ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

Ташкентский университет информационных технологий
Факультет "Информационные технологии"

КУРСОВАЯ РАБОТА

По предмету "Системное программное обеспечение"
ТЕМА: Разработка программного обеспечения для сети Интернет.

Выполнил: студент группы 225-10 ИТр

Аршаниц С.А.

Ташкент – 2013 г.

Аннотация

В курсовой работе описан язык JavaScript, предназначенный для создания активных страниц сервера Web, а также классы JavaScript. На конкретных примерах исходных текстов документов HTML рассмотрены различные способы применения этого языка в формах и фреймах. Показано, как с помощью сценария JavaScriptможно работать с графическими изображениями и анимацией.

В отдельных главах рассказано об организации взаимодействия программ JavaScript и аплетов Java, расположенных в документе HTML, а также о применении cookie для хранения данных посетителей сервера Web.

Содержание

1. Назначение и применение JavaScript, общие сведения

1.1 Введение

Гипертекстовая информационная система состоит из множества информационных узлов, множества гипертекстовых связей, определенных на этих узлах и инструментах манипулирования узлами и связями. Технология **World Wide Web** - это технология ведения гипертекстовых распределенных систем в **Internet**, и, следовательно, она должна соответствовать общему определению таких систем. Это означает, что все перечисленные выше компоненты гипертекстовой системы должны быть и в **Web.**

Web, как гипертекстовую систему, можно рассматривать с двух точек зрения. Во-первых, как совокупность отображаемых страниц, связанных гипертекстовыми переходами (ссылками - контейнер ANCHOR). Во-вторых, как множество элементарных информационных объектов, составляющих отображаемые страницы (текст, графика, мобильный код и т.п.). В последнем случае множество гипертекстовых переходов страницы - это такой же информационный фрагмент, как и встроенная в текст картинка.

При втором подходе гипертекстовая сеть определяется на множестве элементарных информационных объектов самими HTML-страницами, которые и играют роль гипертекстовых связей. Этот подход более продуктивен с точки зрения построения отображаемых страниц "на лету" из готовых компонентов.

При генерации страниц в Web возникает дилемма, связанная с архитектурой "клиент-сервер". Страницы можно генерировать как на стороне клиента, так и на стороне сервера. В 1995 году специалисты компании Netscape создали механизм управления страницами на клиентской стороне, разработав язык программирования **JavaScript.**

Таким образом, JavaScript - это язык управления сценариями просмотра гипертекстовых страниц Web на стороне клиента. Если быть более точным, то JavaScript - это не только язык программирования на стороне клиента. Liveware, прародитель JavaScript, является средством подстановок на стороне сервера Netscape. Однако наибольшую популярность *JavaScript* обеспечило программирование на стороне клиента.

Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит.

На практике это выражается в том, что можно, например, изменить цвет фона страницы или интегрированную в документ картинку, открыть новое окно или выдать предупреждение.

Название "JavaScript" является собственностью Netscape. Реализация языка, осуществленная разработчиками Microsoft, официально называется Jscript. Версии JScript совместимы (если быть совсем точным, то не до конца) с соответствующими версиями JavaScript, т.е. JavaScript является подмножеством языка JScript.

JavaScript стандартизован ECMA (European Computer Manufacturers Association - Ассоциация европейских производителей компьютеров). Соответствующие стандарты носят названия ECMA-262 и ISO-16262. Этими стандартами определяется язык ECMAScript, который примерно эквивалентен JavaScript 1.1. Отметим, что не все реализации JavaScript на сегодня полностью соответствуют стандарту ECMA. В рамках данного курса мы во всех случаях будем использовать название JavaScript.

1.2 Понятие объектной модели применительно к JavaScript

Для создания механизма управления страницами на клиентской стороне было предложено использовать объектную модель документа. Суть модели в том, что каждый HTML-контейнер - это объект, который характеризуется тройкой:

- свойства;
- методы;
- события.

Объектную модель можно представить как способ связи между страницами и браузером. Объектная модель - это представление объектов, методов, свойств и событий, которые присутствуют и происходят в программном обеспечении браузера, в виде, удобном для работы с ними кода HTML и исходного текста сценария на странице. Мы можем с ее помощью сообщать наши пожелания браузеру и далее - посетителю страницы. Браузер выполнит наши команды и соответственно изменит страницу на экране.

Объекты с одинаковым набором свойств, методов и событий объединяются в классы однотипных объектов. Классы - это описания возможных объектов. Сами объекты появляются только после загрузки документа браузером или как результат работы программы. Об этом нужно всегда помнить, чтобы не обратиться к объекту, которого нет.

1.2.1 Свойства

Многие HTML-контейнеры имеют атрибуты. Например, контейнер якоря <A ...>...</A> имеет атрибут ***HREF***, который превращает его в гипертекстовую ссылку:

```
<A HREF=intuit.htm>intuit</A>
```

Если рассматривать контейнер якоря <A ...>...</A> как объект, то атрибут ***HREF*** будет задавать свойство объекта "якорь". Программист может изменить значение атрибута и, следовательно, свойство объекта:

```
document.links[0].href="intuit.htm";
```

Не у всех атрибутов можно изменять значения. Например, высота и ширина графической картинки определяются по первой загруженной в момент отображения страницы картинке. Все последующие картинки, которые заменяют начальную, масштабируются до нее. Справедливости ради

следует заметить, что в Microsoft Internet Explorer размер картинки может меняться.

Для общности картины свойствами в JavaScript наделены объекты, которые не имеют аналогов в HTML-разметке. Например, среда исполнения, называемая объектом Navigator, или окно браузера, которое является вообще самым старшим объектом JavaScript.

1.2.2 Методы

В терминологии JavaScript методы объекта определяют функции изменения его свойств. Например, с объектом "документ" связаны методы ***open(), write(), close().*** Эти методы позволяют сгенерировать или изменить содержание документа. Приведем простой пример:

```
function hello()
{
  id=window.open("","example","width=400, height=150");
  id.focus(); id.document.open();
  id.document.write("<H1>Привет!</H1>");
  id.document.write("<HR><FORM>");
  id.document.write("<INPUT TYPE=button VALUE='Закрыть окно' ");
  id.document.write("onClick='window.opener.focus();
                              window.close();'>");
  id.document.close();
}
```

В этом примере метод ***open()*** открывает поток записи в документ, метод ***write()*** осуществляет эту запись, метод ***close()*** закрывает поток записи в документ. Все происходит так же, как и при записи в обычный файл. Если у окна есть поле статуса (обычно в нем отображается уровень загрузки документа), то при незакрытом потоке записи в документ в нем будет "метаться" прямоугольник продолжения записи, как это происходит при загрузке документа.

1.2.3 События

Кроме методов и свойств объекты характеризуются событиями. Собственно, суть программирования на JavaScript заключается в написании обработчиков этих событий. Например, с объектом типа ***button*** (контейнер ***INPUT*** типа ***button*** - "Кнопка") может происходить событие ***click***, т.е. пользователь может нажать на кнопку. Для этого атрибуты контейнера

INPUT расширены атрибутом обработки события ***click - onClick***. В качестве значения этого атрибута указывается программа обработки события, которую должен написать на JavaScript автор HTML-документа:

```
<INPUT TYPE=button VALUE="Нажать" onClick=
        "window.alert('Пожалуйста, нажмите еще раз');">
```

Обработчики событий указываются в тех контейнерах, с которыми эти события связаны. Например, контейнер ***BODY*** определяет свойства всего документа, поэтому обработчик события завершения загрузки всего документа указывается в этом контейнере как значение атрибута ***onLoad.***

Примечание. Строго говоря, каждый браузер, будь то Internet Explorer, Netscape Navigator или Opera, имеет свою объектную модель. Объектные модели разных браузеров (и даже разные версии одного) отличаются друг от друга, но имеют принципиально одинаковую структуру. Поэтому нет смысла останавливаться на каждой из них по отдельности. Мы будем рассматривать общий подход применительно ко всем браузерам, иногда, конечно, заостряя внимание на различиях между ними.

1.3 Размещение кода на HTML-странице

Главный вопрос любого начинающего программиста: "Как оформить программу и выполнить ее?". Попробуем на него ответить как можно проще, но при этом не забывая обо всех способах применения JavaScript-кода.

Во-первых, исполняет JavaScript-код браузер. В него встроен интерпретатор JavaScript. Следовательно, выполнение программы зависит от того, когда и как этот интерпретатор получает управление. Это, в свою очередь, зависит от функционального применения кода. В общем случае можно выделить четыре способа функционального применения JavaScript:

- гипертекстовая ссылка (схема URL);
- обработчик события (handler);
- подстановка (entity) (в Microsoft Internet Explorer реализована в версиях от 5.X и выше);
- вставка (контейнер SCRIPT).

В учебниках по JavaScript описание применения JavaScript обычно начинают с контейнера ***SCRIPT***. Но с точки зрения программирования это не совсем правильно, поскольку такой порядок не дает ответа на ключевой вопрос: как JavaScript-код получает управление? То есть каким образом вызывается и исполняется программа, написанная на JavaScript и размещенная в HTML-документе.

В зависимости от профессии автора HTML-страницы и уровня его знакомства с основами программирования возможны несколько вариантов начала освоения JavaScript. Если вы программист классического толка (C, Fortran, Pascal и т.п.), то проще всего начинать с программирования внутри тела документа, если вы привыкли программировать под Windows, то в этом случае начинайте с программирования обработчиков событий, если вы имеете только опыт HTML-разметки или давно не писали программ, то тогда лучше начать с программирования гипертекстовых переходов.

1.4 URL-схема JavaScript

Схема **URL** (Uniform Resource Locator) - это один из основных элементов Web-технологии. Каждый информационный ресурс в Web имеет свой уникальный URL. URL указывают в атрибуте ***HREF*** контейнера ***A***, в атрибте ***SRC*** контейнера ***IMG***, в атрибуте ***ACTION*** контейнера ***FORM*** и т.п. Все URL подразделяются на схемы доступа, которые зависят от протокола доступа к ресурсу, например, для доступа к FTP-архиву применяется схема ***ftp***, для доступа к Gopher-архиву - схема ***gopher***, для отправки электронной

почты - схема ***smtp***. Тип схемы определяется по первому компоненту URL: ***http://intuit.ru/directory/page.html*** .В данном случае URL начинается с ***http*** - это и есть определение схемы доступа (схема http).

Основной задачей языка программирования гипертекстовой системы является программирование гипертекстовых переходов. Это означает, что при выборе той или иной гипертекстовой ссылки вызывается программа реализации гипертекстового перехода. В Web-технологии стандартной программой является программа загрузки страницы. JavaScript позволяет поменять стандартную программу на программу пользователя. Для того чтобы отличить стандартный переход по протоколу HTTP от перехода, программируемого на JavaScript, разработчики языка ввели новую схему URL - JavaScript:

```
<A HREF="JavaScript:JavaScript_код">...</A>
<IMG SRC="JavaScript:JavaScript_код">
```

В данном случае текст "JavaScript_код" обозначает программы-обработчики на JavaScript, которые вызываются при выборе гипертекстовой ссылки в первом случае и при загрузке картинки - во втором. Например, при нажатии на гипертекстовую ссылку ***Внимание!!!*** можно получить окно предупреждения:

```
<A HREF="JavaScript:alert('Внимание!!!');"> Внимание!!!</A>
```

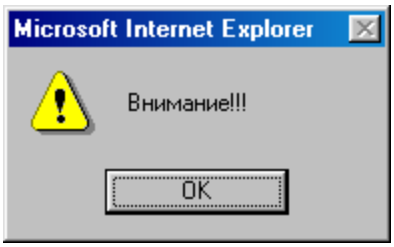

Рис. 1.1.

А при нажатии на кнопку типа ***submit*** в форме можно заполнить текстовое поле этой же формы:

```
<FORM NAME=f METHOD=post
    ACTION="JavaScript:window.document.f.i.VALUE=
                    'Нажали кнопку Click';void(0);">
<TABLE BORDER=0>
<TR>
<TD><INPUT NAME=i></TD>
<TD><INPUT TYPE=submit VALUE=Click></TD>
```

```
<TD><INPUT TYPE=reset VALUE=Reset></TD>
</TABLE>
</FORM>
```

В URL можно размещать сложные программы и вызовы функций. Следует только помнить, что схема JavaScript работает не во всех браузерах, а только в версиях Netscape Navigator и Internet Explorer, начиная с четвертой.

Таким образом, при программировании гипертекстового перехода интерпретатор получает управление после того, как пользователь "кликнул" по гипертекстовой ссылке.

1.5 Обработчики событий

Такие программы, как обработчики событий (***handler***), указываются в атрибутах контейнеров, с которыми эти события связаны. Например, при нажатии на кнопку происходит событие ***click***:

```
<FORM><INPUT TYPE=button VALUE="Кнопка" onclick=
                    "window.alert('intuit');"></FORM>
```

1.6 Подстановки

Подстановка (***entity***) встречается на Web-страницах довольно редко. Тем не менее это достаточно мощный инструмент генерации HTML-страницы на стороне браузера. Подстановки используются в качестве значений атрибутов HTML-контейнеров. Например, как значение по умолчанию поля формы, определяющего домашнюю страницу пользователя, будет указан URL текущей страницы:

```
<SCRIPT>
function l()
{
  str = window.location.href;
  return(str.length);
}
</SCRIPT>
<FORM><INPUT VALUE="&{window.location.href};" SIZE="&{l()};">
</FORM>
<SCRIPT>
<!-- Это комментарий ...JavaScript-код...// -->
```

```
</SCRIPT>
<BODY>
... Тело документа ...
</BODY>
</HTML>
```

HTML-комментарии здесь вставлены для защиты от интерпретации данного фрагмента страницы HTML-парсером в старых браузерах (у высокого начальства еще встречаются). В свою очередь, конец HTML-комментария защищен от интерпретации JavaScript-интерпретатором (// в начале строки). Кроме того, в качестве значения атрибута ***LANGUAGE*** у тега начала контейнера указано значение "JavaScript". VBScript, который является альтернативой JavaScript - это скорее экзотика, чем общепринятая практика, поэтому данный атрибут можно опустить - значение "JavaScript" принимается по умолчанию.

Очевидно, что размещать в заголовке документа генерацию текста страницы бессмысленно - он не будет отображен браузером. Поэтому в заголовок помещают декларации общих переменных и функций, которые будут затем использоваться в теле документа. При этом браузер Netscape Navigator более требовательный, чем Internet Explorer. Если не разместить описание функции в заголовке, то при ее вызове в теле документа можно получить сообщение о том, что данная функция не определена. Приведем пример размещения и использования функции:

```
<HTML>
<HEAD>
<SCRIPT>
function time_scroll()
{
  var d = new Date();
  window.status = d.getHours() +":"+d.getMinutes() +":"+
                                        d.getSeconds();
  setTimeout('time_scroll();',500);
}
</SCRIPT>
</HEAD>
<BODY onLoad=time_scroll()>
<CENTER>
<H1>Часы в строке статуса</H1>
```

В Internet Explorer 4.0 подстановки не поддерживаются, поэтому пользоваться ими следует аккуратно. Прежде чем выдать браузеру страницу с подстановками, нужно проверить тип этого браузера.

В случае подстановки интерпретатор получает управление в момент разбора браузером (компонент парсер) HTML-документа. Как только парсер встречает конструкцию &{..} у атрибута контейнера, он передает управление интерпретатору JavaScript, который, в свою очередь, после исполнения кода это управление возвращает парсеру. Таким образом данная операция аналогична подкачке графики на HTML-страницу.

2. Программируем свойства окна браузера

Класс объектов ***Window*** — это самый старший класс в иерархии объектов JavaScript. К нему относятся объект ***Window*** и объект ***Frame***. Объект ***Window*** ассоциируется с окном программы-браузера, а объект ***Frame*** — с окнами внутри окна браузера, которые порождаются последним при использовании автором HTML-страниц контейнеров ***FRAMESET*** и ***FRAME***.

При программировании на JavaScript чаще всего используют следующие свойства и методы объектов типа ***Window***:

Таблица 2.1.

Свойства	**Методы**	**События**
status	***open( )***	Событий нет
location	***close( )***	
history	***focus( )***	
navigator		

Объект ***Window*** создается только в момент открытия окна. Все остальные объекты, которые порождаются при загрузке страницы в окно, есть свойства объекта ***Window***. Таким образом, у ***Window*** могут быть разные свойства при загрузке разных страниц.

2.1 Поле статуса

Поле статуса — это первое, что начали использовать авторы HTML-страниц из арсенала JavaScript. Калькуляторы, игры, математические вычисления и другие элементы выглядели слишком искусственно. На их фоне бегущая строка в поле статуса была изюминкой, которая могла действительно привлечь внимание пользователей к Web-узлу. Постепенно ее популярность сошла на нет. Бегущие строки стали редкостью, но программирование поля статуса встречается на многих Web-узлах.

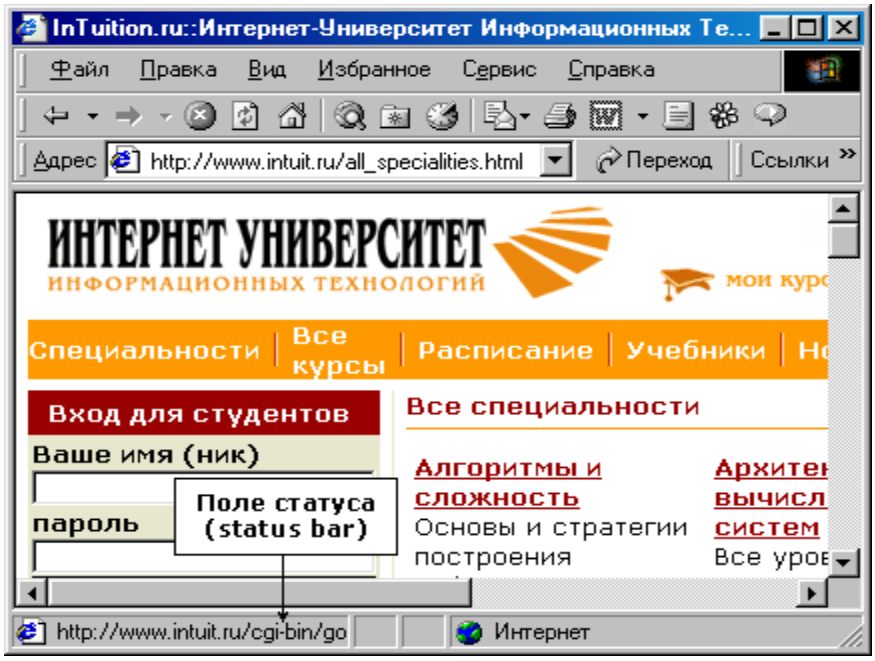

Рис. 2.1. Поле статуса

Полем статуса (***status bar***) называют среднее поле нижней части окна браузера сразу под областью отображения HTML-страницы. В поле статуса отображается информация о состоянии браузера (загрузка документа, загрузка графики, завершение загрузки, запуск апплета и т.п.). Программа на JavaScript имеет возможность работать с этим полем как с изменяемым свойством окна. При этом фактически с ним связаны два разных свойства:

- ***window.status***;
- ***window.defaultStatus.***

Разница между ними заключается в том, что браузер на самом деле имеет несколько состояний, связанных с некоторыми событиями. Состояние браузера отражается в сообщении в поле статуса. По большому счету, существует только два состояния: нет никаких событий (***defaultStatus***) и происходят какие-то события (***status***).

2.2 Программируем status

Свойство ***status*** связано с отображением сообщений о событиях, отличных от простой загрузки страницы. Например, когда курсор мыши проходит над гипертекстовой ссылкой, ***URL***, указанный в атрибуте ***HREF,*** отображается в поле статуса. При попадании курсора мыши на поле, свободное от ссылок, в поле статуса восстанавливается сообщение по умолчанию (***Document:Done***). Эта техника реализована на данной странице при переходе на описание свойств ***status*** и ***defaultStatus***:

```
<A HREF = #status onMouseover =
        "window.status='Jump to status description'; return true;"
onMouseout="window.status = 'Status bar programming';return true;">
                                window.status </A>
```

В документации по JavaScript указано, что обработчик событий ***mouseover*** и mouseout должен возвращать значение ***true***. Это нужно для того, чтобы браузер не выполнял действий, заданных по умолчанию. Практика показывает, что Netscape Navigator 4.0 прекрасно обходится и без возврата значения ***true***.

2.3 Программируем defaultStatus

Свойство ***defaultStatus*** определяет текст, отображаемый в поле статуса, когда никаких событий не происходит. В нашем документе мы определили его при загрузке документа:

```
<BODY onLoad="window.defaultStatus='Status bar programming';">
```

Это сообщение появляется в тот момент, когда загружены все компоненты страницы (текст, графика, апплеты и т.п.). Оно восстанавливается в строке статуса после возврата из любого события, которое может произойти при просмотре документа. Любопытно, что движение мыши по свободному от гипертекстовых ссылок полю страницы приводит к постоянному отображению ***defaultStatus***.

2.4 Поле location

В поле ***location*** отображается ***URL*** загруженного документа. Если пользователь хочет вручную перейти к какой-либо странице (набрать ее ***URL***), он делает это в поле ***location***. Поле располагается в верхней части окна браузера ниже панели инструментов, но выше панели личных предпочтений.

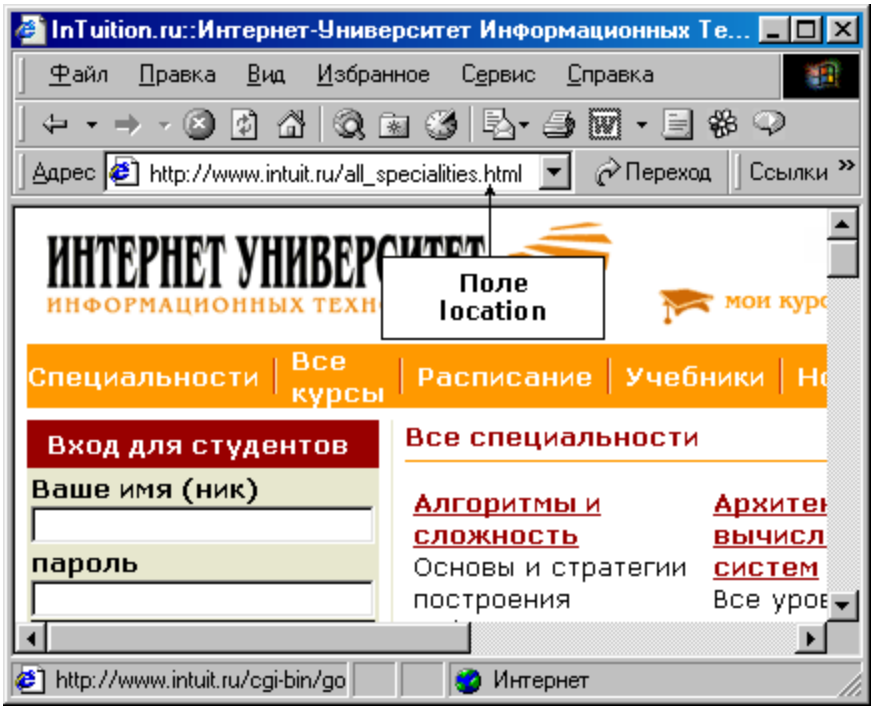

Рис. 2.2 Поле Location

Вообще говоря, ***Location*** — это объект. Из-за изменений в версиях JavaScript класс ***Location*** входит как подкласс и в класс ***Window***, и в класс ***Document***. Мы будем рассматривать ***Location*** только как ***window.location***. Кроме того, ***Location*** — это еще и подкласс класса ***URL***, к которому относятся также объекты классов ***Area*** и ***Link***. ***Location*** наследует все свойства ***URL***, что позволяет получить доступ к любой части схемы ***URL***.

Рассмотрим характеристики и способы использования объекта ***Location:***

- свойства;
- методы;
- событий, характеризующих ***Location***, нет.

Как мы видим, список характеристик объекта ***Location*** неполный.

2.4.1 Свойства

Предположим, что браузер отображает страницу, расположенную по адресу:

http://intuit.ru:80/r/dir/page?search#mark

Тогда свойства объекта Location примут следующие значения:

window.location.href = http://intuit.ru:80/r/dir/page?search#mark

```
window.location.protocol = http;
window.location.hostname = intuit.ru;
window.location.host = intuit.ru:80;
window.location.port = 80
window.location.pathname = /r/dir/;
window.location.search = search;
window.location.hash = mark;
```

2.4.2 Методы

Методы ***Location*** предназначены для управления загрузкой и перезагрузкой страницы. Это управление заключается в том, что можно либо перезагрузить документ (***reload***), либо загрузить (***replace***). При этом в историю просмотра страниц (***history***) информация не заносится:

```
window.location.reload(true);
window.location.replace('#top');
```

Метод ***reload( )*** полностью моделирует поведение браузера при нажатии на кнопку ***Reload*** в панели инструментов. Если вызывать метод без аргумента или указать его равным ***true***, то браузер проверит время последней модификации документа и загрузит его либо из кеша (если документ не был модифицирован), либо с сервера. Такое поведение соответствует простому нажатию на кнопку ***Reload***. Если в качестве аргумента указать ***false***, то браузер перезагрузит текущий документ с сервера, несмотря ни на что. Такое поведение соответствует одновременному нажатию на ***Reload*** и кнопку клавиатуры ***Shift (Reload+Shift).***

Метод ***replace( )*** позволяет заменить одну страницу другой таким образом, что это замещение не будет отражено в трассе просмотра HTML-страниц (***history***), и при нажатии на кнопку ***Back*** из панели инструментов пользователь всегда будет попадать на первую загруженную обычным способом (по гипертекстовой ссылке) страницу. Напомним, что при изменении свойств ***Location*** также происходит перезагрузка страниц, но в этом случае записи об их посещении в ***history*** пропадают.

2.4.3 История посещений (History)

История посещений (трасса) страниц World Wide Web позволяет пользователю вернуться к странице, которую он просматривал несколько минут (часов, дней) назад. История посещений в JavaScript трансформируется в объект класса history. Этот объект указывает на массив

URL-страниц, которые пользователь посещал и которые он может получить, выбрав из меню браузера режим ***GO***. Методы объекта ***history*** позволяют загружать страницы, используя ***URL*** из этого массива.

Чтобы не возникло проблем с безопасностью браузера, путешествовать по ***History*** можно, только используя индекс ***URL***. При этом ***URL***, как текстовая строка, программисту недоступен. Чаще всего этот объект используют в примерах или страницах, на которые могут быть ссылки из нескольких разных страниц, предполагая, что можно вернутся к странице, из которой пример будет загружен:

```
<FORM><INPUT TYPE=button VALUE="Назад"
on-Click=history.back()></FORM>
```

Данный код отображает кнопку "Назад", нажав на которую мы вернемся на предыдущую страницу.

3. Заключение

В этом проекте был рассмотрен язык разметки гипертекстовых документов HTML, его основные функции свойства и параметры. Сегодня применение HTML практикуется во всех без исключения электронных документах, независимо от тематики, величины и коммерческой направленности Интернет проекта.

В данной работе такие технологии, как CSS JavaScript, были затронуты лишь поверхностно дабы показать эффективность совокупности использования HTML с интерактивными скриптовыми технологиями. Использование последних является отдельной темой, подходящей для отдельного проекта.

4. Литература

1. J. Gosling, "The Java Language Environment", A white paper, Sun Microsystems, Mountain View, CA, 1995; http://java.sun.com
2. П. Храмцов, *Лабиринт Internet. Практическое руководство*. - М.: "ЭЛЕКТРОИНФОРМ", 1996.
3. "The Java Language Specification", technical report, Sun Microsystems, Mountain View, CA, 1995; http://java.sun.com
4. "The Java Virtual Machine Specification", technical report, , Sun Microsystems, Mountain View, CA, 1995; http://java.sun.com
5. П. Нотон, *Java. Справочное руководство (перевод с английского под общей редакцией А. Тихонова)*. - М.: Восточная Книжная Компания, 1996.
6. R. Rew, "The Java Generic Library (JGL)", Boulder Java Users Group, UCAR Unidata, September 1996,www.unidata.ucar.edu/staff/russ/java/jgl-bjug
7. Homepage of SJL [a Simple Java(tm) Library], Version 0.11, java.sun.com
8. D.R.Musser, A.Saini *STL Tutorial and Reference Guide. C++ Programming with the Standard Template Library*Addison-Wesley, 1996 (доступна также по www.aw.com/cp/musser-saini)
9. The HotJava Browser: A white paper, sunsite.math.klte.hu/hotjava/doc/overview/hotjava/index
10. G. McComb, *JavaScript Sourcebook*, Wiley Computer Publishing, 1996; доступна по Web: gmccomb.com/javascript/index3 см. также: JavaScript for Interactive Web Design, www.westlake.com/training/classes/outlines/js-outline