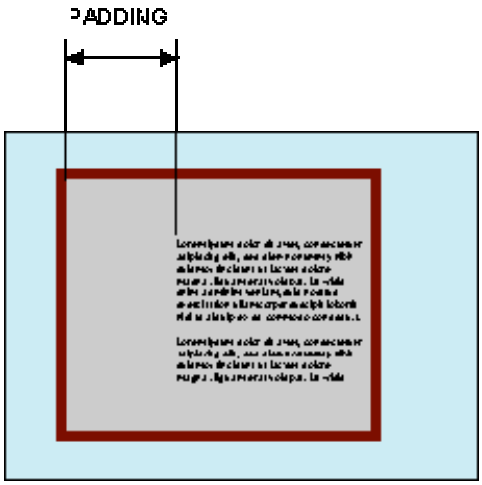
Внешние отступы. Свойство margin позволяет задать величину отступа сразу для всех сторон элемента или определить ее только для

указанных сторон. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента.




```
<style>
   body {
    margin: 0; /* Убираем отступы */
   }
   .parent {
    margin: 20%; /* Отступы вокруг элемента */
   }
   .child {
    margin: 10px; /* Отступы вокруг */
   }
  </style>
```

Внутренние отступы. Свойство padding позволяет задать величину поля сразу для всех сторон элемента или определить ее только для указанных сторон. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое.




```
<style>
   .layer {
    padding: 20px; /* Поля вокруг текста */
   }
  </style>
```

Свойства фона. Универсальное свойство background позволяет установить одновременно до пяти характеристик фона. Значения могут идти в любом порядке, браузер сам определит, какое из них соответствует нужному свойству. В CSS3 допустимо указывать параметры сразу нескольких фонов, перечисляя их через запятую.

```
background:[background-attachment || background-color || background-
image || background-position || background-repeat]
```

```
<style type="text/css">
   div {
        background: url(images/hand.png) repeat-y #fc0;
/* Цвет фона, путь к фоновому изображению и повторение фона по вер-
тикали */
   }
</style>
```

Свойство background-attachment устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента. Изображение может быть зафиксировано и оставаться неподвижным, либо перемещаться совместно с документом. В CSS3 можно указать несколько значений для ряда фоновых изображений, перечисляя значения через запятую.

fixed □ делает фоновое изображение элемента неподвижным;
scroll □ позволяет перемещаться фону вместе с содержимым;
local □фон фиксируется с учётом поведения элемента. Если элемент имеет прокрутку, то фон будет прокручиваться вместе с содержимым, но фон выходящий за рамки элемента остаётся на месте.
Свойство background-color устанавливает цвет фона элемента.

```
background-color: цвет | transparent | inheritОпределяет цвет фона
элемента.
```

Свойство background-image устанавливает фоновое изображение для элемента. Если одновременно для элемента задан цвет фона, он будет показан, пока фоновая картинка не загрузится полностью. То же произойдет, если изображения не доступны или их показ в браузере отключен. В случае наличия в рисунке прозрачных областей, через них будет проглядывать фоновый цвет. В CSS3 допустимо указывать несколько фоновых изображений, перечисляя их параметры через запятую.

```
background-image: url(путь к файлу) | none | inherit
```

Свойство background-repeat определяет, как будет повторяться фоновое изображение, установленное с помощью свойства background-image. Можно установить повторение рисунка только по горизонтали, по вертикали или в обе стороны. В CSS3 допустимо указывать несколько значений для каждого фона, перечисляя значения через запятую.

```
background-repeat: no-repeat | repeat | repeat-x | repeat-y
```

Устанавливает одно фоновое изображение в элементе без его повторений, положение которого определяется свойством background-position (по умолчанию в левом верхнем углу).

Свойства шрифта. Универсальное свойство font позволяет одновременно задать несколько характеристик шрифта и текста. В качестве обязательных значений свойства font указывается размер шрифта и его семейство. Остальные значения являются опциональными и задаются при желании.

```
font: [font-style||font-variant||font-weight] font-size font-family
```

Свойство font-family устанавливает семейство шрифта, которое будет использоваться для оформления текста содержимого. Список шрифтов может включать одно или несколько названий, разделенных запятой. Если в имени шрифта содержатся пробелы, например, Trebuchet MS, оно должно заключаться в одинарные или двойные кавычки.

```
font-family: имя шрифта [, имя шрифта[, ...]]
```

Свойство font-size определяет размер шрифта элемента. Размер может быть установлен несколькими способами. Набор констант (xx-small, x-small, small, medium, large, x-large, xx-large) задает размер, который называется абсолютным. Результат зависит от настроек браузера и операционной системы.

```
font-size: абсолютный размер | относительный размер | значение |
проценты | inherit
```

Свойство font-style определяет начертание шрифта — обычное, курсивное или наклонное. Когда для текста установлено курсивное или наклонное начертание, браузер обращается к системе для поиска подходящего шрифта. Если заданный шрифт не найден, браузер использует специальный алгоритм для имитации нужного вида текста. Результат и качество при этом могут получиться неудовлетворительными, особенно при печати документа.

```
font-style: normal | italic | oblique | inherit
```

Свойство font-variant определяет, как нужно представлять строчные буквы оставить их без модификаций или делать их все прописными уменьшенного размера. Такой способ изменения символов называется капителью.

```
font-variant: normal | small-caps | inherit
```

Свойство font-weight устанавливает насыщенность шрифта. Значение устанавливается от 100 до 900 с шагом 100. Сверхсветлое начертание, которое может отобразить браузер, имеет значение 100, а сверхжирное — 900. Нормальное начертание шрифта (которое установлено по умолчанию) эквивалентно 400, стандартный полужирный текст значению 700.

```
font-weight:
bold|bolder|lighter|normal|100|200|300|400|500|600|700|800|900
```

Свойства рамки. Универсальное свойство border позволяет одновременно установить толщину, стиль и цвет границы вокруг элемента. Значения могут идти в любом порядке, разделяясь пробелом, браузер сам определит, какое из них соответствует нужному свойству. Для установки границы только на определенных сторонах элемента, воспользуйтесь свойствами border-top, border-bottom, border-left, border-right.

```
border: [border-width || border-style || border-color] | inherit
```

Свойство border-width задает толщину границы одновременно на всех сторонах элемента или индивидуально для каждой стороны. Способ изменения толщины зависит от числа значений.

```
border-width: [значение | thin | medium | thick] {1,4} | inherit
```

Свойство border-color устанавливает цвет границы на разных сторонах элемента. Свойство позволяет задать цвет границы сразу для всех сторон элемента или только для указанных.

```
border-color: [цвет | transparent] {1,4} | inherit
```

Свойство border-radius устанавливает радиус скругления уголков рамки. Если рамка не задана, то скругление также происходит и с фоном.

```
border-radius: <радиус>{1,4} [ / <радиус>{1,4}]
```

Свойство box-shadow добавляет тень к элементу. Допускается использовать несколько теней, указывая их параметры через запятую, при наложении теней первая тень в списке будет выше, последняя ниже. Если для элемента задается радиус скругления через свойство border-radius, то тень также получится с закругленными уголками. Добавление тени

увеличивает ширину элемента, поэтому возможно появление горизонтальной полосы прокрутки в браузере.

```
box-shadow: none | <тень> [,<тень>]
где <тень>:
<сдвиг по x> <сдвиг по y> <радиус размытия> <растяжение> <цвет>
```

Позиционирование элементов. Свойство position устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице.

```
position: absolute | fixed | relative | static | inherit
```

absolute □ указывает, что элемент абсолютно позиционирован, при этом другие элементы отображаются на веб-странице словно абсолютно позиционированного элемента и нет. Положение элемента задается свойствами left, top, right и bottom, также на положение влияет значение свойства position родительского элемента. Так, если у родителя значение position установлено как static или родителя нет, то отсчет координат ведется от края окна браузера. Если у родителя значение position задано как fixed, relative или absolute, то отсчет координат ведется от края родительского элемента.

fixed □ по своему действию это значение близко к absolute, но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы. Браузер Firefox вообще не отображает полосы прокрутки, если положение элемента задано фиксированным, и оно не помещается целиком в окно браузера. В браузере Opera хотя и показываются полосы прокрутки, но они никак не влияют на позицию элемента.

relative □ положение элемента устанавливается относительно его исходного места. Добавление свойств left, top, right и bottom изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения.

static □ элементы отображаются как обычно. Использование свойств left, top, right и bottom не приводит к каким-либо результатам.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>position</title>
  <style>
   .layer1 {
    position: relative; /* Относительное позиционирование */
```

```
    background: #f0f0f0; /* Цвет фона */
    height: 200px; /* Высота блока */
   }
   .layer2 {
    position: absolute; /* Абсолютное позиционирование */
    bottom: 15px; /* Положение от нижнего края */
    right: 15px; /* Положение от правого края */
    line-height: 1px;
   }
  </style>
 </head>
 <body>
  <div class="layer1">
   <div class="layer2">
     <img src="images/girl.jpg" alt="Девочка" />
   </div>
  </div>
 </body>
</html>
```

Любые позиционированные элементы на веб-странице могут накладываться друг на друга в определенном порядке, имитируя тем самым третье измерение, перпендикулярное экрану. Каждый элемент может находиться как ниже, так и выше других объектов веб-страницы, их размещением по z-оси и управляет z-index. Это свойство работает только для элементов, у которых значение position задано как absolute, fixed или relative.

```
z-index: число | auto | inherit
```

Свойства текста. Свойство color определяет цвет текста элемента.

```
color: цвет | inherit
```

Свойство text-align определяет горизонтальное выравнивание текста в пределах элемента.

```
text-align: center | justify | left | right | inherit
```

Свойство text-shadow добавляет тень к тексту, а также устанавливает её параметры: цвет тени, смещение относительно надписи и радиус размытия.

```
text-shadow: none | тень [,тень]
где тень:
 <сдвиг по x> <сдвиг по y> <радиус размытия> <цвет>
```

Свойство text-indent устанавливает величину отступа первой строки блока текста (например, для абзаца <p>). Воздействия на все остальные строки не оказывается. Допускается отрицательное значение для созда-

ния выступа первой строки, но следует проверить, чтобы текст не выходил за пределы окна браузера [2].

```
text-indent: <значение> | <проценты> | inherit
```

3.4 Блочная верстка

Блочная верстка — это условное название метода верстки HTML–документов, при котором в качестве структурной основы для расположения текстовых и графических элементов документа используют тег . Главная особенность блочной верстки сайтов состоит в том, что в ней элементы могут быть расположены слоями. Это существенно расширяет визуальные возможности [4].

3.5 Динамические эффекты

Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице [2].

Стандарт CSS3 позволяет создавать анимацию, которая может заменить анимированные изображения, Flash-анимацию и скрипты JavaScript во многих случаях. Анимация − это эффект, который позволяет элементу плавно изменять один стиль на другой. Любое свойство CSS имеющее числовое представление можно анимировать.

Для создания анимации можно использовать свойства CSS3: transition, transform, animation; правило @keyframes [5].

ТЕМА №4 ЯЗЫК ПРОГРАММИРОВАНИЯ JAVASCRIPT

JavaScript − прототипно-ориентированный скриптовый язык программирования. Обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам [6].

4.1 Способы внедрения JavaScript

Тег <script>. Сценарии клиентского JavaScript представляют собой часть HTML-файла и находятся между тегами <script> и </script>. Количество инструкций JavaScript между этими тегами может быть любым; инструкции выполняются по порядку, как часть процесса загрузки документа. Теги <script> могут присутствовать как в разделе <head>, так и в разделе HTML-документа.

```
<html>
<head>
<title>Сегодняшняя дата</title><script language="JavaScript">
// Определяем функцию для дальнейшего использования
function print_todays_date() {
var d = new Date(); // Получаем текущую дату и время
document.write(d.toLocaleString()); // Вставляем это в документ
}
</script>
</head>

Дата и время:

<script language="JavaScript">
// Теперь вызываем функцию, определенную выше
print_todays_date();
</script>
</html>
```

Обработчики событий. Код JavaScript, расположенный в теге <script>, исполняется один раз, когда содержащий его HTML-файл читается в веб-браузер. Такие статические сценарии не могут динамически реагировать на действия пользователя. Более динамические программы определяют обработчики событий, автоматически вызываемые веб-браузером при возникновении определенных событий. Например, при нажатии кнопки формы.

События в клиентском JavaScript генерируются объектами HTML (например, кнопками), поэтому обработчики событий определяются как

атрибуты этих объектов. Например, чтобы определить обработчик события, который вызывается, когда пользователь щелкает по флажку в форме, код обработчика указывается в качестве атрибута HTML-тега, определяющего флажок:

```
<input type="checkbox" name="opts" value="ignore-
case"onclick="ignore-case = this.checked;">
```

Включение файлов JavaScript. В JavaScript тег <script> поддерживает атрибут src. Значение этого атрибута задает URL файла, содержащего код JavaScript.

```
<script src="../../javascript/util.js"></script>
```

Файл JavaScript обычно имеет расширение .js и содержит код JavaScript в чистом виде, без тегов <script> или любого другого HTML-кода [7].

4.2 Базовый синтаксис

В JavaScript все идентификаторы регистрозависимы; в названиях переменных можно использовать буквы, подчёркивание, символ доллара, арабские цифры; названия переменных не могут начинаться с цифры; для оформления однострочных комментариев используются //; многострочные и внутристрочные комментарии начинаются с /* и заканчиваются */ [6].

Типы данных JavaScript: целый (integer), вещественный (double), строковый (string), логический (boolean) и массив (array).
Переменная любого типа начинается с ключевого слова "var", и лишь значение переменной определяет тип переменной. Более того, тип переменной в JavaScript может быть изменён в любой точке программы [8].

```
var number = -323;
var str = "some string"
var bool = true;
var array = new Array(3, 7, 12, true, 4.5, "some string", true);
```

Область видимости переменной – это блок программы, в котором переменная определена. Глобальная переменная, объявленная вне какой бы то ни было функции, определена для всего сценария. Переменная, объявленная при помощи инструкции var в теле функции, является локальной, то есть определена только в теле функции. Ее значение доступно лишь при вызове функции.

Возможное совпадение идентификаторов глобальных и локальных переменных – синтаксически допустимая, но крайне неудобная при отладке и нежелательная в программном коде ситуация [9].

4.3 Основные конструкции

Условный оператор. Условный оператор if…else позволяет проверить определенное условие и, в зависимости от его истинности, выполнить ту или иную последовательность операторов. Условие — это любое выражение, значение которого может быть преобразовано к логическому типу.

```
if (условие) {
  //Блок операторов
}
else {
  //Блок операторов
}
```

Оператор множественного выбора. Оператор switch выполняет ту или иную последовательность операторов в зависимости от значения определенного выражения.

Выражение это любое выражение, значение — это возможное значение выражения, а операторы — любые группы операторов JavaScript. Оператор выбора сначала вычисляет значение выражения, а затем проверяет, нет ли этого значения в одной из меток case значение. Если такая метка есть, то выполняются операторы, следующие за ней; если нет, то выполняются операторы, следующие за меткой default (если она отсутствует, то управление передается оператору, следующему за switch). Необязательный оператор break указывает, что после выполнения операторов управление передается оператору, следующему за switch. Если break отсутствует, то после выполнения операторов начинают выполняться операторы, стоящие после следующей метки case.

```
switch(выражение) {
  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
  }
}
```

Цикл − это последовательность операторов, выполнение которой повторяется до тех пор, пока определенное условие не станет ложным. JavaScript содержит три оператора цикла: for, while и do…while, а также операторы break и continue, которые используются внутри циклов. Близок к операторам цикла и оператор итерации for…in, используемый при работе с объектами.

Цикл с заданным числом повторений. Оператор **for** выполняется следующим образом:

1. Выполняется выражение инициализация (обычно это выражение инициализирует счетчик или счетчики цикла).
2. Вычисляется значение условия. Если оно ложно, то управление передается блоку операторов.
3. Выполняется блок операторов.
4. Выполняется выражение изменение (обычно это выражение увеличивает или уменьшает счетчик или счетчики цикла) и управление передается этапу 2.

Данный оператор обычно используется в тех случаях, когда количество повторений цикла известно заранее.

```
for (инициализация; условие; изменение) {Блок операторов}
```

Оператор for…in выполняет заданные действия для каждого свойства объекта или для каждого элемента массива.

```
var foo=new Array ();

foo['foo1'] = 'Foo1';
foo['foo2'] = 'Foo2';

for (var i in foo) {
    alert(foo[i]);
}
```

Цикл с предусловием. Если блок операторов содержит более одного оператора, то он должен быть заключен в фигурные скобки {}. Оператор while выполняется следующим образом:

1. Вычисляется условие. Если оно ложно, то управление передается блоку операторов.
2. Выполняется блок операторов и управление передается этапу 1.

При использовании данного оператора необходимо убедиться, что рано или поздно условие станет ложным, т. к. иначе сценарий войдет в бесконечный цикл.

```
while (условие) {
  //Блок операторов
}
```

Цикл с постусловием. Оператор do … while выполняется следующим образом:

1. Выполняется оператор.
2. Вычисляется значение выражения условие. Если оно ложно, то управление передается оператору, следующему за данным оператором.
3. Управление передается этапу 1.
4. Этот оператор отличается от оператора while тем, что цикл обязательно выполняется хотя бы раз.

```
do {
  //Блок операторов
} while (условие)
```

Оператор прерывания. Оператор break позволяет досрочно выйти из цикла. Оператор break прерывает выполнение текущего цикла или оператора switch и передает управление оператору, следующему за прерванным. Этот оператор может употребляться только внутри циклов while, do…while, for или for…in, а также внутри оператора switch.

```
for (i = 0; i < 100; i++) {
  if (i == 50) break;
  document.write(i + " ");
}
```

Оператор перехода. Оператор continue завершает текущую итерацию текущего цикла или цикла, помеченного соответствующей меткой, и начинает новую итерацию. Этот оператор может употребляться только внутри циклов while, do…while, for или for…in [10].

```
for (i = 0; i < 100; i++) {
  if (i == 50) continue;
  document.write(i + " ");
}
```

4.4 Пользовательские функции

Функции являются одним из основных механизмов языка JavaScript; они охватывают ту область, которая в других языках программирования реализуется подпрограммами, процедурами и функ-

циями. Функция в JavaScript − это набор операторов, выполняющих определенную задачу. Для того, чтобы пользоваться функцией, мы должны сначала ее определить. Декларация функции имеет вид:

```
function имя(аргументы) {
  //Блок операторaторов
}
```

Здесь имя − идентификатор, задающий имя функции, аргументы − необязательный список идентификаторов, разделенных запятыми, который содержит имена формальных аргументов функции.

Функция может возвращать значение через свое имя, если внутри функции используется оператор return [10].

```
function cube(number) {
  return number * number * number;
}
```

Важной особенностью функций в JavaScript является то, что функция может вызываться с произвольным количеством аргументов вне зависимости от того, сколько их было указанно при её объявлении. Если количество аргументов, передаваемых функции при её вызове меньше, чем было указано при объявлении, то не переданные параметры будут равны undefined.

Передача в качестве аргумента функции значения *по ссылке* означает, что изменения, произведённые со значением аргумента в функции сохранятся после вызова функции. В этом случае в функцию передаётся ссылка и она, фактически, получает прямой доступ к области памяти, где хранится переменная.

Передача в качестве аргумента функции переменной *по значению* означает, что изменения ,произведённые со значением этой переменной не сохраняться после вызова функции. В этом случае в функцию передаётся копия переменной.

В JavaScript нельзя задать каким образом значение будет передаваться в функцию, но важно запомнить, что ссылочные типы данных (массивы и объекты) передаются по ссылке, а примитивные (числа, строки и логические значения) передаются по значению [11].

4.5 Объектная модель документа

Объектная модель документа не является частью языка JavaScript. DOM (Document Object Model) – это интерфейс прикладного программирования для представления документа (например, документа HTML ,

а также иных) и обеспечения доступа к его элементам и интерактивного изменения их свойств.

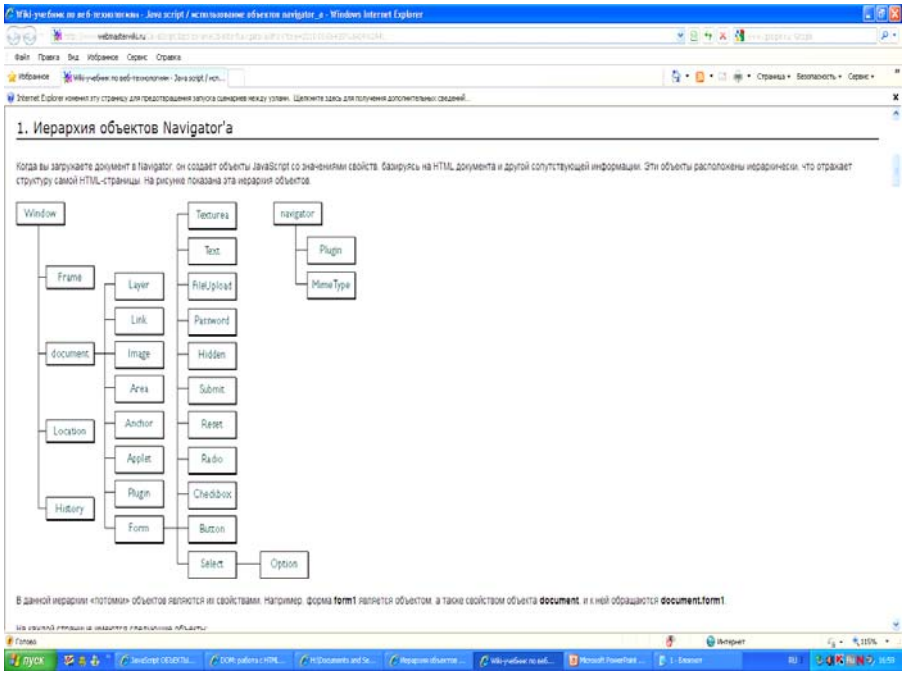


Рисунок 3 – Иерархия объектов DOM

Надо отметить, что разные браузеры, поддерживая рассматриваемую иерархию (Рисунок 3 ***– Иерархия объектов DOM***), предлагают и дополнительные свойства почти для каждого объекта [9,12].

Язык JavaScript позволяет легко управлять объектами веб-страницы, для этого важно понимать иерархию объектов, на которые опирается разметка HTML [13,14].

Получить доступ к полю e-mail (Рисунок 4 ***- Форма для ввода данных***, Рисунок 5 – ***Объектная модель документа***) при помощи DOM можно следующим способами:

```
document.myForm.email.value
document.forms[0].elements[1].value
document.forms[0].email.value
document.myForm.elements[1].value
```

Получение данных с формы:

```
var x= document.forms[0].elements[0].value;
var x = document.myForm.name.value;
alert(x);
```

Так же для доступа к элементам существуют специальные методы. getElementById(id) получает доступ к объекту, при помощи id, для этого у элемента обязательно должен присутствовать атрибут id.

```
document.getElementById('dataKeeper').value
```

Для доступа к элементам, у которых в спецификации предусмотрен атрибут name (form, input, a, select, textarea) существует свойство getElementsByName(name) .

Метод getElementsByTagName(tag) возвращает объект или массив объеков.

```
document.getElementsByTagName('LI')[1]
document.getElementsByTagName('DIV')[0].
getElementsByTagName('LI')
```

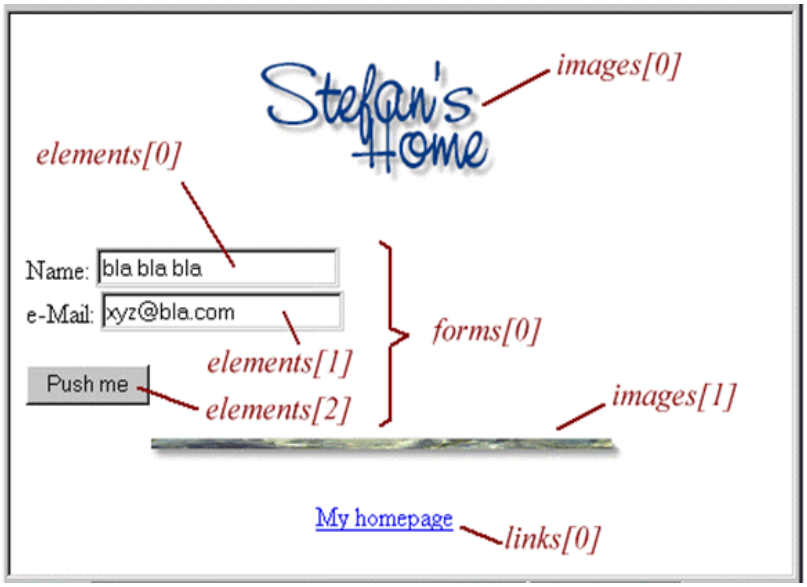


Рисунок 4 - Форма для ввода данных

Рисунок 5 – Объектная модель документа

Свойство для управления стилем – style.Оно аналогично установке стиля в CSS. Например, можно установить element.style.width:

```
element.style.width=100px
element.style.zIndex = 1000
```

Свойство innerHTML применяется, в основном, для динамического изменения содержания страницы. Оно содержит весь HTML-код внутри узла, и его можно менять [13,14].

```
document.getElementById('footer').innerHTML = '<h1>Bye!</h1>
```

4.7. События javascript

Практически все JavaScript-приложения выполняют те или иные действия, откликаясь на различные события. Событие − это сигнал от браузера о том, что что-то произошло.

Для того, чтобы скрипт реагировал на событие − нужно назначить хотя бы одну функцию-обработчик. Обычно обработчики называют "on+имя события", например:onclick. Обработчик события можно указать в виде inline-записи, прямо в атрибуте oncобытие. Например, для обработки события click на кнопке input, можно назначить обработчик onclick вот так:

```
<input id="b1" value="Нажми Меня" onclick="alert('Спасибо!');"
type="button"/>
```

Обработчики поддерживаются практически всеми HTML-элементами (*Таблица 1* ***- Обработчики событий***). Некоторые события можно имитировать с помощью соответствующих методов [13,15].

Таблица 1 - Обработчики событий

Обработчик события	Поддерживающие HTML-элементы	Описание
onBlur	A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA	Потеря текущим элементом фокуса, т.е. переход к другому элементу. Возникает при щелчке мышью вне элемента либо нажатии клавиши табуляции
onChange	INPUT, SELECT, TEXTAREA	Изменение значений элементов формы. Возникает после потерей элементом фокуса, т.е. после события blur
onClick	*Практически все HTML-элементы*	Одинарный щелчок (нажата и отпущена кнопка мыши)
onDblClick	*Практически все HTML-элементы*	Двойной щелчок
onFocus	A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA	Получение элементом фокуса (щелчок мышью на элементе или очередное нажатие клавиши табуляции)
onKeyPress	*Практически все HTML-элементы*	Нажата и отпущена клавиша на клавиатуре

ТЕМА №5 РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Регулярные выражения − это мощное средство для обработки входящих данных. Задача, требующая замены или поиска текста, может быть красиво решена с помощью этого "языка внутри языка". И хотя максимальный эффект от регулярных выражений можно добиться при использовании серверных языков, всё же не стоит недооценивать возможности этого приложения и на стороне клиента [16].

5.1 Синтаксис регулярных выражений

Классы символов. Вы можете определить класс, заключив символы в []. Класс будет совпадать с любым *одним* символом перечисленным в нем. Если первый символ класса (сразу после "[") - "^", то такой класс совпадает с любым символом *не перечисленным* в нем.

```
foob[aeiou]r
```
находит 'foobar', 'foober' и т.д., но не 'foobbr', 'foobcr' и т.д.;

```
foob[^aeiou]r
```
находит 'foobbr', 'foobcr' и т.д., но не 'foobar', 'foober' и т.д.

Внутри класса символ "-" может быть использован для определения диапазонов символов, например a-z представляет все символы между "a" и "z", включительно. Если Вам необходимо включить в класс сам символ "-", поместите его в начало или конец перечня или предварите '\'. Если Вам необходимо поместить в перечень сам символ ']', поместите его в самое начало или предварите '\'.

`[-az]` 'a', 'z' и '-';
`[az-]` 'a', 'z' и '-';
`[a\-z]` 'a', 'z' и '-';
`[a-z]` все 26 малых латинских букв от 'a' до 'z'.

Метасимволы □ это специальные символы, являющиеся важнейшим понятием в регулярных выражениях. Существует несколько групп метасимволов.

Метасимволы □ разделители строк:

^ начало строки;
$ конец строки;
. любой символ в строке.

```
^foobar
```

находит 'foobar' только если он в начале строки;

```
foobar$
```

находит 'foobar' только если он в конце строки;

```
^foobar$
```

находит 'foobar' только если это единственное слово в строке;

```
foob.r
```

находит 'foobar', 'foobbr', 'foob1r' и т.д.

Метасимволы стандартные классы символов

\w буквенно-цифровой символ или "_", то же, что и `[a-zA-Z_0-9]` ;
\W не \w, то же, что и `[^a-zA-Z_0-9]`;
\d цифровой символ, то же, что и `[0-9]`;
\D не \d, то же, что и `[^0-9]`;
\s любой "пробельный" символ;
\S не \s.

Стандартные классы \w, \d и \s можно использовать внутри классов символов.

```
foob\dr
```

находит 'foob1r', "foob6r' и т.д. но не 'foobar', 'foobbr' и т.д.;

```
foob[\w\s]r
```

находит 'foobar', 'foob r', 'foobbr' и т.д. но не 'foob1r', 'foob=r' и т.д.

Метасимволы – повторения. После любого элемента регулярного выражения может следовать очень важный тип метасимвола - повторитель. Используя их Вы можете определить число допустимых повторений предшествующего символа, метасимвола или подвыражения.

* ноль или более раз , то же что {0,};
\+ один или более раз, то же что {1,};
? ноль или один раз, то же что {0,1};
{n} точно n раз;
{n,} не менее n раз;
{n,m} не менее n но не более m раз;
{n,}? не менее n раз;
{n,m}? не менее n но не более m раз.

Если фигурные скобки встречаются в "неправильном" месте, где они не могут быть восприняты как повторитель, то они воспринимаются просто как символы.

```
foob.*r
```

находит 'foobar', 'foobalkjdflkj9r' и 'foobr'

```
foob.+r
```

находит 'foobar', 'foobalkjdflkj9r' но не 'foobr'

```
foob.?r
```

находит 'foobar', 'foobbr' и 'foobr' но не 'foobalkj9r'

```
fooba{2}r
```

находит 'foobaar'

```
fooba{2,}r
```

находит 'foobaar', 'foobaaar', 'foobaaaar' и т.д.

```
fooba{2,3}r
```

находит 'foobaar', или 'foobaaar' но не 'foobaaaar'

Метасимволы варианты. Вы можете определить перечень вариантов, используя метасимвол "|" для их разделения.

```
fee|fie|foe
```

найдет "fee" или "fie" или "foe", (так же как "f(e|i|o)e").

В качестве первого варианта воспринимается все от предыдущего метасимвола "(" или "[" или от начала выражения до первого метасимвола "|", в качестве последнего все от последнего "|" до конца выражения или до ближайшего метасимвола ")". Обычно, чтобы не запутаться, набор вариантов всегда заключают в скобки, даже если без этого можно было бы обойтись.

Обратите внимание, что метасимвол "|" воспринимается как обычный символ внутри перечней символов, например, `[fee|fie|foe]` означает ровно то же самое что и `[feio|]` [17].

```
foo(bar|foo)
```

находит 'foobar' или 'foofoo'.

5.2 Функции JavaScript для работы с регулярными выражениями

Создание регулярного выражения. Регулярное выражение можно создать двумя способами:

1. Используя литерал регулярного выражения

```
var re = /ab+c/
```

2. Вызывая функцию конструктор объекта RegExp

```
var re = new RegExp("ab+c")
```

Проверка соответствия заданной строки регулярному выражению. Чтобы просто проверить, подходит ли строка под регулярное выражение, используется метод test.

```
if ( /\s/.test("строка") ) {
...В строке есть пробелы!..}
```

Поиск и замена. Метод replace выполняет поиск совпадения в строке, и заменяет совпавшую подстроку другой подстрокой переданной как аргумент в этот метод. Метод replace не меняет строку, на которой вызван, а просто возвращает новую, измененную строку [18].

```
var re = /(\w+)\s(\w+)/;
var str = "John Smith";
var newstr = str.replace(re);
```

ТЕМА №6 БИБЛИОТЕКИ JAVASCRIPT

С увеличением популярности JavaScript, простота создания динамических элементов пользовательского интерфейса стала играть ключевую роль в веб-разработке. Этим обусловлен лавинообразный характер появления различных библиотек JavaScript

Практически все библиотеки JavaScript выпускаются под лицензиями копицентр и копилефт, чтобы обеспечить свободное от лицензионных отчислений разработку, использование и модификацию.

jQuery − библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API по работе с AJAX.

MooTools — это свободный JavaScript-фреймворк для разработки кроссбраузерных dвеб-приложений и веб-сервисов. MooTools является модульным, объектно-ориентированным фреймворком, созданным для помощи разработчикам JavaScript. MooTools совместим и протестирован с браузерами:Safari 2+,Internet Explorer 6+, Firefox2+ (и другими, основанными на движке Gecko),Opera 9+.

Prototype – JavaScript фреймворк, упрощающий работу с Ajax и некоторыми другими функциями. В Prototype присутствуют самые разные способы упрощения создания JavaScript приложений, от сокращённого вызова некоторых функций языка до сложных методов обращения к XMLHttpRequest.

Dojo (доджо) — свободная модульная библиотека JavaScript. Разработана с целью упростить ускоренную разработку основанных на JavaScript или AJAX приложений и сайтов. Разработка библиотеки была начата Алексом Русселом в 2004 году [6].

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Технологии создания веб-страниц.
2. Подготовка изображений. Форматы файлов.
3. Подготовка изображений. Табличная верстка средствами AdobePhotoshop.
4. Подготовка изображений. Блочная верстка средствами AdobePhotoshop (инструменты).
5. Основные принципы верстки веб-страниц. Табличная верстка.
6. Основные принципы верстки веб-страниц. Блочная верстка.
7. Основные принципы верстки веб-страниц. Типовые макеты (по ширине).
8. Основные принципы верстки веб-страниц. Типовые макеты (по колонкам).
9. Язык гипертекстовой разметки HTML. Структура документа (сравнить структуру документа HTML4 и HTML5)
10. Язык гипертекстовой разметки HTML.Параграфы, списки, заголовки, изображения.
11. Язык гипертекстовой разметки HTML.Ссылки. Карта ссылок.
12. Язык гипертекстовой разметки HTML.Таблицы.
13. Язык гипертекстовой разметки HTML.Элементы формы HTML4.
14. Язык гипертекстовой разметки HTML.Семантическая разметка HTML5.
15. Язык гипертекстовой разметки HTML.Элементы формы HTML5.
16. Язык гипертекстовой разметки HTML.Добавление видео и звука на страницу(HTML5).
17. Язык гипертекстовой разметки HTML.Элемент SVG и Canvas (область применения).
18. Каскадные таблицы стилей CSS. Способы внедрения в HTML файл.
19. Каскадные таблицы стилей CSS. Селекторы.
20. Каскадные таблицы стилей CSS. Псевдоклассы.
21. Каскадные таблицы стилей CSS. Внутренние и внешние отступы.
22. Каскадные таблицы стилей CSS. Форматирование текста.
23. Каскадные таблицы стилей CSS. Позиционирование, свойство z-index.
24. Каскадные таблицы стилей CSS. Свойства фона. Работа с цветом (способы задания значения цвета).

25. Каскадные таблицы стилей CSS. CSS3 тень объекта, скругленные углы.
26. Каскадные таблицы стилей CSS. Анимация CSS3.
27. Язык программирования JavaScript. Объектная модель документа.
28. Язык программирования JavaScript. Доступ к объектам страницы.
29. Язык программирования JavaScript. Свойства объектов.
30. Язык программирования JavaScript. События.
31. Язык программирования JavaScript. Основные конструкции. Типы данных
32. Язык программирования JavaScript. Регулярные выражения
33. Библиотеки JavaScript. Применение.

ЛИТЕРАТУРА

1. Материалы сайта «Structuralist» [Электронный ресурс]. – Режим доступа: http://www.structuralist.narod.ru
2. Материалы сайта «htmlbook.ru» [Электронный ресурс]. – Режим доступа: http://htmlbook.ru/
3. Материалы сайта «Спецификация HTML 4.01» [Электронный ресурс]. – Режим доступа: http://www.umade.ru/resources
4. Материалы сайта «Создание и продвижение сайтов» [Электронный ресурс]. – Режим доступа: http://myrusakov.ru/javascript-tipy-peremennyh.html.
5. Материалы сайта «Уроки CSS» [Электронный ресурс] – Режим доступа: http://www.wisdomweb.ru/http://uroki-css.ru/
6. Материалы сайта «Википедия» [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/JavaScript
7. Материалы сайта «Портал» [Электронный ресурс]. – Режим доступа: http://oooportal.ru/?cat=article&id=997
8. Материалы сайта «Создание и продвижение сайтов» [Электронный ресурс]. – Режим доступа: http://myrusakov.ru/javascript-tipy-peremennyh.html.
9. Материалы сайта «Технологии создания школьных Интернет-ресурсов» [Электронный ресурс]. – Режим доступа: http://vvz.nw.ru/Lessons/JavaScript/scope.htm
10. Материалы сайта «JavaScript» [Электронный ресурс]. – Режим доступа: http://www.j-s.ru/
11. Материалы сайта «Тру-кодер.ru» [Электронный ресурс]. – Режим доступа:true-coder.ru/
12. Материалы сайта «Diera.ru» [Электронный ресурс]. – Режим доступа:http://www.diera.ru
13. Материалы сайта «Центральный JavaScript-ресурс» [Электронный ресурс]. – Режим доступа:http://javascript.ru/
14. Материалы сайта «Веб-библиотека» [Электронный ресурс]. – Режим доступа:http://www.weblibrary.biz/javascript/document/hierarchy
15. Материалы сайта «Технологии создания школьных Интернет-ресурсов» [Электронный ресурс]. – Режим доступа: http://vvz.nw.ru/Lessons/JavaScript/scope.htm

16. Материалы сайта «PHP.SU» [Электронный ресурс]. – Режим доступа: http://php.su/articles/?cat=regexp&page=010

17. Материалы сайта «RegExp» [Электронный ресурс]. – Режим доступа: http://regexpstudio.com/ru/TRegExpr/Help/regexp_syntax.html

18. Материалы сайта «Mozilla Developer Network. It’s the web. You drive» [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/JavaScript/Guide/Regular_Expressions.