

## Оглавление

Введение .....	5
1. Обзор возможностей языка JavaScript .....	6
1.1. Общий обзор языка .....	7
Основные определения .....	7
Понятие объектной модели применительно к JavaScript.....	9
Размещение операторов языка JavaScript на странице.....	9
1.2. Язык ядра JavaScript.....	10
Синтаксис языка .....	10
Переменные и литералы в JavaScript .....	12
Выражения JavaScript .....	13
1.3. Управляющие конструкции языка JavaScript.....	14
Операторы JavaScript .....	14
Создание и вызов функций в JavaScript.....	17
1.4. Стандартные объекты и функции ядра JavaScript .....	18
Объект Array .....	18
Объект Date .....	19
Объект Math .....	20
Объект String.....	20
Стандартные функции верхнего уровня .....	20
1.5. Объекты клиента .....	21
Иерархия объектов .....	21
Объект navigator .....	22
Объект window.....	23
Объект document.....	25
Объект location.....	28
Объект form.....	29
1.6. Обработка событий .....	30
Атрибут onClick.....	31
Работа с меню .....	32
Управление логикой программного кода при помощи событий .....	32
Определение событий формы .....	33
Вставка звука .....	35
1.7. DHTML.....	35
Объединение JavaScript и CSS .....	36
1.8. Создание анимационных объектов.....	41
1.9. Слои .....	45
Позиционирование слоя .....	45
Свойство z-index .....	46
Свойства visibility и display .....	47
Динамическое управление слоями .....	47
Динамическое изменение цвета фона ячеек.....	49
2. Практика.....	51

Постановка задачи.....	51
2.1. Практическая работа №1. Размещение скриптов в HTML-документе.....	51
2.2. Практическая работа №2. Операторы управления, функции. Объекты ядра JavaScript.....	52
2.3. Практическая работа №3. Объекты клиентских приложений. Обработка событий. ....	55
2.4. Практическая работа №4. Объединение JavaScript и CSS.....	58
2.4. Практическая работа №5. Слои. Движущиеся элементы. ....	59
Литература .....	62

## **Введение**

В результате курса, проводимого под руководством преподавателя, студенты познакомятся с:

- технологиями и основными принципами объектно-ориентированного программирования;
- принципами создания динамических Web-документов;
- основными элементами языка;
- взаимосвязью языков скриптов и таблицей стилей для оформления Web-документов;
- организацией проверки данных введенных пользователем.

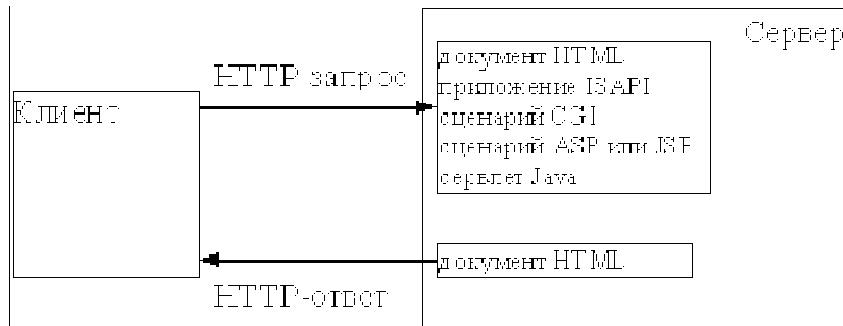
## **Цель курса**

По окончании данного курса студенты смогут:

- иметь представление об основах технологии объектно-ориентированного программирования, необходимых для Web-разработки;
- иметь представление о языке создания сценариев (то есть уметь понимать конструкции языка и интерпретировать результат);
- создавать Web-документы с динамически изменяемым содержимым;
- использовать стилевое форматирование совместно с языками сценариев для расширения возможностей оформления документов.

## 1. Обзор возможностей языка JavaScript

Взаимодействие клиента и сервера в Интернете осуществляется с помощью запросов, посылаемых клиентом серверу, и ответов сервера на запрос клиента:



Его основу составляют HTTP-сообщения, подразделяемые на:

- запрос (request) клиента к серверу;
- ответ (response) сервера клиенту.

Стандартный язык разметки HTML позволяет легко создавать статичные Web-страницы. Пользователь не может менять их содержимое, не может взаимодействовать с ними. Для того чтобы сделать страницу настоящему интерактивной, нужен еще один язык, выполняемый в контексте браузера, - скриптовый язык.

Исследования работы приложений интернета показали, что для выполнения определенных действий пользователя нет необходимости постоянно обращаться к серверу - эти действия можно реализовать на стороне клиента, если бы он позволял каким-то образом их запрограммировать. Так появился встроенный в программу просмотра Web-страниц (браузер) язык JavaScript, который расширил возможности языка разметки HTML, предоставляя разработчику возможность встраивать в документ HTML код программы, выполняющейся на клиенте.

Скриптовый язык используется для создания интерактивных страниц. Обычно он не содержит всех возможностей настоящих языков программирования, таких, например, как работа с файлами или управление графикой. Созданные с помощью скриптовых языков программы не могут выполняться самостоятельно - они работают только в контексте браузера, поддерживающего выполнения скриптовых программ. Создаваемые на скриптовых языках программы, называются сценариями или скриптами, включаются в состав Web-страниц и распознаются и обрабатываются браузером отдельно от остального HTML - кода.

Язык программирования JavaScript - объектно-ориентированный язык разработки встраиваемых приложений, выполняющихся как на стороне клиента, так и на стороне сервера.

Веб-обозреватель, работающий на компьютере-клиенте, обеспечивает среду, в которой JavaScript имеет доступ к объектам, которые представляют собой окна, меню, диалоги, текстовые области и т. д. Кроме того, обозреватель позволяет присоединить сценарии на языке JavaScript к

таким событиям, как загрузка и выгрузка страниц и графических образов, нажатие клавиш и движение мыши, выбор текста и пересылка форм. При этом программный код сценариев только реагирует на события и поэтому не нуждается в главной программе. Набор объектов, предоставляемых обозревателем, известен под названием Document Object Model (DOM).

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.

Основные области использования JavaScript при создании интерактивных HTML-страниц:

- Динамического создания содержимого страницы во время ее загрузки или уже после того, как она полностью загружена;
- Отображения диалоговых панелей и сообщений в статусной строке браузера;
- Оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер;
- Создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа (DHTML);

## 1.1. Общий обзор языка

### *Основные определения*

Любая программа оперирует некоторыми данными: именем стилевого класса, размерами элемента, цветом шрифта и прочие. JavaScript может манипулировать данными, относящимися к разным типам.

Тип данных описывает их возможные значения и набор применимых к ним операций. Типы данных бывают простыми и сложными. Сущность, относящаяся к простому типу данных может хранить только одно значение (это строковые, числовые и логические типы данных). Сущность сложного типа данных может хранить сразу несколько значений. Например – массивы. Другой пример сложного типа данных – объекты.

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

Существование программных объектов самих по себе не имеет никакого смысла. Они дадут преимущества при программировании тогда, когда можно организовать их взаимодействие.

1. Объекты – это сложные сущности, позволяющие хранить сразу несколько значений разных типов данных, они представляют собой блоки, из которых строится JavaScript. Применяются для возвращения значений и изменения состояния форм, страниц, браузера и определенных программистом переменных. Объекты можно сопоставить с существительными. Кошка, автомобиль, дом, компьютер, форма – все это существительные, они могут быть представлены как объекты.
2. Экземпляры объекта – сущности, хранящие реальные данные и созданные на основе этого объекта. То есть конкретный, реально существующий дом, находящийся по заданному адресу можно рассматривать, как экземпляр объекта типа дом.
3. Свойства – набор внутренних параметров объекта. Используются для того, чтобы различать экземпляры одного объекта – например, все экземпляры типа дом. Свойства сравнимы с прилагательными и ссылаются на уникальные для каждого экземпляра объекта особенности. Один и тот же объект может обладать многими свойствами: дом может быть большим и маленьким, синим и красным. Разные объекты могут обладать одинаковыми свойствами: дерево, так же, как и дом, может быть большим и маленьким, синим и красным... Большинство свойств объекта мы можем изменять, воздействуя на них через методы.
4. Методы - это действие или способ, при помощи которого мы можем изменять определенные свойства объекта, то есть управлять этими объектами, а также в некоторых случаях менять их содержимое.
5. События – это очень важное в программировании на JavaScript понятие. События главным образом порождаются пользователем, являются следствиями его действий. Если пользователь нажимает кнопку мыши, то происходит событие, которое называется Click. Если экранный указатель мыши движется по ссылке HTML-документа, происходит событие MouseOver. Существует несколько различных событий.
6. Оператор - это команда, инструкция для компьютера. Встретив в программе тот или иной оператор, машина четко его выполняет.
7. Функция - это определенная последовательность операторов, то есть набор команд, последовательное выполнение которых приводит к какому-то результату. Например, выполнение кем-то заданной Вами функции (процедуры) "возьми стакан, открой кран, набери в него воды и принеси мне" приведет к результату: Вы получите стакан воды из-под крана.
8. Переменная - в языках программирования переменные используются для хранения данных определенного типа, например параметров свойств объекта. Каждая переменная имеет свое имя (идентификатор) и хранит только одно значение, которое может

меняться в ходе выполнения программы. Данные могут быть разных типов: целое число, десятичная дробь, логическая константа, текстовая строка.

### *Понятие объектной модели применительно к JavaScript*

При загрузке HTML-страницы в браузер интерпретатор языка создает объекты со свойствами, определенными значениями тэгов страницы. Для правильного использования объектных моделей следует четко понимать, как браузер компонует страницы и, тем самым, создает иерархию объектов. При загрузке страницы просматриваются сверху вниз, тем самым последовательно происходит компоновка страницы и ее отображение в окне браузера. А это означает, что и объектная модель страницы также формируется последовательно, по мере ее обработки. Поэтому невозможно обратиться из сценария, расположенного ранее какой-либо формы на странице, к элементам этой формы. Всегда следует помнить о том, что браузер последовательно сверху вниз интерпретирует содержимое HTML-страницы.

Еще один аспект работы с объектами языков сценариев заключается в том, что нельзя изменить свойства объектов. Браузер обрабатывает страницу только один раз, компонуя и отображая ее. Поэтому попытка в сценарии изменить свойство отображеного элемента страницы, обречена на провал. Только повторная загрузка страницы приведет к желаемому результату.

### *Размещение операторов языка JavaScript на странице*

Встроить сценарий JavaScript в HTML-страницу можно несколькими способами.

1. Задание операторов языка внутри тэга `<script>` языка HTML.

Для внедрения в HTML-страницу сценария JavaScript в спецификацию языка HTML был введен тэг-контейнер `<script>...</script>`, внутри которого могут располагаться операторы языка JavaScript. Обычно браузеры, не поддерживающие какие-нибудь тэги HTML, просто их игнорируют, анализируя, однако, содержимое пропускаемых тэгов с точки зрения синтаксиса языка HTML, что может приводить к ошибкам при отображении страницы. Во избежание подобной ситуации следует помещать операторы языка JavaScript в контейнер комментария `<!-- ... -->`, как показано ниже

```
<script (language="javascript")>
<!--
операторы javascript
//-->
</script>
```

Параметр `language` задает используемый язык сценариев. В случае языка JavaScript его значение задавать не обязательно, так как этот язык используется браузерами по умолчанию.

Примечание:

символы `//` перед закрывающим тэгом комментария `-->` являются оператором комментария JavaScript. Он необходим для правильной работы интерпретатора.

Документ может содержать несколько тэгов `<script>`, расположенных в любом месте документа. Все они последовательно обрабатываются интерпретатором JavaScript по мере отображения частей документа в окне браузера. В связи с этим ссылка на переменную, определенную в сценарии, размещенном в конце документа, может привести к генерации ошибки интерпретатора при обращении к такой переменной из сценария в начале документа.

## 2. Задание файла с кодом JavaScript.

Тэг `<script>` имеет параметр `src`, позволяющий связать встраиваемый сценарий с внешним файлом, содержащим программный код на языке JavaScript. В качестве значения параметра задается полный или относительный URL-адрес ресурса. Задание закрывающего тэга `</script>` обязательно, независимо от того, заданы или нет операторы внутри тэга. Следующий фрагмент кода связывает документ HTML с файлом-источником, содержащим некоторый набор функций:

```
<script language="JavaScript" src="http://url/file.js">
    операторы javascript
</script>
```

Примечание:

связываемый внешний файл не должен содержать тэгов HTML и должен иметь расширение `.js`.

## 3. Использование выражений JavaScript в качестве значений параметров тэгов HTML.

Переменные и выражения JavaScript можно использовать в качестве значений параметров тэгов HTML. Например:

```
<a href="javascript: window.open('name.htm', '_self')">

</a>
```

## 4. Определение обработчика событий в тэге HTML.

## 1.2. Язык ядра JavaScript

### *Синтаксис языка*

Язык JavaScript чувствителен к регистру.

Приложение JavaScript представляет собой набор операторов языка (команд), последовательно обрабатываемых встроенным в браузер интерпретатором. Каждый оператор можно располагать в отдельной строке. В этом случае разделитель ‘;’, отделяющий один оператор от другого, не обязателен. Его используют только в случае задания нескольких операторов на одной строке. Любой оператор можно расположить в нескольких строках без всякого символа продолжения. Например, следующие два вызова функции alert эквивалентны:

```
...
alert("Подсказка");
alert(
"Подсказка"
);
...
```

Нельзя перемещать на другую строку единый строковый литерал - он должен располагаться полностью на одной строке текста программы или разбит на два строковых литерала, соединенных операцией конкатенации ‘+’:

```
...
alert("Подсказка");// правильно
alert("Под
сказка"); // не правильно
alert("Под" +
"сказка"); // правильно (но браузер выведет текст одной строкой!)
...
```

Пробельные символы в тексте программы являются незначащими, если только они не используются в строковых литералах.

В JavaScript строковые литералы можно задавать двумя равноправными способами - последовательность символов, заключенная в двойные или одинарные кавычки:

```
"Анна"
'Анна'
```

В строковых литералах можно использовать ESC-последовательности, которые начинаются с символа обратной наклонной черты, за которой следует обычный символ. Некоторые подобные комбинации трактуются как один специальный символ.

Таблица 1.

Esc-последовательности	Символ
\b	Возврат на один символ
\f	Переход на новую страницу
\n	Переход на новую строку
\r	Возврат каретки
\t	Горизонтальная табуляция Ctrl-I

\'	Апостроф
\\"	Двойные кавычки
\\"\\	Обратная наклонная черта

ESC-последовательности форматирования используются при отображении информации в диалоговых окнах, отображаемых функциями `alert()`, `prompt()` и `confirm()`, а также, если методом `document.write()` записывается содержимое элемента `pre`.

Комментарии в программе JavaScript двух видов - односторонние и многострочные:

```
// комментарий, расположенный на одной строке.  
/*  
   комментарий, расположенный  
   на нескольких строках.  
*/
```

Ссылка на объект осуществляется по имени, заданному параметром `name` тэга HTML, с использованием точечной нотации. Например, пусть в документе задана форма с двумя полями ввода:

```
<form name="form1">  
Фамилия: <input type = "text" name = "student" size = 20>  
Курс: <input type = "text" name = "course" size = 2>  
</form>
```

Для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript следует использовать ссылку `document.form.student.value`, а чтобы определить курс, на котором обучается студент, необходимо использовать ссылку `document.form.course.value`.

### *Переменные и литералы в JavaScript*

В JavaScript все переменные вводятся с помощью одного ключевого слова `var`. Синтаксическая конструкция для ввода в программе новой переменной с именем `name1` выглядит следующим образом:

```
var name1;
```

Объявленная таким образом переменная `name1` имеет значение `'undefined'` до тех пор, пока ей не будет присвоено какое-либо другое значение, которое можно присвоить и при ее объявлении:

```
var name1 = 5;  
var name1 = "новая строковая переменная";
```

JavaScript поддерживает четыре простых типа данных:

- Целый
- Вещественный
- Строковый
- Логический (булевый)

Для присваивания переменным значений основных типов применяются литералы – буквальные значения данных соответствующих типов.

### *Выражения JavaScript*

Выражение – комбинация переменных, литералов и операторов, в результате вычисления которой получается одно единственное значение. Переменные в выражениях должны быть инициализированы.

#### 1. Присваивание

Оператор присваивания (`=`) рассматривается как выражение присваивания, которое вычисляется равным выражению правой части, и в то же время он присваивает вычисленное значение выражения переменной заданной в левой части:

```
var name2=10;
```

#### 2. Арифметическое выражение

Вычисляемым значением арифметического выражения является число. Создаются с помощью арифметических операторов.

Таблица 2.

Оператор	Действие
<code>+</code>	Сложение
<code>-</code>	Вычитание
<code>*</code>	Умножение
<code>/</code>	Деление
<code>%</code>	Остаток от деления целых чисел
<code>++</code>	Увеличение значения на единицу
<code>--</code>	Уменьшение значения на единицу

#### 3. Логическое выражение

Вычисляемым значением логического выражения может быть `true` или `false`. Для создания используются операторы сравнения или логические операторы, применяемые к переменным любого типа.

Таблица 3.

Операторы сравнения	Значение	Логические Операторы	Значение
<code>==</code>	Равно	<code>&amp;&amp;</code>	логическое И
<code>!=</code>	Не равно	<code>  </code>	логическое ИЛИ
<code>&gt;=</code>	Больше или равно	<code>!</code>	логическое НЕ

$\leq$	Меньше или равно		
$>$	Строго больше		
$<$	Строго меньше		

#### 4. Строковые выражения

Вычисляемым значением строкового выражения является число. В JavaScript существует только один строковый оператор – оператор конкатенации (сложения) строк:

```
string1 = "Моя " + "строка"
```

### 1.3. Управляющие конструкции языка JavaScript

#### *Операторы JavaScript*

Операторы служат для управления потоком команд в JavaScript. Блоки операторов должны быть заключены в фигурные скобки.

##### 1. Операторы выбора

- условный оператор if

Эта управляющая структура используется, когда необходимо выполнить некий программный код в зависимости от определенных условий. Также предусмотрена конструкция if-else (если-тогда-иначе).

```
if (условие_1)
{
    оператор_1;      // эти операторы выполняются, если условие_1
верно
    оператор_2;
}
else
{
    оператор_3; // эти операторы выполняются, если условие_1 ложно
    оператор_4;
}
```

Условие для проверки (вопрос компьютеру) записывается сразу после слова if в круглых скобках. После этого в фигурных скобках пишется то, что будет предприниматься в случае выполнения условия. Далее else и снова в фигурных скобках то, что выполнится в случае, если условие не сработает. Количество различных действий между фигурными скобками неограниченно, фактически можно выполнить две различные программы. При сравнении можно использовать логические выражения. Например:

```
<script language="JavaScript">
var x = 5;
var y = 10;
if (x>y) {
    alert('x - максимальное число')
}
else
{
    alert('y - максимальное число')
}
</script>
```

- оператор выбора switch

Это фактически несколько условных операторов, объединенных в одном. В данном операторе вычисляется одно выражение и сравнивается со значениями, заданными в блоках case. В случае совпадения выполняются операторы соответствующего блока case.

```
switch (выражение) {
    case значение1:
        оператор_1;
        break;
    case значение2:
        оператор_2;
        break;
    ....
    default:
        оператор;
}
```

Если значение выражения не равняется ни одному из значений, заданных в блоках case, то вычисляется группа операторов блока default, если этот блок задан, иначе происходит выход из оператора switch. Необязательный оператор break, задаваемый в блоках case, выполняет безусловный выход из оператора switch.

## 2. Операторы цикла

Оператор цикла повторно выполняет последовательность операторов JavaScript, определенных в его теле, пока не выполниться некоторое заданное условие.

- цикл for (цикл со счетчиком)

```
for (i=1; i<10; i++){
    <тело цикла>
}
```

Первый параметр (i=1) определяет счетчик и указывает его начальное значение. Этот параметр называется начальным выражением, поскольку в нем задается начальное значение счетчика (начальное значение в данном

случае равно единице). Это выражение инициализации выполняется самым первым и всего один раз.

Второй параметр ( $i < 10$ ) - это условие, которое должно быть истинным, чтобы цикл выполнялся, как только условие цикла становится ложным, работа цикла завершается. Он называется условием цикла. Проверка условия цикла осуществляется на каждом шаге; если условие истинно, то выполняется тело цикла (операторы в теле цикла). Цикл в данном случае выполнится только девять раз так как задано условие  $i < 10$ .

Третий параметр ( $i++$ ) - это оператор, который выполняется при каждом последовательном прохождении цикла. Он называется выражением инкремента, поскольку в нем задается приращение счетчика (приращение счетчика в данном случае равно единице). Пример автоматической прорисовки нескольких линий с помощью цикла for.

```
<script language="JavaScript" type="text/JavaScript">
for (var i=1; i<10; i++){
    document.write("<hr align='center' width='100'>");
}
</script>
```

- цикл while (цикл с предусловием)  
while (условие)  
{  
 <тело цикла>  
}

Пока значение условия - true (истинно), выполняется тело цикла. Тело цикла может быть представлено простым или составным оператором.

Оператор while содержит в скобках все необходимые параметры условия цикла (логическое выражение). После определения всех параметров цикла вводится открывающая фигурная скобка, символизирующая начало тела цикла. Закрывающая фигурная скобка вводится в конце тела цикла. Все операторы, введенные в скобках, выполняются при каждом прохождении цикла.

```
<script language="JavaScript">
var i=1;
while(i<=10){
    document.write('число=' + i + '<br>');
    i=i+2;
}
</script>
```

- прерывание и перезапуск цикла

Оператор прерывания break позволяет прервать выполнение цикла и перейти к следующему за ним выражению:

```
a = 10;
i = 1;
while (a<100){
```

```
a = a * i;  
if (i>4) break;  
++i;  
}
```

Если значение *i* превысит 4, то прерывается выполнение цикла.

Оператор перезапуска *continue* позволяет перезапустить цикл, т.е. оставить невыполненным все последующие выражения, входящие в тело цикла, и запустить выполнение цикла с самого начала.

```
a = 10;  
i = 1;  
while (a<100){  
    ++i;  
    if (i>2 && i<11) continue;  
    a = a * i;  
}
```

### *Создание и вызов функций в JavaScript*

В JavaScript функцией называется именованная часть программного кода, которая выполняется только при обращении к ней посредством указания ее имени. Функции создаются с помощью ключевого слова *function*. Обычно функции располагают в секции *<head>*. Такое расположение функций в HTML-документе гарантирует их полную загрузку до того момента, когда их можно будет вызвать из секции *<body>*.

После названия функции (*func\_name*) ставятся двойные круглые скобки, программный код при этом заключается в фигурные скобки:

```
<script language="JavaScript">  
function func_name()  
{  
    программный код функции (тело функции)  
}  
</script>
```

Для того, чтобы вызвать функцию в нужном месте, необходимо просто указать ее имя в тексте:

```
<script language="JavaScript">  
func_name();  
</script>
```

Второй вариант вызова функции непосредственно в HTML теге:

```
<a href="javascript:func_name()">Текст ссылки</a>
```

Ниже приведен код страницы HTML, после загрузки которой каждые три секунды будет появляться сообщение, генерируемое вызовом функции *myMessage()*:

```
<script>
```

```

function myMessage()
{
    alert("My Message")
}
</script>
<body onload='setTimeout ("myMessage()",3000)'>
<p>Каждые три секунды будет появляться сообщение</p>
</body>

```

Метод `setTimeout()` запускает выполнение кода JavaScript, задаваемого первым строковым параметром, через определенный промежуток времени после выполнения метода.

Интервал задается в миллисекундах (1000 соответствует 1 секунде).

#### **1.4. Стандартные объекты и функции ядра JavaScript**

##### *Объект Array*

Массив - упорядоченный набор однородных данных, к элементам которого можно обращаться по имени и индексу. Язык JavaScript не имеет встроенного типа данных для создания массивов, поэтому для решения используется объект `Array` и его методы.

Для создания объекта `Array` вызывается оператор `new` и конструктор массива - системная функция (ее имя совпадает с именем объекта), инициализирующая элементы массива:

`m=new Array();`

Заполнение массива происходит позже. Например:

```

<script language="JavaScript">
//создание нового массива
m=new Array();
//заполнение массива
m[0]=1;
m[1]=2;
m[2]=4;
m[3]=56;
</script>

```

В приведенном выше примере с помощью команды `new` создается массив `m`, а затем происходит его заполнение - каждому элементу присваивается определенное значение.

`m=new Array(1,2,4,56);`

Вызывается команда `new` и сразу задаются значения всех элементов массива.

```

<script language="JavaScript">
//создание нового массива и его заполнение

```

```
m=new Array(1,2,4,56)
</script>
```

Объявление строковых массивов проводится тем же способом, что и объявление числовых массивов.

Таблица 4.

Методы объекта Array	Действие
join()	Объединяет все элементы массива в одну строку с указанием разделителя.
reverse()	Изменяет порядок элементов в массиве - первый элемент становится последним, последний - первым
sort()	Выполняет сортировку элементов массива
split()	Разделяет строку на составные части
concat()	Объединяет два массива в один
slice()	Выделяет часть массивы
toString()	Возвращает строку - результат конкатенации всех элементов массива. Элементы массива в строке разделены запятой.
Свойство length	Возвращает длину массива (число элементов в нем).

Пусть определены два массива:

```
array1 = new Array("Первый","Второй","Третий");
```

```
array2 = new Array("Один","Два","Три");
```

Тогда метод join() первого массива array1.join() возвратит строку:  
"Первый,Второй,Третий"

Метод sort() первого массива array1.sort() упорядочит элементы массива array1 (переставив их местами непосредственно в самом массиве array1) в алфавитном порядке:

```
array1[0] = "Второй";
array1[1] = "Первый";
array1[2] = "Третий";
```

Поскольку некоторые методы массива возвращают массив, то к нему можно сразу же применить какой-либо метод, продолжив "точечную" нотацию. Например, array1.concat(array2).sort() объединит два массива в один новый и отсортирует его.

### *Объект Date*

Используется для представления дат в программах JavaScript. Время храниться в виде числа миллисекунд, прошедших от 1 января 1970 года.

Данный объект создается также, как и любой объект в JavaScript – с помощью оператора new и конструктора, в данном случае Date():

```
date1 = new Date(); // значением переменной date1 будет текущая дата
```

Параметром конструктора может быть строка, в которой записана нужная дата:

```
date1 = new Date("january 14, 2000, 12:00:00");
```

Можно задать список параметров:

```
date1 = new Date(2000, 1, 14, 12, 0, 0);
```

### *Объект Math*

В свойствах данного объекта хранятся основные математические константы, а его методы вычисляют основные математические функции. При обращении к данному объекту, создавать его не надо, но необходимо явно указывать его имя Math. Например:

```
p = Math.PI; // хранится значение числа пи.
```

### *Объект String*

Можно явно создавать строковый объект, используя оператор new и конструктор:

```
myString = new String("Hello!");
```

Данный объект имеет единственное свойство length, хранящее длину строки, содержащейся в строковом объекте, и два типа методов: одни непосредственно влияют на содержание самой строки, вторые возвращают отформатированный HTML-вариант строки.

### *Стандартные функции верхнего уровня*

В JavaScript существуют несколько функций, для вызова которых не надо создавать никакого объекта, она находится вне иерархии объектов.

Функция parseFloat(parameter) анализирует значение переданного ей строкового параметра на соответствие представлению вещественного числа.

Функция parseInt(parameter, base) пытается возвратить целое число по основанию, заданному вторым параметром.

Эти функции полезны при анализе введенных пользователем данных в полях формы до их передачи на сервер.

Функции Number(object) и String(object) преобразуют объект, заданный в качестве его параметра в число или строку.

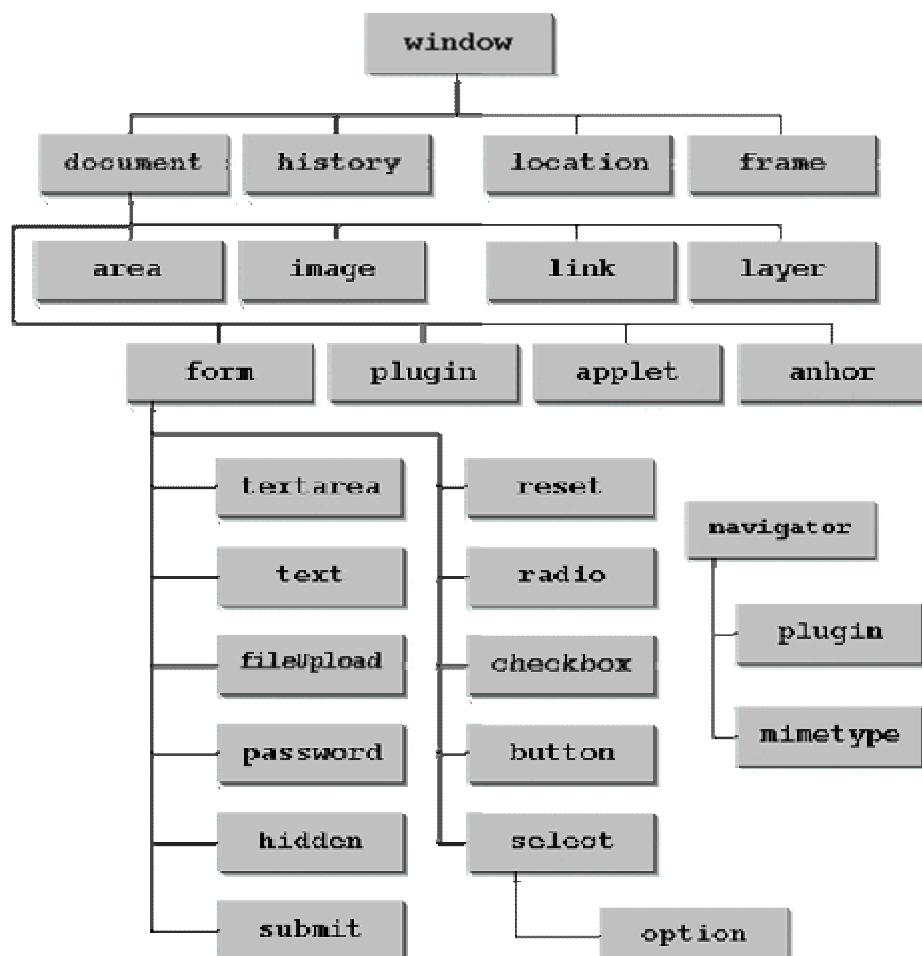
## 1.5. Объекты клиента

При интерпритации страницы HTML браузером создаются объекты JavaScript, свойства которых представляют значения параметров тэгов языка HTML.

### Иерархия объектов

Созданные объекты существуют в виде иерархической структуры, отражающей структуру самой HTML-страницы. На верхнем уровне расположен объект window, представляющий собой активное окно браузера. Далее вниз по иерархической лестнице следуют объекты frame, document, location и history и т.д.

Значения свойств объектов отражают значения соответствующих параметров тэгов страницы или установленных системных параметров. На рисунке показана структура объектов клиента (браузера).



Особняком стоит объект navigator с двумя дочерними (подчиненными) объектами. Он относится к самому браузеру, и его

свойства позволяют определить характеристики программы просмотра. Каждая страница в добавление к объекту `navigator` обязательно имеет еще четыре объекта:

- `window` — объект верхнего уровня, свойства которого применяются ко всему окну, в котором отображается документ;
- `document` — свойства которого определяются содержимым самого документа: связи, цвет фона, формы и т. д.;
- `location` — свойства которого связаны с url-адресом отображаемого документа;
- `history` — представляет адреса ранее загружавшихся HTML-страниц.

Кроме указанных объектов страница может иметь дополнительные объекты, зависящие от ее содержимого, которые являются дочерними объектами объекта `document`. Если на странице расположена форма, то все ее элементы являются дочерними объектами этой формы. Для задания точного имени объекта используется точечная нотация с полным указанием всей цепочки наследования объекта. Наиболее общий объект высшего уровня находится слева в выражении, и слева направо происходит переход к более частным объектам, являющимся при этом наследниками высших в иерархии объектов.

Кроме этих классов объектов пользователь может создавать и свои собственные. Но обычно большинство программ используют эту систему классов и не создают новых.

### *Объект navigator*

Этот объект применяется для получения информации о версиях.

Синтаксис:

`navigator.name_properties`

Методы и события, догадаться не определены для этого объекта. Да и свойства только для чтения, так как ресурс с информацией о версии недоступен для редактирования.

Свойства

- `appCodeName` - кодовое имя браузера;
- `appName` - название браузера;
- `appVersion` - информация о версии браузера;
- `userAgent` - кодовое имя и версия браузера;

Ниже приведен пример использования объекта `navigator`.

```
<html>
<head>
<title> navigator </title>
<script language="javascript">
var firstn = window.prompt("Введите ваше имя: ","ваше имя")
function welcome(){
```

```

var appname=navigator.appName
var appver=navigator.appVersion
window.alert("Привет, " + firstn + ". Вы используете " +
appname + ". Версия " + appver + ". Спасибо за визит.");
}
function writename(){
document.write(firstn + ".");
}
</script>
</head>
<body onLoad="welcome()">
Добро пожаловать,
<script language="JavaScript">
writename();
</script>
</body>
</html>

```

### *Объект window*

Объект `window` создается автоматически при запуске браузера, так как для отображения документа необходимо окно. Одно из назначений объекта окна - это создание нового окна. Новое окно браузера создается с помощью метода `window.open()`. Метод `window.open()` имеет ряд дополнительных аргументов, которые позволяют задать местоположение окна, его размер и тип, а также указывают, должно ли окно иметь полосы прокрутки, полосу команд и т. п. Помимо этого можно задавать и имя окна.

В общем виде данный метод можно представить следующим образом:

```
window.open('url', 'name', 'parameters')
```

Рассмотрим синтаксис более подробно:

- Первый параметр метода `window.open()` - это url документа, загружаемого в окне. Если его не заполнить, то окно останется пустым.
- Второй параметр определяет название окна (`name`). Это имя может использоваться для обращения к созданному окну.
- Третий параметр представляет список необязательных опций, разделенных запятой. С их помощью Вы определяете вид нового окна: наличие в нем панелей инструментов, строки состояния и других элементов.

Приведем таблицу с описанием параметров нового окна, задаваемого третьим параметром (`parameters`) метода `open()`.

Таблица 5.

Параметр	Значение		Описание
fullscreen	yes 1	no 0	указывает, показывается ли новое окно на полный экран или как обычное окно. По умолчанию показывается обычное окно
channelmode	yes 1	no 0	позволяет указать, отображается ли полоса каналов
toolbar	yes 1	no 0	позволяет указать, отображается ли полоса кнопок
location	yes 1	no 0	позволяет указать, отображается ли полоса для ввода адреса
directories	yes 1	no 0	позволяет указать, отображается ли полоса кнопок для выбора каталогов
status	yes 1	no 0	позволяет указать, отображается ли полоса статуса
menubar	yes 1	no 0	позволяет указать, отображается ли полоса меню
scrollbars	yes 1	no 0	задает отображение горизонтальной и вертикальной полос прокрутки
resizable	yes 1	no 0	позволяет указать, может ли окно изменять свой размер
width	yes 1	no 0	задает ширину окна в пикселях. Минимальное значение - 100
height	yes 1	no 0	задает высоту окна в пикселях. Минимальное значение - 100
top	yes 1	no 0	задает вертикальную координату левого верхнего угла окна
left	yes 1	no 0	задает горизонтальную координату левого верхнего угла окна

Объект window использует три метода отображения сообщений:

- метод `prompt()` – выводит диалоговое окно с полем ввода, куда пользователь может ввести информацию
- метод `alert()` – выводит на экран окно - сообщение с кнопкой OK и определенным программистом текстом
- метод `confirm()` – выводит диалоговое окно с кнопками OK и Cancel. Дает возможность пользователю продолжить или отменить предложенную операцию.

Сообщение, которое вы хотите вывести на экран, набирается в кавычках внутри круглых скобок.

Данный скрипт запрашивает имя посетителя и выдает приветствие с введенным именем.

```
<script language="JavaScript">
name=window.prompt ("Введите, пожалуйста, свое имя", "Ваше имя");
window.alert ("Вас зовут, " + name);
</script>
```

В этом фрагменте кода метод `prompt` имеет следующие параметры: текст запроса и значение, заполняющее поле ввода по умолчанию; переменная `name` - имя переменной, куда сохраняется введенная информация (имя может быть любым).

### *Объект document*

Объект `document` имеет дело прежде всего с телом HTML-страницы. Он имеет несколько дочерних объектов (коллекций): `all`, `images`, `link`, `anchor` и `form`. Пользуясь объектной моделью построения документа можно, например, обратиться к любой картинке на странице через следующий синтаксис:

```
document.images.name.src
```

Для `document` не существует никаких событий. Некоторые свойства и методы перечислены в таблице, из методов наиболее употребимы `write` и `writeln`.

Таблица 6.

Свойства	Назначение
<code>bgColor</code>	Устанавливает цвет фона текущего документа. Этот цвет может иметь шестнадцатеричное представление <code>#rrggbb</code> или соответствующее название. Синтаксис: <code>document.bgColor="#e7e6d8"</code>
<code>fgColor</code>	Устанавливает цвет текста документа. Аналогичен по функциям свойству <code>bgColor</code>
<code>referrer</code>	Указывает url документа, на который ссылается пользователь в настоящее время. Например, если кто-то обратился по адресу: <code>http://www.nm.org/welcome.htm</code> с сервера

	http://www.someplace.com, то свойством referrer будет: http://www.someplace.com, если это свойство было в странице вышеупомянутого расположения; в противном случае оно обращается в null
location	Соответствует адресу url текущего документа

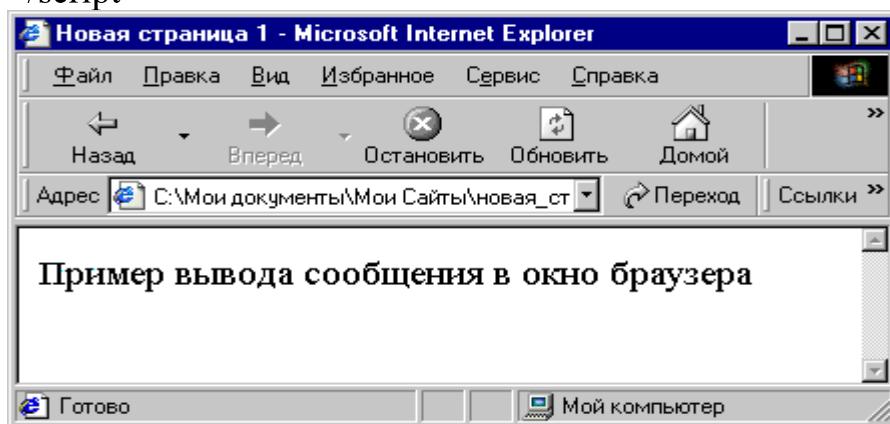
Таблица 7.

Методы	Назначение
write() writeln()	Записывает HTML-текст в текущий документ и должен вызываться, когда документ открывается для записи. Синтаксис: document.write('somestring'), где somestring может быть одной строкой, переменной или же несколькими связанными строками в формате HTML, которые выводятся на экран
lastModified()	Показывает дату последней модификации документа: date1 = document.lastModified
open()	Открывает документ для записи дополнительных строк в формате HTML: document.open()
close()	Закрывает документ, который вызывался методом document.open(): document.close().

Методы write() / writeln().

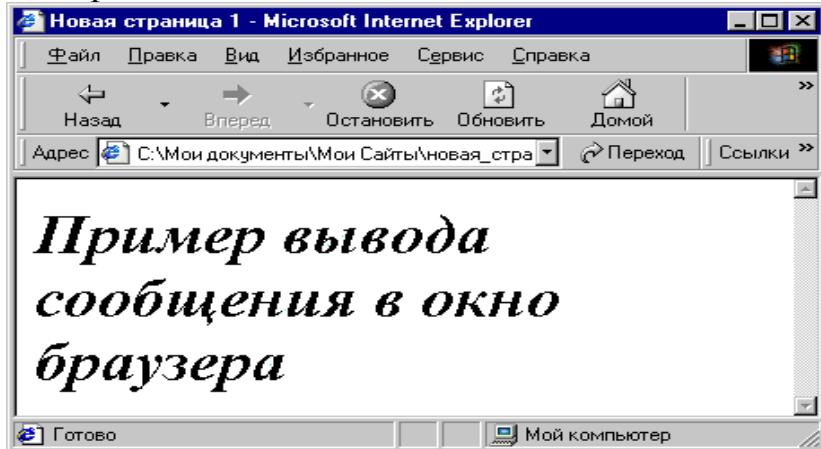
Вызов метода document.write() с указанием определенных параметров приводит к отображению текста в окне браузера. В качестве параметра при вызове метода document.write() мы указываем строку, которую хотели бы увидеть на экране.

```
<script language="JavaScript">
document.write('Пример вывода сообщения в окно браузера')
</script>
```



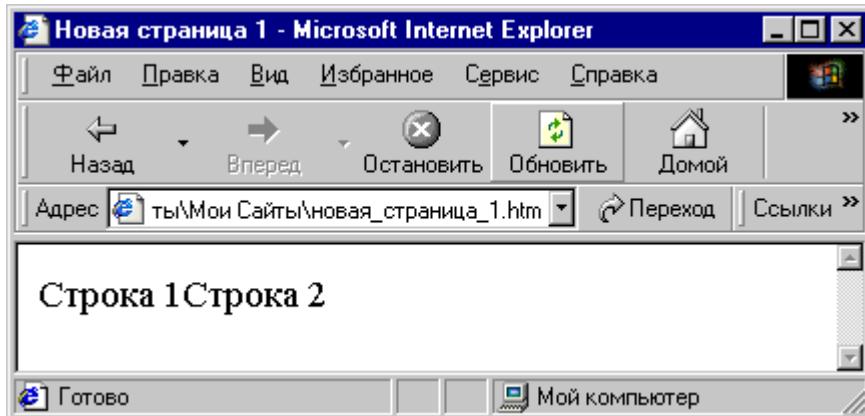
Выводимая строка может содержать и тэги языка HTML. В этом случае браузер выведет данную строку точно так же, как если бы она была размещена непосредственно в HTML документе.

```
<script language="JavaScript">
    document.write('<h1><b><i>Пример вывода сообщения в окно
браузера</i></b></h1>')
</script>
```



При написании скрипта, содержащего несколько команд `document.write()` подряд, при выводе в браузер текст окажется на одной строке

```
<script language="JavaScript">
    document.write('Строка 1');
    document.write('Строка 2');
</script>
```



Для размещения каждого куска текста в новом абзаце можно использовать 2 способа:

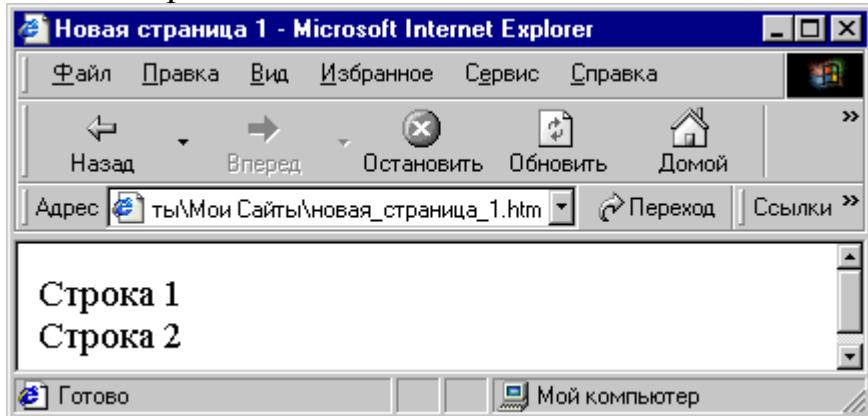
1. либо тег `<p>`, либо тег `<br>`, который включается в состав выводимой строки;
2. используя метод `document.writeln()`.

Следующие примеры приведут к одинаковому результату:

```
<script language="JavaScript">
    document.write('Строка 1<br>');
    document.write('Строка 2<br>');
</script>
```

и

```
<script language="JavaScript">
document.writeln('Строка 1') ;
document.writeln('Строка 2') ;
</script>
```



### *Объект location*

Объект *location* содержит информацию о местонахождении текущего документа, т.е. его интернет-адрес. Его также можно использовать для перехода на другой документ и перезагрузки текущего документа.

Таблица 8.

Свойство	Описание
hash	Имя "якоря" в интернет-адресе документа, если оно есть.
host	Имя компьютера в сети интернет вместе с номером порта, если он указан.
hostname	Имя компьютера в сети Интернет.
href	Полный интернет-адрес документа.
pathname	Путь и имя файла, если они есть.
port	Номер порта. Если не указан, возвращает номер 80 - стандартный порт, через который работает протокол http.
protocol	Идентификатор протокола. Если не указан, возвращается "http:".
search	Строка параметров, если она есть.

Таблица 9.

Метод	Описание
assign(url)	Загружает документ, адрес которого передан в качестве параметра. Поддерживает только IE начиная с 4.0
reload()	Перезагружает документ с Web-сервера.
replace(url)	Загружает документ, адрес которого передан в качестве параметра, и заменяет в списке истории Web-обозревателя адрес предыдущего документа адресом нового.

Пользуясь объектом location, можно загрузить другой документ на место текущего. Для этого просто необходимо присвоить значение нового интернет-адреса свойству href.

```
document.location.href = "http://www.---.ru";
```

Если вы хотите полностью заменить текущий документ, чтобы даже адрес его не появлялся в списке истории, воспользуйтесь методом replace:

```
document.location.replace("http://www.--.ru");
```

### *Объект form*

Каждая форма в документе, определенная тегом <form>, создает объект form, порождаемый объектом document. Ссылка на этот объект осуществляется с помощью переменной, определенной в атрибуте name тега <form>. В документе может быть несколько форм, поэтому для удобства ссылок и обработки в объект document введено свойство-массив forms, в котором содержатся ссылки на все формы документа. Ссылка на первую форму задается как document.forms[0], на вторую - document.forms[1] и т.д. Вместо индекса в массиве forms можно указывать имя формы. Например, если в документе присутствует единственная форма со значением атрибута name=form1, то любой из следующих операторов JavaScript содержит ссылку на эту форму:

```
document.forms[0];
document.forms["form1"];
document.form1;
```

Последний оператор возможен в силу того, что объект document порождает объект form (как и все остальные объекты, соответствующие элементам HTML страницы) и ссылку на него можно осуществлять по обычным правилам наследования языка JavaScript.

Все элементы формы порождают соответствующие объекты, подчиненные объекту родительской формы. Таким образом, для ссылки на объект text (с параметром name = text1) формы form1 можно пользоваться любым из нижеприведенных операторов:

```
document.forms[0].text1;
document.forms["form1"].text1;
```

```
document.form1.text1;
```

Кроме имени элементы формы, имеют свойство value, значение которого определяется смыслом атрибута value элемента формы. Например, для элементов text и textarea значением этого свойства будет строка содержимого полей ввода этих элементов; для кнопки подтверждения - надпись на кнопке и т.д. Обратиться к свойству value можно по тому же принципу, например:

```
document.form1.text1.value
```

## 1.6. Обработка событий

Использование языка JavaScript при обработке событий значительно расширило возможности языка HTML. Чаще всего программы создаются для обработки информации, вводимой пользователем в поля форм. Возможности управления элементами форм обеспечиваются главным образом за счет функций обработки событий, которые могут быть заданы для всех элементов формы. События делятся на несколько категорий:

- события, связанные с документами (события документа) - загрузка и выгрузка документов;
- события, связанные с гиперсвязью (события гиперсвязи) - помещение указателя мыши на гиперсвязь и активизация гиперсвязи;
- события, связанные с формой (события формы) –
  - щелчки мыши на кнопках;
  - получение и потеря фокуса ввода и изменение содержимого полей ввода, областей текста и списков;
  - выделение текста в полях ввода и областях текста;

События, связанные с документами, возникают при загрузке и выгрузке документа, в то время как события гиперсвязей возникают при их активизации или при помещении на них указателя мыши. Чтобы обеспечить перехват события, необходимо написать функцию-обработчик события. В качестве обработчиков событий могут быть заданы целые функции языка JavaScript или только группы из одного или нескольких операторов. В таблице перечислены имена событий и условия их возникновения:

Таблица 10.

Имя события	Атрибут HTML	Условие возникновения события
Blur	onBlur	Потеря фокуса ввода элементом формы
Change	onChange	Изменение содержимого поля ввода или области текста, либо выбор нового элемента списка

Click	onClick	Щелчок мыши на элементе формы или гиперсвязи
Focus	onFocus	Получение фокуса ввода элементом формы
Load	onLoad	Завершение загрузки документа
Unload	onUnload	Выгрузка текущего документа и начало загрузки нового
MouseOver	onMouseOver	Помещение указателя мыши на гиперсвязь
MouseOut	onMouseOut	Помещение указателя мыши не на гиперсвязь
Select	onSelect	Выделение текста в поле ввода или области текста
Submit	onSubmit	Передача данных формы

### *Атрибут onClick*

Атрибут onClick может использоваться в следующих тегах HTML:

- <a href="url" onClick="function()"><...>
- <input type="checkbox" onClick="function()"><...>
- <input type="radio" onClick="function()"><...>
- <input type="reset" onClick="function()"><...>
- <input type="submit" onClick="function()"><...>
- <input type="button" onClick="function()"><...>

Операторы языка JavaScript, заданные в атрибуте onClick, выполняются при щелчке мыши на таких объектах как гиперсвязь, кнопка перезагрузки формы или контрольный переключатель. Для контрольных переключателей и селекторных кнопок событие Click возникает не только при выборе элемента, но и при разблокировании.

Разберем пример использования атрибута onClick для кнопок, определенных тегами

<input type="button"> в контейнере <form> ... </form>:

```

...
<script language="JavaScript">
function but1() {
    alert("Вы нажали первую кнопку");
}
function but2() {
    alert("Вы нажали вторую кнопку");
}
</script>
...
<form>
<input type="button" value="Первая кнопка" onClick="but1()">

```

```
<input type="button" value="Вторая кнопка" onClick="but2()">
</form>
...

```

### *Работа с меню*

Список в форме задается с помощью объекта select, обработка событий выполняется с помощью следующих параметров:  
onChange - вызывается при изменении выбора;  
onBlur - вызывается при снятии фокуса с объекта;  
onFocus - вызывается при перемещении фокуса на объект.

Рассмотрим следующий пример:

```
...
<script language="JavaScript">
function selectBlur()
{
    document.myForm7.myText.value="Вы нажали поле вне списка ";
}
function selectFocus()
{
    document.myForm7.myText.value="Вы нажали ту же кнопку ";
}
function selectChange()
{
    document.myForm7.myText.value="Вы нажали другую кнопку ";
}
</script>
...
<form name="myForm7">
<input type="text" name="myText" size=40 value="Город"><br>
<select      name="script"      multiple      onBlur="selectBlur()"
onFocus="selectFocus()" onChange="selectChange()">
    <option value="town1" selected>Париж
    <option value="town2">Лондон
    <option value="town3">Рим
    <option value="town4">Берлин
</select>
</form>
...

```

### *Управление логикой программного кода при помощи событий*

В объектно-ориентированном программировании нет единой структуры управления работой программы. Есть независимые друг от

друга объекты. Когда пользователь щелкает, например, по ссылке на экране, браузер передает событие Click объекту, тега <a>. Для события “щелчок мыши” в этом объекте предусмотрен стандартный обработчик — он загружает в окно новый документ.

Давайте попробуем “перехватить” это событие:

```
<a href="page1.htm" onClick="alert('Хода нет?')">документ page1</a>
```

Если щелкнуть по ссылке, на экране возникнет надпись “Хода нет?”. Событие перехвачено, но, при закрытии окна alert, видим, что браузер по-прежнему грузит документ page1.htm. При помощи атрибута onClick мы установили в объекте, “отвод” на собственный обработчик. Но когда скрипт нашего обработчика выполнен, управление возвращается к стандартному обработчику, и это вызывает загрузку документа page1.htm.

Отключение стандартной обработки кодируется так:

```
<a href=page1.htm onClick="alert('Хода нет!');return false">документ page1</a>
```

Оператор return указывает возвращаемое функцией значение. Если ее операнд true, то документ загружается, если false, нет.

Подтверждение активизации гиперсвязи.

Аналогичный пример управления логикой программного кода при помощи событий рассмотрен и в следующем примере. Гиперссылка обычно всегда срабатывает по клику мыши, но иногда нужно, чтобы пользователь был уверен, что хочет перейти по ссылке в следующий документ. Для этого существует метод confirm(), который отображает на экране окно сообщения с кнопками "Ok" и "Cancel". Для перехвата события в теге <a href= ... > мы применим событие onClick. Рассмотрите пример подтверждения активизации гиперсвязи:

```
<a href="form.htm" onClick="return confirm('Вы действительно хотите  
перейти по ссылке?')">Подтверждение активизации гиперсвязи</a>
```

### *Определение событий формы*

Объект form имеет два обработчика событий: onSubmit и onReset. В эти обработчики событий, задаваемые в пределах дескриптора <form>, добавляется группа операторов JavaScript или функция, управляющая формой.

Если вы добавите оператор (или функцию) в обработчик onSubmit, то он (или она) вызывается до отправки данных в сценарий CGI. Для того чтобы отменить отправку данных на обработку сценарием CGI, обработчик событий onSubmit должен возвратить значение false. Если же он возвращает значение true, то данные отправляются на сервер. В некоторых случаях необходимо добавить в форму кнопку reset, запускающую обработчик событий onReset.

Для формы одним из важных действий на странице является проверка правильности заполнения полей пользователем на машине клиента до

пересылки их на сервер. В следующем примере разъясняется, как выполнять эту процедуру.

Рассмотрим скрипт, который будет проверять правильность заполнения формы. Необходимо проверить нет ли пустых строк и правильно ли введен e-mail:

```
<html>
<head>
<title>пример формы</title>
<script language="JavaScript">
    function doSend(){
        var v=document.user.e.value.indexOf("@",1)
        if(document.user.f.value==""){
            alert('Вы должны заполнить поле ФИО')
            document.user.f.focus()
        }
        if(document.user.a.value==""){
            alert('Вы должны заполнить поле адреса')
            document.user.a.focus()
        }
        if(document.user.e.value==""){
            alert('Вы должны заполнить поле e-mail')
            document.user.e.focus()
        }
        if(v===-1){
            alert('Адрес e-mail указан неверно')
            document.user.e.select()
            document.user.e.focus()
        }
        else
            document.user.submit()
    }
    </script>
</head>
<body>
<p align="center"><font size=6>Данные о пользователе</font>
<form name="user">
<b>Пожалуйста, укажите данные о себе:</b>
<br>
ФИО<input type="text" name="f" size="30"><br>
Адрес<input type="text" name="a" size="35"><br>
e-mai<input type="text" name="e" size="30"><br>
<input type="button" value="Послать" onClick="doSend()">
<input type="reset" value="Отменить">
</form>
```

```
</p>
</body>
</html>
```

### *Вставка звука*

Если вам необходимо озвучить страницу, вот простейшая инструкция:

```
<bgsound src="music/gimn.mid" loop=infinite>
```

С помощью JavaScript можно разнообразить страницы сайта.

Пример скрипта проигрывания музыки при наведении на заголовок текста:

```
...
<script>
function playHome() {
document.all.sound.src = "music/file.mid"
}
</script>
...
<bgsound id=sound>
<h1 onMouseover=playHome()>Заголовок с музыкой</h1>
...
...
```

## **1.7. DHTML**

DHTML (динамический HTML) – это набор средств, которые позволяют создавать более интерактивные Web-страницы без увеличения загрузки сервера. Другими словами, определенные действия посетителя ведут к изменениям внешнего вида и содержания страницы без обращения к серверу.

DHTML построен на объектной модели документа (Document Object Model, DOM), которая расширяет традиционный статический HTML-документ. DOM обеспечивает динамический доступ к содержимому документа, его структуре и стилям. В DOM каждый элемент Web-страницы является объектом, который можно изменять. DOM не определяет новых тэгов и атрибутов, а просто обеспечивает возможность программного управления всеми тэгами, атрибутами и каскадными таблицами стилей (CSS).

Каждой гиперссылке, заголовку или текстовому параграфу можно присвоить имя, атрибуты стиля или цвета текста и указать это имя в сценарии, имеющемся на данной странице. Сценарий может быть написан на любом существующем скриптовом языке, но здесь подразумевается использование языка JavaScript. Элементы страницы впоследствии могут изменяться в результате определенного события, например при наведении курсора мыши, при щелчке, нажатии клавиши на клавиатуре либо после

истечения заданного промежутка времени. Изменяться может не только стиль и цвет текста, но и весь объект, текст или рисунок.

## *Объединение JavaScript и CSS*

### 1. Пример изменения цвета текста.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 style="color:red">Добро пожаловать на нашу страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

Данный пример применения CSS позволяет сделать заголовок красного цвета. Допустим, вы хотите, чтобы текст заголовка только тогда становился красным, когда пользователь наводит на него курсор. Этого можно добиться с помощью CSS и JavaScript.

Шаг 1. Удаление существующей информации о стиле

Это действие может показаться вам шагом назад, но оно действительно необходимо:

```
<html>
<head>
    <title>Простая страница</title>
</head>
<body>
<h1>Добро пожаловать на нашу страницу! </h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

Шаг 2. Добавление идентификатора

Поскольку вам нужно как-то обращаться к элементу, с которым будут производиться манипуляции, необходимо в тэг <h1> добавить атрибут id - это краткое обозначение, позволяющее указать нужный элемент:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 id="head1">Добро пожаловать на нашу страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

### Шаг 3. Добавление обработчика событий

Следующий шаг — добавление обработчика событий. Этому действию соответствует событие onMouseover. Также следует указать имя функции, которая будет вызываться при выполнении события:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 id="head1" onMouseover ="colorchange ()">Добро пожаловать на
нашу
страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

### Шаг 4. Написание сценария JavaScript

Вам потребуется единственная строка, состоящая из следующих частей:

- имя объекта на странице, с которым должен выполняться ваш сценарий - в данном случае head1;
- применяемый аспект JavaScript - в данном случае style;
- атрибут стиля, который будет изменяться - color;
- новое значение, принимаемое атрибутом стиля - red.

Соедините это, и получится следующая строка:

Head1.style.color = "red"

Добавьте ее в функцию и сохраните файл. В окончательном варианте страница должна выглядеть так:

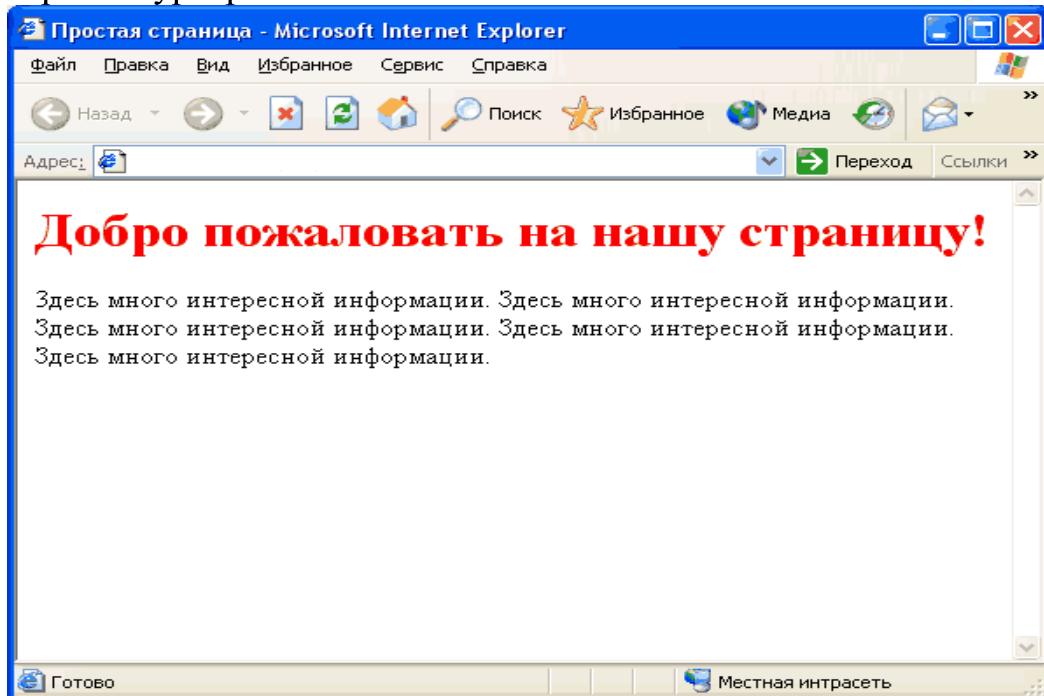
```

<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function colorchange()
{
head1.style.color = "red";
}
</script>
</head>
<body>
<h1 id="head1" onMouseover="colorchange()">Добро пожаловать на
нашу страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>

```

Откройте эту страницу в браузере и посмотрите, что происходит, когда вы наводите курсор на заголовок. Если все было сделано правильно, то цвет заголовка изменится.

Но обратите внимание, что цвет текста не становится прежним, когда вы убираете курсор с заголовка.



## 2. Пример использования атрибута text-decoration.

Используя в сценариях JavaScript атрибуты, название которых пишется через дефис, убирайте дефис и пишите оба слова слитно, причем второе слово должно начинаться с заглавной буквы. Таким образом, textDecoration в сценариях должно выглядеть как textDecoration.

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function addunderline()
{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">Добро пожаловать на нашу
страницу! </h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.</p>
</body>
</html>
```

Необходимо обратить внимание на прописную букву в слове textDecoration — если все слово набрать в нижнем регистре, сценарий выполняться не будет. Теперь при наведении курсора заголовок станет подчеркнутым, а затем, если убрать курсор, вернется в прежнее состояние.

### 3. Пример точного позиционирования текста

Сначала необходимо рассмотреть, каким образом осуществляется позиционирование текста.

Наиболее часто применяются следующие атрибуты позиционирования:

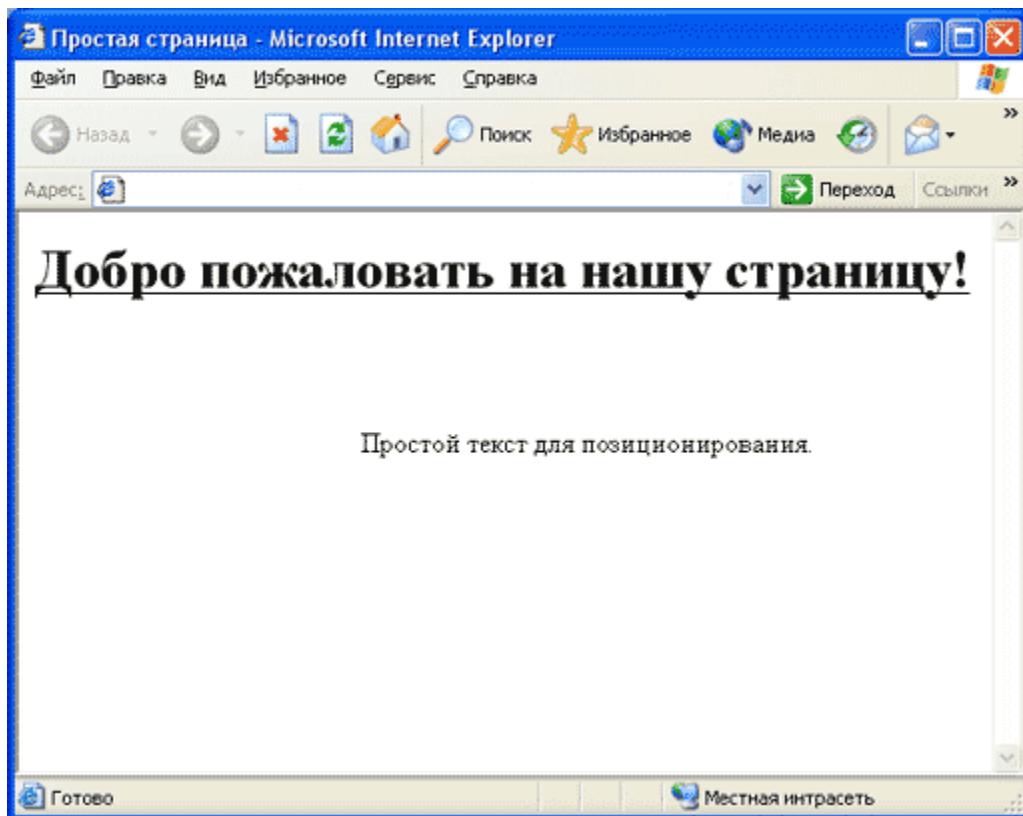
- position - имеет два интересующих нас значения: absolute и relative (по умолчанию значение static). Для значения absolute в качестве точки отсчета используется верхний левый угол окна браузера, и все параметры местоположения отмеряются от него. В свою очередь, для relative точкой отсчета является то место, в котором разместился бы

элемент страницы, если бы не было представлено никакой информации о местоположении;

- top - используется для указания вертикального смещения элемента от точки отсчета. Величина смещения может выражаться в различных единицах (пиксели, дюймы, сантиметры, миллиметры и т.п.). В наших примерах используются пиксели. Положительное значение top соответствует смещению элемента страницы вниз, в то время как отрицательное - по направлению к верхней границе окна браузера;
- left - подобен атрибуту top, но применяется для указания горизонтального направления. Положительное значение соответствует сдвигу элемента вправо, отрицательное - влево.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 style="text-decoration: underline">Добро пожаловать на нашу
страницу!</h1>
<p style="position:absolute; top:125px; left:200px">Простой текст для
позиционирования.</p>
</body>
</html>
```

С помощью CSS текст расположен со значением absolute, то есть его положение отсчитывается от верхнего левого угла окна браузера. Значение атрибута top равно 125px, таким образом, текст будет расположен на 125 пикселов ниже верхнего края страницы. Значение атрибута left равно 200px, то есть текст будет сдвинут на 200 пикселов от левого края окна браузера.



## 1.8. Создание анимационных объектов

Анимация - это процесс «оживления» объекта

Анимация включает в себя две составляющие: расстояние между соседними кадрами, называемое скачком (jump), и временной промежуток между двумя последовательными скачками, называемый интервалом (interval). При больших скачках и длительных интервалах анимация выглядит медленной и грубой. Движение объектов кажется неестественным и воспринимается как мелькание. При малых скачках и кратких интервалах анимация выглядит более плавной, хотя, если чересчур увлечься, движение покажется нарочитым.

### 1. Пример перемещения текста слева направо

Сначала следует ввести текст в тэге <div>, ограничивающем текст, добавить идентификатор id.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<div id="anim">
Текст, шагом марш!
</div>
```

```
</body>
</html>
```

Затем воспользуемся CSS, чтобы поместить текст в начальное положение:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Далее начинаем работать над сценарием JavaScript. Поскольку не нужно, чтобы текст вечно двигался вправо, надо предусмотреть возможность контролирования этого процесса. Чтобы запустить сценарий на выполнение только при условии, если текст находится, например, менее чем в 500 пикселях от левой границы экрана, удобнее всего воспользоваться оператором if. Для этого понадобится атрибут CSS pixelLeft.

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
}
}
</script>
</head>
<body>
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Теперь рассмотрим операторы, управляющие анимацией. Прежде всего нужно задать скачок. Каждый раз текст будет перемещаться вправо на 50 пикселей. Атрибут pixelLeft используется не только для определения положения текста, но и для изменения положения на 50 пикселей:

```

<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
anim.style.pixelLeft +=50;
}
}
</script>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

Далее речь пойдет об интервале. Он задается с помощью метода setTimeout, позволяющего вновь запустить функцию после истечения определенного промежутка времени. Давайте установим интервал до повторного запуска функции moveTxt(), равным 5000 мс:

```

<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
<!-- Маскируемся!
function moveTxt()
{
if (anim.style.pixelLeft &lt; 500)
{
anim.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
&lt;/script&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;div id="anim" style="position:absolute; left:10; top:10"&gt;
Текст, шагом марш!
&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>

```

Процесс будет повторяться до тех пор, пока условие оператора if не станет ложным. Последнее, что нужно сделать, - запустить сценарий на выполнение. Для этого следует воспользоваться событием onLoad:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{
anil.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

## 2. Пример движения текста по диагонали

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop < 500)
{
anim.style.pixelTop += 2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
```

```
</body>
</html>
```

## 1.9. Слои

### *Позиционирование слоя*

Для создания слоев следует использовать тег `<div>` или `<span>`. Эти теги взаимозаменяемы и различаются лишь внешним видом в браузере. Если требуются отступы до и после текста, следует использовать элемент `<div>`. При размещении текста внутри параграфа применяется тег `<span>`.

В Таблице 11 перечислены наиболее важные атрибуты.

Таблица 11.

<code>id</code>	Имя слоя, используемое для указания его в <code>&lt;script&gt;</code>
<code>left</code>	Позиция слоя по x координате
<code>top</code>	Позиция слоя по y координате
<code>position</code>	Задает относительную или абсолютную позицию относительно других объектов
<code>z-index</code>	Позиция слоя при наложении нескольких объектов друг на друга
<code>width</code>	Ширина слоя в пикселях или %
<code>height</code>	Высота слоя в пикселях или %
<code>bgColor</code>	Цвет фона слоя
<code>background</code>	Картинка фона
<code>src</code>	Внешний html документ, содержащийся в слое

Пример наложения текста:

```
<html>
<body>
Слой1 наверху
<div style="position:relative; font-size:50px; z-index:2; color: navy">
Слой 1
</div>
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px;
z-index:1">
Слой 2
</div>
```

Слой 2 наверху

```
<div style="position:relative; font-size:50px; z-index:3; color: navy">
```

Слой 1

```
</div>
```

```
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px; z-index:4">
```

Слой 2

```
</div>
```

```
</body>
```

```
</html>
```

Тип позиционирования слоя определяется параметром position, положение элемента - двумя координатами top и left.

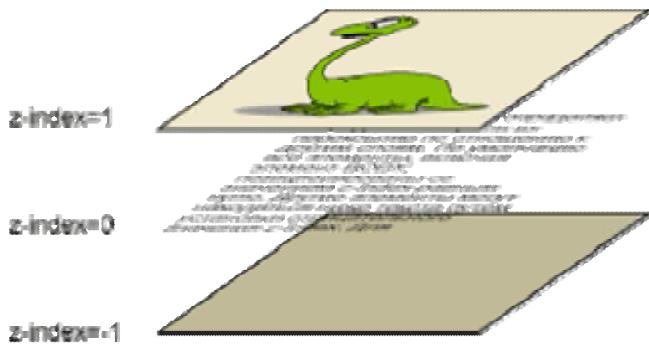
Кроме тегов <div> и <span> абсолютное позиционирование поддерживают следующие элементы: <applet>, <input>, <button>, <object>, <select>, <fieldset>, <iframe>, <table>, <img>, <textarea>.

Параметр position:relative используется для смещения слоя относительно родительского элемента. Установка этого значения не изменяет размещение элемента, но если установлены значения свойств top или left, то слой смещается от своего нормального положения в документе.

В то время как свойство position указывает тип системы координат, параметры top и left определяют точную позицию слоя. Значения этих параметров могут определяться в процентном отношении или пикселях, принимать положительные и отрицательные величины. Это дает возможность размещать контент выше или ниже на странице независимо от физической позиции кода HTML. То есть, в верхней части страницы можно поместить слой, который описан внизу HTML-документа.

### *Свойство z-index*

Свойство z-index определяет порядок слоев, или их перекрытие по отношению к другим слоям. По умолчанию все слои позиционированы со значением z-index равным нулю. Другие слои могут размещаться ниже путем установки отрицательного значения z-index. Для слоев, у которых z-index не установлен, это значение назначается неявно в соответствии с их положением в документе. Поэтому слой, который помещен в документ позже, размещается выше остальных элементов, позиционированных ранее.



### *Свойства visibility и display*

Для отображения или скрытия слоя используется свойство `visibility`. Он может принимать значения `visible`, установленное по умолчанию, для показа слоя, и `hidden`, которое его прячет.

Например, скрытый блок текста можно оформить следующим образом:

```
<div style="visibility: hidden">Спрятанный слой</div>
```

При этом, когда используется данное свойство для скрытия элемента, соответствующий данному элементу блок занимает прежнее положение на странице, но сома содержимо не отображается.

Чтобы на странице не оставалось пустого блока, соответствующего скрываемому элементу, можно использовать свойство `display` со значением `none`. Для отображения элемента `display` равно `block`.

### *Динамическое управление слоями*

Сценарии JavaScript позволяют динамически управлять параметрами установленных слоев. Это позволяет получить такие эффекты, как скрытие и отображение слоя, изменение порядка отображения, перемещение слоя в окне браузера. Все эти эффекты достигаются с помощью изменения соответствующих стилевых параметров установленных слоев.

Для обращения к слоям из сценариев JavaScript, удобнее всего каждому слою дать собственное имя при помощи параметра `id`. Например:

```
<div id="div1">  
...  
</div>
```

Для того, чтобы скрыть отображение слоя `div1`, можно использовать следующую команду:

```
div1.style.visibility='hidden';
```

Для повторного отображения слоя следует выполнить следующее присвоение:

```
div1.style.visibility='visible';
```

Пример динамической смены слоев: в данном примере для отображения некоторого слоя следует нажать на соответствующую

ссылку. Этую идею можно применить и для организации выпадающих меню.

```
<html>
<head>
<style type="text/css">
div {
position: absolute;
top: 20;
left: 160;
visibility: hidden;
}
</style>
<script language="JavaScript">
function show_d(d)
{
div1.style.visibility='hidden';
div2.style.visibility='hidden';
div3.style.visibility='hidden';
div4.style.visibility='hidden';
div5.style.visibility='hidden';
d.style.visibility='visible';
}
</script>
</head>
<body>
<a href="javascript:void(0)" onClick="show_d(div1);">
показать слой 1
</a><br>
<a href="javascript:void(0)" onClick="show_d(div2);">
показать слой 2
</a><br>
<a href="javascript:void(0)" onClick="show_d(div3);">
показать слой 3
</a><br>
<a href="javascript:void(0)" onClick="show_d(div4);">
показать слой 4
</a><br>
<a href="javascript:void(0)" onClick="show_d(div5);">
показать слой 5
</a><br>
<div id="div1">
<h3>Слой номер один</h3>
```

Некоторый текст, на слое расположенный. Его можно скрыть и показать. Текст может содержать несколько строк.

```
</div>
<div id="div2">
<h3>Слой номер два</h3>
Содержит свой текст. Если показывается, то текст на других слоях не
виден.
```

```
</div>
<div id="div3">
<h3>Слой номер три</h3>
```

Тоже текст. При работе со слоями надо следить, чтобы текст одного слоя не "выглядывал" из-под другого слоя при самых различных размерах окна браузера и используемых шрифтах.

```
</div>
<div id="div4">
<h3>Слой номер четыре</h3>
Здесь нет текста.
</div>
<div id="div5">
<h3>Слой номер пять</h3>
И тут тем более нет.
</div>
</body>
</html>
```

### *Динамическое изменение цвета фона ячеек*

Использование стилей и управление ими с помощью JavaScript позволяет менять вид ячейки "на ходу", при выполнении определенных условий, таких как наведение курсора на ссылку или саму ячейку.

Рассмотрим самый простой прием - цвет фона ячейки меняется, когда курсор мыши наводится на нее. Наведение мыши на область отслеживается событием onmouseover, а вывод мыши за ее пределы - событием onmouseout. Поскольку цвет фона меняется у той же самой ячейки, на которую наводим курсор мыши, то изменение стиля делается с помощью метода this.style.background.

```
...
<table width=60% border=1 cellspacing=0 cellpadding=4
bordercolor=#333333 align=center>
<tr>
<td align=center bgcolor="#cccccc"
onMouseover="this.style.background='#ffcc33'"
onmouseout="this.style.background='#cccccc'"><a href="#">Пункт
1</a></td>
<td align=center bgcolor="#cccccc"><a href="#">Пункт 2</a></td>
```

```
</tr>  
</table>
```

```
...
```

## **2. Практика**

### **Постановка задачи**

Необходимо выполнить практические работы №1-№5 и итоговое задание, предложенное в конце.

Для выполнения итогового задания Вам необходимо иметь созданный в курсе «HTML» Web-сайт, состоящий из нескольких (не менее трех) связанных между собой статических HTML-страниц и использующий основные возможности языка HTML.

Для выполнения всех практических работ Вам необходимо иметь текстовый редактор (возможна работа в специальных редакторах Web-документов, например Adobe Dreamweaver), несколько браузеров для просмотра Ваших страниц (Internet Explorer, Mozilla Firefox, Opera и другие).

#### **2.1. Практическая работа №1. Размещение скриптов в HTML-документе.**

##### **Задание 1.**

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды внутри тэга <script>.
4. Добавьте команду вывода аналогичного приветственного сообщения в окно браузера после закрытия диалогового окна.
5. Сохраните документ с именем Ex1.html в рабочей папке.

##### **Задание 2.**

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды JavaScript во внешнем файле. Для этого:
  - создайте новый текстовый файл,
  - поместите в него код JavaScript,

- сохраните файл с именем main.js следующим образом: укажите тип файла “Все файлы”, кодировку “UTF-8”.
4. Добавьте ссылку на внешний скриптовый файл из рабочего HTML-документа.
  5. Сохраните документ с именем Ex2.html в рабочей папке.

### **Задание 3.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex3.html в рабочей папке.
3. Добавьте в документ код JavaScript так, чтобы в диалоговом окне появлялось поле с надписью "Введите сюда своё имя" и со значением по умолчанию в поле "Введите имя". Для этого используйте метод prompt(...) объекта window. Для хранения введенного значения заведите новую переменную.
4. Организуйте вывод введенного значения имени в окно браузера в виде: "Ваше имя <.....>".
5. Дополните код, чтобы в новом диалоговом окне появилась надпись "Начать заново? " При положительном ответе появлялось диалоговое окно: "Не надоело? ", при отказе – "Ну и правильно!". Используйте для написания методы alert(...) и confirm(...) объекта window.

## **2.2. Практическая работа №2. Операторы управления, функции. Объекты ядра JavaScript.**

### **Задание 4.**

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>if</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
var x, y;
x=parseInt(prompt("Введите значение x","")); // метод parseInt()
переводит строку в целое
y=parseInt(prompt("Введите значение y","")); // число
if(x<y)
{
alert("Максимальное число - y")
}
else {
alert("Максимальное число - x")
```

переводит строку в целое

```
y=parseInt(prompt("Введите значение y","")); // число
```

```
if(x<y)
```

```
{
```

```
alert("Максимальное число - y")
```

```
}
```

```
else {
```

```
alert("Максимальное число - x")
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Допишите скрипт так, чтобы при введении пользователем одинаковых чисел, открывалось сообщение "Введенные числа равны!".
3. Напишите скрипт, в котором пользователя просят ввести правильный пароль. При вводе правильного пароля, в окне браузера появляется сообщение о том, что пароль верен. При вводе неправильного пароля – выпадает сообщение о неправильно введенном пароле. Для выполнения задания введите переменную password, в которую сохраните верное значение пароля.
4. Сохраните документ с именем Ex4.html в рабочей папке.

### **Задание 5.**

1. Рассмотрите пример скрипта:

```
<html>
```

```
<head>
```

```
<title>for</title>
```

```
</head>
```

```
<body>
```

```
<h1>Пример простой</h1>
```

```
<script language="JavaScript" type="text/JavaScript">
```

```
function line() {
```

```
    document.writeln("<hr align='center' width='100'>");
```

```
}
```

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

```
    line();
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Создайте вариант прорисованных линий со следующим условием:
  - десять линий должны располагаться друг под другом,
  - первая должна быть длинной 10 пикселей,
  - каждая последующая на 10 пикселей больше.
3. Сохраните документ с именем Ex5.html в рабочей папке.

### **Задание 6.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex6.html в рабочей папке.

3. Добавьте в документ код JavaScript так, чтобы в окне браузера была выведена таблица степеней двойки вида:

Степень	Результат
$2^0$	1
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32

Для этого в сценарии используйте метод write(...) объекта document для формирования содержимого страницы. На каждой итерации цикла for сформируйте очередную строку таблицы, в первую ячейку которой заносится соответствующая степень двойки, а во вторую результат ее возведения в указанную степень. Для выполнения этого действия используется встроенный объект Math и его метод pow(...), возводящий первый параметр в степень, заданную вторым параметром. Обратите внимание, что метод write(...) может вызываться с любым количеством фактических параметров. Результатом его работы в любом случае является вывод в документ строки, полученной конкатенацией всех параметров, переданных в метод.

### Задание 7.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>array</title>
</head>
<body>
<script language="JavaScript">
year=new Array("декабрь","январь","февраль","март","апрель","май",
"июнь","июль","август","сентябрь","октябрь","ноябрь");
summer=new Array(); //летние месяцы
summer=year.slice(6,9);
document.write(summer+"<br>");
</script>
</body>
</html>
```

2. Создайте массив, содержащий названия школьных предметов. Выделите из него два массива. Пусть к первому относятся предметы из раздела точных наук, а ко второму - из раздела гуманитарных

наук. Для создания и вывода в окно браузера новых массивов используйте метод slice(...) и write(...) объекта document. Оформите исполняющий скрипт в виде отдельной функции, описанной в разделе <head> и вызванной в разделе <body>.

3. Сохраните документ с именем Ex7.html в рабочей папке.

### **Задание 8.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex8.html в рабочей папке.
3. Добавьте скрипт, на основе которого будут выполняться следующие условия:
  - если на страницу зашел пользователь через браузер Microsoft Internet Explorer, перенаправьте его автоматически на страницу Ex1.html;
  - если на страницу зашел пользователь через любой другой браузер, перенаправьте его на страницу Ex3.html.Для выполнения задания используйте свойство appName объекта navigator.

## **2.3. Практическая работа №3. Объекты клиентских приложений. Обработка событий.**

### **Задание 9.**

1. Рассмотрите скрипт:

```
<html>
<head>
<title>document</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
document.write("Спасибо, что пришли к нам на курсы!");
</script>
</body>
</html>
```

2. Допишите скрипт так, чтобы

- цвет фона документа был #E7E6D8,
- цвет шрифта – красный,

- внизу выводилась дата последней модификации документа, используйте для этого слияние методов `wtime(...)` и `lastModified(...)` объекта `document`.

3. Сохраните документ с именем Ex9.html в рабочей папке.

### **Задание 10.**

1. Рассмотрите пример скрипта открытия нового окна на странице:

```
<html>
<head>
<title>window</title>
</head>
<body>
<h1>Создание нового окна</h1>
<hr>
<script language="JavaScript" type="text/JavaScript">
window.open("http://www.google.com","","toolbar=no,scrollbars=yes,width=250, height=250, resizable=yes, top=100, left=500")
</script>
</body>
</html>
```

2. Измените скрипт так, чтобы выполнялись следующие условия:

- открытие нового окна происходило при нажатии на ссылку с текстом: «Щелкните на ссылке для получения справочной информации»,
- размеры окна - 500x500,
- есть возможность изменения размеров окна.

Для выполнения задания используйте написание функции.

3. Сохраните документ с именем Ex10.html в рабочей папке.

### **Задание 11.**

1. Создайте страницу с переадресацией на другой адрес (redirect).
2. Измените скрипт так, чтобы переадресация на другой адрес была с задержкой 5 секунд.
3. Сохраните документ с именем Ex11.html в рабочей папке.

### **Задание 12.**

1. Создайте HTML-документ, в котором будет 2 ссылки:

- первая ссылка должна ссылаться на PDF файл; при нажатии на нее выпадает сообщение с предупреждением о том, что для загрузки документа требуется программа Acrobat, и

продолжить загрузку или нет; используйте для написания метод confirm(...) для подтверждения загрузки;

- вторая ссылка должна содержать такой код, чтобы при наведении на нее мыши менялся цвет фона документа на красный.

2. Сохраните документ с именем Ex12.html в рабочей папке.

### Задание 13.

1. Создайте HTML-документ, содержащий любую картинку.

2. Добавьте скрипт с условиями:

- при наведении курсора мыши на картинку она увеличивается,
- при отведении курсора мыши – уменьшается до исходного размера.

Постройте скрипт через использование функций и событий MouseOver и MouseOut.

3. Сохраните документ с именем Ex13.html в рабочей папке.

### Задание 14.

1. Создайте HTML-страницу содержащую следующую форму заполнения данных:

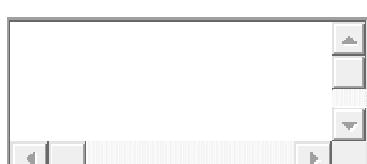
Ваше имя: \*

Пароль \*

Подтверждение пароля\*

Электронный адрес: \*

Тема сообщения:



Сообщение:

\* - необходимые для заполнения поля

2. Добавьте скрипт, проверяющий следующие данные:
  - заполнено ли поле имени,
  - введен ли пароль и содержит ли он больше 4-х символов. Используйте для этого свойство length данного поля,
  - совпадают ли значения, введенные в оба поля для паролей,
  - заполнено ли поле электронного адреса и содержит ли оно символ @,
  - заполнено ли поле сообщения и содержит ли оно больше 10 символов,
3. При несоблюдении условий, курсор должен установиться в то поле, где пользователем введено неверное значение.
4. Сохраните документ с именем Ex15.html в рабочей папке.

## 2.4. Практическая работа №4. Объединение JavaScript и CSS.

### Задание 15.

1. Рассмотрите скрипт:

```
<head>
<title>h1</title>
<script language="JavaScript">
function colorchange()
{
head.style.color = "red";
}
</script>
</head>
<body>
<h1 id="head" onmouseover="colorchange()">Добро пожаловать на
нашу страницу!</h1>
</body>
</html>
```

2. Допишите скрипт страницы таким образом, чтобы красный цвет исчезал после отвода курсора мыши с заголовка.
3. Сохраните документ с именем Ex15.html в рабочей папке.

### Задание 16.

1. Рассмотрите скрипт:

```
<html>
<head>
<title>text decoration</title>
<script language="JavaScript">
function addunderline()
```

```

{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">
Добро пожаловать на нашу страницу!
</h1>
</body>
</html>

```

2. Допишите скрипт страницы таким образом, чтобы на одинарный щелчок мыши появлялось полоса над заголовком, а на двойной щелчок – текст зачеркивался. Используйте события onclick, ondblclick и значения рассматриваемого свойства overline и line-through.

3. Сохраните документ с именем Ex18.html в рабочей папке.

### **Задание 17.**

1. Создайте HTML-документ, содержащий любое изображение.
2. Поместите изображение в тег <div>. Задайте для него абсолютное позиционирование со смещением вниз и влево на 500 пикселей.
3. Сохраните документ с именем Ex17.html в рабочей папке.

## **2.4. Практическая работа №5. Слои. Движущиеся элементы.**

### **Задание 18.**

1. Рассмотрите скрипт:

```

<html>
<head>
<title>simple animation</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{

```

```

anil.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

2. Измените скрипт страницы:

- добейтесь плавного передвижения текста;
- измените направление текста - задайте направление сверху вниз при помощи атрибута pixelTop.

3. Сохраните документ с именем Ex18.html в рабочей папке.

### **Задание 19.**

1. Рассмотрите скрипт:

```

<head>
<title>animal</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop <500)
{
anim.style.pixelTop +=2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

2. Измените направление текста. Задайте направление с верхнего правого угла экрана (приблизительно) по диагонали к середине экрана.

3. Сохраните документ с именем Ex19.html в рабочей папке.

### **Задание 20.**

1. Создайте HTML-страницу, на которой будет три слоя. Верхний и нижней представляют из себя статичные квадраты разного цвета с текстом, а между ними должна проплывать любая картинка слева направо.

### **Абсолютное позиционирование и Z-index**



2. Сохраните документ с именем Ex20.html в рабочей папке.

### **Итоговое задание**

1. Перейдите к Web-сайту, созданному в курсе “Web-программирование: HTML”.
2. Добавьте к странице, содержащей форму, скрипт, осуществляющий проверку введенных в форму данных.
3. Добавьте к остальным страницам скрипты на свое усмотрение.

## **Литература**

1. Пол Вилтон, Джереми МакПик. JavaScript. Руководство программиста. СПб: Питер, 2009-720 с.
2. Стоян Стефанов. JavaScript. Шаблоны. СПб: Символ-плюс, 2011-272 с.
3. Дунаев В.. HTML, скрипты и стили. СПб: БХВ-Петербург, 2011- 816 с.
4. Дронов В.. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. СПб: БХВ-Петербург, 2011- 416 с.
5. Джон Поллок. JavaScript. Руководство разработчика. СПб: Питер, 2011-544 с.
6. Дэвид Макфарланд. JavaScript. Подробное руководство. Эксмо, 2009- 608 с.
7. Климов А. JavaScript на примерах. СПб: БХВ-Петербург, 2009-336 с.
8. Шафер С., HTML, XHTMLи CSS. Библия пользователя. М.: Вильямс, 2010 – 656 с.
9. Лабберс К., Олберс Н., Салим К.. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений. М.: Вильямс, 2011 – 272 с.

*Интернет-ресурсы*

[www.w3.org](http://www.w3.org)



В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

---

### **Кафедра Программных систем**

Кафедра **Программных систем** входит в состав нового факультета **Инфокоммуникационные технологии**, созданного решением Ученого совета университета 17 декабря 2010 г. по предложению инициативной группы сотрудников, имеющих большой опыт в реализации инфокоммуникационных проектов федерального и регионального значения.

На кафедре ведется подготовка бакалавров и магистров по направлению **210700 «Инфокоммуникационные технологии и системы связи»:**

**210700.62.10 – ИНТЕЛЛЕКТУАЛЬНЫЕ  
ИНФОКОММУНИКАЦИОННЫЕ СИСТЕМЫ (Бакалавр)**  
**210700.68.10 – ИНТЕЛЛЕКТУАЛЬНЫЕ  
ИНФОКОММУНИКАЦИОННЫЕ СИСТЕМЫ (Магистр)**

Выпускники кафедры получают фундаментальную подготовку по: математике, физике, электронике, моделированию и проектированию инфокоммуникационных систем (ИКС), информатике и программированию, теории связи и теории информации.

В рамках профессионального цикла изучаются дисциплины: архитектура ИКС, технологии программирования, ИКС в Интернете, сетевые технологии, администрирование сетей Windows и UNIX, создание программного обеспечения ИКС, Web программирование, создание клиент-серверных приложений.

**Область профессиональной деятельности бакалавров и магистров включает:**

- сервисно-эксплуатационная в сфере современных ИКС;
- расчетно-проектная при создании и поддержке сетевых услуг и сервисов;
- экспериментально-исследовательская;
- организационно-управленческая – в сфере информационного менеджмента ИКС.

**Знания выпускников востребованы:**

- в технических и программных системах;
- в системах и устройствах звукового вещания, электроакустики, речевой, и мультимедийной информатики;
- в средствах и методах защиты информации;
- в методах проектирования и моделирования сложных систем;
- в вопросах передачи и распределения информации в телекоммуникационных системах и сетях;
- в методах управления телекоммуникационными сетями и системами;
- в вопросах создания программного обеспечения ИКС.

**Выпускники кафедры Программных систем обладают компетенциями:**

- проектировщика и разработчика структур ИКС;
- специалиста по моделированию процессов сложных систем;
- разработчика алгоритмов решения задач ИКС;
- специалиста по безопасности жизнедеятельности ИКС;
- разработчика сетевых услуг и сервисов в ИКС;
- администратора сетей: UNIX и Windows;
- разработчика клиентских и клиент-серверных приложений;
- разработчика Web – приложений;
- специалиста по информационному менеджменту;
- менеджера проектов планирования развития ИКС.

**Трудоустройство выпускников:**

1. ОАО «Петербургская телефонная сеть»;
2. АО «ЛЕНГИПРОТРАНС»;
3. Акционерный коммерческий Сберегательный банк Российской Федерации;
4. ОАО «РИВЦ-Пулково»;
5. СПБ ГУП «Петербургский метрополитен»;
6. ООО «СоюзБалтКомплект»;

7. ООО «ОТИС Лифт»;
8. ОАО «Новые Информационные Технологии в Авиации»;
9. ООО «Т-Системс СиАйЭс» и др.

**Кафедра** сегодня имеет в своем составе высококвалифицированный преподавательский состав, в том числе:

- 5 кандидатов технических наук, имеющих ученые звания профессора и доцента;
- 4 старших преподавателя;
- 6 штатных совместителей, в том числе кандидатов наук, профессиональных ИТ- специалистов;
- 15 Сертифицированных тренеров, имеющих Западные Сертификаты фирм: Microsoft, Oracle, Cisco, Novell.

Современная техническая база; лицензионное программное обеспечение; специализированные лаборатории, оснащенные необходимым оборудованием и ПО; качественная методическая поддержка образовательных программ; широкие Партнерские связи существенно влияют на конкурентные преимущества подготовки специалистов.

Авторитет специализаций кафедры в области компьютерных технологий подтверждается Сертификатами на право проведения обучения по методикам ведущих Западных фирм - поставщиков аппаратного и программного обеспечения.

Заслуженной популярностью пользуются специализации кафедры ПС по подготовке и переподготовке профессиональных компьютерных специалистов с выдачей **Государственного Диплома** о профессиональной переподготовке по направлениям: "**Информационные технологии (инженер-программист)**" и "**Системный инженер**", а также Диплома о дополнительном (к высшему) образованию с присвоением квалификации: "**Разработчик профессионально-ориентированных компьютерных технологий**". В рамках этих специализаций высокопрофессиональные преподаватели готовят компетентных компьютерных специалистов по современным в России и за рубежом операционным системам, базам данных и языкам программирования ведущих фирм: Microsoft, Cisco, IBM, Intel, Oracle, Novell и др.

Профессионализм, компетентность, опыт, и качество программ подготовки и переподготовки ИТ- специалистов на кафедре ПС неоднократно были удостоены **высокими наградами «Компьютерная Элита»** в номинации лучший учебный центр России.

#### **Партнеры:**

1. Microsoft Certified Learning Solutions;
2. Novell Authorized Education Center;
3. Cisco Networking Academy;

4. **Oracle Academy;**
5. **Sun Java Academy** и др;
6. **Prometric;**
7. **VUE.**

**Мы готовим квалифицированных инженеров в области инфокоммуникационных технологий с новыми знаниями, образом мышления и способностями быстрой адаптации к современным условиям труда.**

Т.В. Зудилова, М.Л. Буркова

## **Web-программирование JavaScript**

### **ПРАКТИКУМ**

В авторской редакции

Редакционно-издательский отдел НИУ ИТМО

Зав. РИО

Н.Ф. Гусарова

Лицензия ИД № 00408 от 05.11.99

Подписано к печати

Заказ №

Тираж 100 экз.

Отпечатано на ризографе

**Редакционно-издательский отдел**  
Санкт-Петербургского национального  
исследовательского университета  
информационных технологий, механики  
и оптики



197101, Санкт-Петербург, Кронверкский пр., 49

## **2. Практика**

### **Постановка задачи**

Необходимо выполнить практические работы №1-№5 и итоговое задание, предложенное в конце.

Для выполнения итогового задания Вам необходимо иметь созданный в курсе «HTML» Web-сайт, состоящий из нескольких (не менее трех) связанных между собой статических HTML-страниц и использующий основные возможности языка HTML.

Для выполнения всех практических работ Вам необходимо иметь текстовый редактор (возможна работа в специальных редакторах Web-документов, например Adobe Dreamweaver), несколько браузеров для просмотра Ваших страниц (Internet Explorer, Mozilla Firefox, Opera и другие).

#### **2.1. Практическая работа №1. Размещение скриптов в HTML-документе.**

##### **Задание 1.**

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды внутри тэга <script>.
4. Добавьте команду вывода аналогичного приветственного сообщения в окно браузера после закрытия диалогового окна.
5. Сохраните документ с именем Ex1.html в рабочей папке.

##### **Задание 2.**

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды JavaScript во внешнем файле. Для этого:
  - создайте новый текстовый файл,
  - поместите в него код JavaScript,

- сохраните файл с именем main.js следующим образом: укажите тип файла “Все файлы”, кодировку “UTF-8”.
4. Добавьте ссылку на внешний скриптовый файл из рабочего HTML-документа.
  5. Сохраните документ с именем Ex2.html в рабочей папке.

### **Задание 3.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex3.html в рабочей папке.
3. Добавьте в документ код JavaScript так, чтобы в диалоговом окне появлялось поле с надписью "Введите сюда своё имя" и со значением по умолчанию в поле "Введите имя". Для этого используйте метод prompt(...) объекта window. Для хранения введенного значения заведите новую переменную.
4. Организуйте вывод введенного значения имени в окно браузера в виде: "Ваше имя <.....>".
5. Дополните код, чтобы в новом диалоговом окне появилась надпись "Начать заново? " При положительном ответе появлялось диалоговое окно: "Не надоело? ", при отказе – "Ну и правильно!". Используйте для написания методы alert(...) и confirm(...) объекта window.

## **2.2. Практическая работа №2. Операторы управления, функции. Объекты ядра JavaScript.**

### **Задание 4.**

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>if</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
var x, y;
x=parseInt(prompt("Введите значение x","")); // метод parseInt()
переводит строку в целое
y=parseInt(prompt("Введите значение y","")); // число
if(x<y)
{
alert("Максимальное число - y")
}
else {
alert("Максимальное число - x")
```

переводит строку в целое

```
y=parseInt(prompt("Введите значение y","")); // число
```

```
if(x<y)
```

```
{
```

```
alert("Максимальное число - y")
```

```
}
```

```
else {
```

```
alert("Максимальное число - x")
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Допишите скрипт так, чтобы при введении пользователем одинаковых чисел, открывалось сообщение "Введенные числа равны!".
3. Напишите скрипт, в котором пользователя просят ввести правильный пароль. При вводе правильного пароля, в окне браузера появляется сообщение о том, что пароль верен. При вводе неправильного пароля – выпадает сообщение о неправильно введенном пароле. Для выполнения задания введите переменную password, в которую сохраните верное значение пароля.
4. Сохраните документ с именем Ex4.html в рабочей папке.

### **Задание 5.**

1. Рассмотрите пример скрипта:

```
<html>
```

```
<head>
```

```
<title>for</title>
```

```
</head>
```

```
<body>
```

```
<h1>Пример простой</h1>
```

```
<script language="JavaScript" type="text/JavaScript">
```

```
function line() {
```

```
    document.writeln("<hr align='center' width='100'>");
```

```
}
```

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

```
    line();
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Создайте вариант прорисованных линий со следующим условием:
  - десять линий должны располагаться друг под другом,
  - первая должна быть длинной 10 пикселей,
  - каждая последующая на 10 пикселей больше.
3. Сохраните документ с именем Ex5.html в рабочей папке.

### **Задание 6.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex6.html в рабочей папке.

3. Добавьте в документ код JavaScript так, чтобы в окне браузера была выведена таблица степеней двойки вида:

Степень	Результат
$2^0$	1
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32

Для этого в сценарии используйте метод write(...) объекта document для формирования содержимого страницы. На каждой итерации цикла for сформируйте очередную строку таблицы, в первую ячейку которой заносится соответствующая степень двойки, а во вторую результат ее возведения в указанную степень. Для выполнения этого действия используется встроенный объект Math и его метод pow(...), возводящий первый параметр в степень, заданную вторым параметром. Обратите внимание, что метод write(...) может вызываться с любым количеством фактических параметров. Результатом его работы в любом случае является вывод в документ строки, полученной конкатенацией всех параметров, переданных в метод.

### Задание 7.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>array</title>
</head>
<body>
<script language="JavaScript">
year=new Array("декабрь","январь","февраль","март","апрель","май",
"июнь","июль","август","сентябрь","октябрь","ноябрь");
summer=new Array(); //летние месяцы
summer=year.slice(6,9);
document.write(summer+"<br>");
</script>
</body>
</html>
```

2. Создайте массив, содержащий названия школьных предметов. Выделите из него два массива. Пусть к первому относятся предметы из раздела точных наук, а ко второму - из раздела гуманитарных

наук. Для создания и вывода в окно браузера новых массивов используйте метод slice(...) и write(...) объекта document. Оформите исполняющий скрипт в виде отдельной функции, описанной в разделе <head> и вызванной в разделе <body>.

3. Сохраните документ с именем Ex7.html в рабочей папке.

### **Задание 8.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex8.html в рабочей папке.
3. Добавьте скрипт, на основе которого будут выполняться следующие условия:
  - если на страницу зашел пользователь через браузер Microsoft Internet Explorer, перенаправьте его автоматически на страницу Ex1.html;
  - если на страницу зашел пользователь через любой другой браузер, перенаправьте его на страницу Ex3.html.Для выполнения задания используйте свойство appName объекта navigator.

## **2.3. Практическая работа №3. Объекты клиентских приложений. Обработка событий.**

### **Задание 9.**

1. Рассмотрите скрипт:

```
<html>
<head>
<title>document</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
document.write("Спасибо, что пришли к нам на курсы!");
</script>
</body>
</html>
```

2. Допишите скрипт так, чтобы

- цвет фона документа был #E7E6D8,
- цвет шрифта – красный,

- внизу выводилась дата последней модификации документа, используйте для этого слияние методов `wtime(...)` и `lastModified(...)` объекта `document`.

3. Сохраните документ с именем Ex9.html в рабочей папке.

### **Задание 10.**

1. Рассмотрите пример скрипта открытия нового окна на странице:

```
<html>
<head>
<title>window</title>
</head>
<body>
<h1>Создание нового окна</h1>
<hr>
<script language="JavaScript" type="text/JavaScript">
window.open("http://www.google.com","","toolbar=no,scrollbars=yes,width=250, height=250, resizable=yes, top=100, left=500")
</script>
</body>
</html>
```

2. Измените скрипт так, чтобы выполнялись следующие условия:

- открытие нового окна происходило при нажатии на ссылку с текстом: «Щелкните на ссылке для получения справочной информации»,
- размеры окна - 500x500,
- есть возможность изменения размеров окна.

Для выполнения задания используйте написание функции.

3. Сохраните документ с именем Ex10.html в рабочей папке.

### **Задание 11.**

1. Создайте страницу с переадресацией на другой адрес (redirect).
2. Измените скрипт так, чтобы переадресация на другой адрес была с задержкой 5 секунд.
3. Сохраните документ с именем Ex11.html в рабочей папке.

### **Задание 12.**

1. Создайте HTML-документ, в котором будет 2 ссылки:

- первая ссылка должна ссылаться на PDF файл; при нажатии на нее выпадает сообщение с предупреждением о том, что для загрузки документа требуется программа Acrobat, и

продолжить загрузку или нет; используйте для написания метод confirm(...) для подтверждения загрузки;

- вторая ссылка должна содержать такой код, чтобы при наведении на нее мыши менялся цвет фона документа на красный.

2. Сохраните документ с именем Ex12.html в рабочей папке.

### Задание 13.

1. Создайте HTML-документ, содержащий любую картинку.

2. Добавьте скрипт с условиями:

- при наведении курсора мыши на картинку она увеличивается,
- при отведении курсора мыши – уменьшается до исходного размера.

Постройте скрипт через использование функций и событий MouseOver и MouseOut.

3. Сохраните документ с именем Ex13.html в рабочей папке.

### Задание 14.

1. Создайте HTML-страницу содержащую следующую форму заполнения данных:

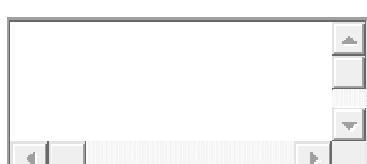
Ваше имя: \*

Пароль \*

Подтверждение пароля\*

Электронный адрес: \*

Тема сообщения:



Сообщение:

\* - необходимые для заполнения поля

2. Добавьте скрипт, проверяющий следующие данные:
  - заполнено ли поле имени,
  - введен ли пароль и содержит ли он больше 4-х символов. Используйте для этого свойство length данного поля,
  - совпадают ли значения, введенные в оба поля для паролей,
  - заполнено ли поле электронного адреса и содержит ли оно символ @,
  - заполнено ли поле сообщения и содержит ли оно больше 10 символов,
3. При несоблюдении условий, курсор должен установиться в то поле, где пользователем введено неверное значение.
4. Сохраните документ с именем Ex15.html в рабочей папке.

## 2.4. Практическая работа №4. Объединение JavaScript и CSS.

### Задание 15.

1. Рассмотрите скрипт:

```
<head>
<title>h1</title>
<script language="JavaScript">
function colorchange()
{
head.style.color = "red";
}
</script>
</head>
<body>
<h1 id="head" onmouseover="colorchange()">Добро пожаловать на
нашу страницу!</h1>
</body>
</html>
```

2. Допишите скрипт страницы таким образом, чтобы красный цвет исчезал после отвода курсора мыши с заголовка.
3. Сохраните документ с именем Ex15.html в рабочей папке.

### Задание 16.

1. Рассмотрите скрипт:

```
<html>
<head>
<title>text decoration</title>
<script language="JavaScript">
function addunderline()
```

```

{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">
Добро пожаловать на нашу страницу!
</h1>
</body>
</html>

```

2. Допишите скрипт страницы таким образом, чтобы на одинарный щелчок мыши появлялось полоса над заголовком, а на двойной щелчок – текст зачеркивался. Используйте события onclick, ondblclick и значения рассматриваемого свойства overline и line-through.

3. Сохраните документ с именем Ex18.html в рабочей папке.

### **Задание 17.**

1. Создайте HTML-документ, содержащий любое изображение.
2. Поместите изображение в тег <div>. Задайте для него абсолютное позиционирование со смещением вниз и влево на 500 пикселей.
3. Сохраните документ с именем Ex17.html в рабочей папке.

## **2.4. Практическая работа №5. Слои. Движущиеся элементы.**

### **Задание 18.**

1. Рассмотрите скрипт:

```

<html>
<head>
<title>simple animation</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{

```

```

anil.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

2. Измените скрипт страницы:

- добейтесь плавного передвижения текста;
- измените направление текста - задайте направление сверху вниз при помощи атрибута pixelTop.

3. Сохраните документ с именем Ex18.html в рабочей папке.

### **Задание 19.**

1. Рассмотрите скрипт:

```

<head>
<title>animal</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop <500)
{
anim.style.pixelTop +=2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

2. Измените направление текста. Задайте направление с верхнего правого угла экрана (приблизительно) по диагонали к середине экрана.

3. Сохраните документ с именем Ex19.html в рабочей папке.

### **Задание 20.**

1. Создайте HTML-страницу, на которой будет три слоя. Верхний и нижней представляют из себя статичные квадраты разного цвета с текстом, а между ними должна проплывать любая картинка слева направо.

### **Абсолютное позиционирование и Z-index**



2. Сохраните документ с именем Ex20.html в рабочей папке.

### **Итоговое задание**

1. Перейдите к Web-сайту, созданному в курсе “Web-программирование: HTML”.
2. Добавьте к странице, содержащей форму, скрипт, осуществляющий проверку введенных в форму данных.
3. Добавьте к остальным страницам скрипты на свое усмотрение.

## Оглавление

Введение .....	5
1. Обзор возможностей языка JavaScript .....	6
1.1. Общий обзор языка .....	7
Основные определения .....	7
Понятие объектной модели применительно к JavaScript.....	9
Размещение операторов языка JavaScript на странице.....	9
1.2. Язык ядра JavaScript.....	10
Синтаксис языка .....	10
Переменные и литералы в JavaScript .....	12
Выражения JavaScript .....	13
1.3. Управляющие конструкции языка JavaScript.....	14
Операторы JavaScript .....	14
Создание и вызов функций в JavaScript.....	17
1.4. Стандартные объекты и функции ядра JavaScript .....	18
Объект Array .....	18
Объект Date .....	19
Объект Math .....	20
Объект String.....	20
Стандартные функции верхнего уровня .....	20
1.5. Объекты клиента .....	21
Иерархия объектов .....	21
Объект navigator .....	22
Объект window.....	23
Объект document.....	25
Объект location.....	28
Объект form.....	29
1.6. Обработка событий .....	30
Атрибут onClick.....	31
Работа с меню .....	32
Управление логикой программного кода при помощи событий .....	32
Определение событий формы .....	33
Вставка звука .....	35
1.7. DHTML.....	35
Объединение JavaScript и CSS .....	36
1.8. Создание анимационных объектов.....	41
1.9. Слои .....	45
Позиционирование слоя .....	45
Свойство z-index .....	46
Свойства visibility и display .....	47
Динамическое управление слоями .....	47
Динамическое изменение цвета фона ячеек.....	49
2. Практика.....	51

Постановка задачи.....	51
2.1. Практическая работа №1. Размещение скриптов в HTML-документе.....	51
2.2. Практическая работа №2. Операторы управления, функции. Объекты ядра JavaScript.....	52
2.3. Практическая работа №3. Объекты клиентских приложений. Обработка событий. ....	55
2.4. Практическая работа №4. Объединение JavaScript и CSS.....	58
2.4. Практическая работа №5. Слои. Движущиеся элементы. ....	59
Литература .....	62

## **Введение**

В результате курса, проводимого под руководством преподавателя, студенты познакомятся с:

- технологиями и основными принципами объектно-ориентированного программирования;
- принципами создания динамических Web-документов;
- основными элементами языка;
- взаимосвязью языков скриптов и таблицей стилей для оформления Web-документов;
- организацией проверки данных введенных пользователем.

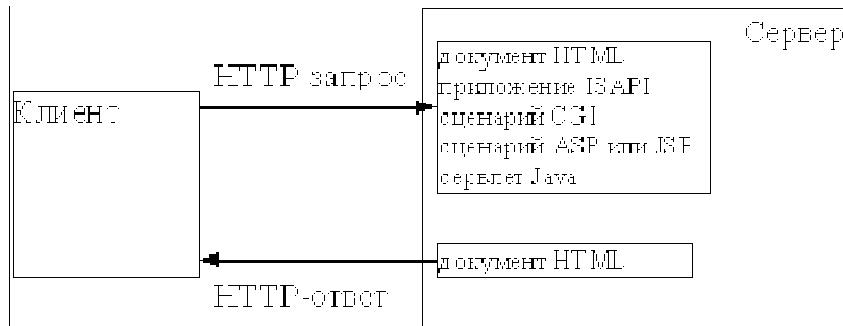
## **Цель курса**

По окончании данного курса студенты смогут:

- иметь представление об основах технологии объектно-ориентированного программирования, необходимых для Web-разработки;
- иметь представление о языке создания сценариев (то есть уметь понимать конструкции языка и интерпретировать результат);
- создавать Web-документы с динамически изменяемым содержимым;
- использовать стилевое форматирование совместно с языками сценариев для расширения возможностей оформления документов.

## 1. Обзор возможностей языка JavaScript

Взаимодействие клиента и сервера в Интернете осуществляется с помощью запросов, посылаемых клиентом серверу, и ответов сервера на запрос клиента:



Его основу составляют HTTP-сообщения, подразделяемые на:

- запрос (request) клиента к серверу;
- ответ (response) сервера клиенту.

Стандартный язык разметки HTML позволяет легко создавать статичные Web-страницы. Пользователь не может менять их содержимое, не может взаимодействовать с ними. Для того чтобы сделать страницу настоящему интерактивной, нужен еще один язык, выполняемый в контексте браузера, - скриптовый язык.

Исследования работы приложений интернета показали, что для выполнения определенных действий пользователя нет необходимости постоянно обращаться к серверу - эти действия можно реализовать на стороне клиента, если бы он позволял каким-то образом их запрограммировать. Так появился встроенный в программу просмотра Web-страниц (браузер) язык JavaScript, который расширил возможности языка разметки HTML, предоставляя разработчику возможность встраивать в документ HTML код программы, выполняющейся на клиенте.

Скриптовый язык используется для создания интерактивных страниц. Обычно он не содержит всех возможностей настоящих языков программирования, таких, например, как работа с файлами или управление графикой. Созданные с помощью скриптовых языков программы не могут выполняться самостоятельно - они работают только в контексте браузера, поддерживающего выполнения скриптовых программ. Создаваемые на скриптовых языках программы, называются сценариями или скриптами, включаются в состав Web-страниц и распознаются и обрабатываются браузером отдельно от остального HTML - кода.

Язык программирования JavaScript - объектно-ориентированный язык разработки встраиваемых приложений, выполняющихся как на стороне клиента, так и на стороне сервера.

Веб-обозреватель, работающий на компьютере-клиенте, обеспечивает среду, в которой JavaScript имеет доступ к объектам, которые представляют собой окна, меню, диалоги, текстовые области и т. д. Кроме того, обозреватель позволяет присоединить сценарии на языке JavaScript к

таким событиям, как загрузка и выгрузка страниц и графических образов, нажатие клавиш и движение мыши, выбор текста и пересылка форм. При этом программный код сценариев только реагирует на события и поэтому не нуждается в главной программе. Набор объектов, предоставляемых обозревателем, известен под названием Document Object Model (DOM).

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.

Основные области использования JavaScript при создании интерактивных HTML-страниц:

- Динамического создания содержимого страницы во время ее загрузки или уже после того, как она полностью загружена;
- Отображения диалоговых панелей и сообщений в статусной строке браузера;
- Оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер;
- Создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа (DHTML);

## 1.1. Общий обзор языка

### *Основные определения*

Любая программа оперирует некоторыми данными: именем стилевого класса, размерами элемента, цветом шрифта и прочие. JavaScript может манипулировать данными, относящимися к разным типам.

Тип данных описывает их возможные значения и набор применимых к ним операций. Типы данных бывают простыми и сложными. Сущность, относящаяся к простому типу данных может хранить только одно значение (это строковые, числовые и логические типы данных). Сущность сложного типа данных может хранить сразу несколько значений. Например – массивы. Другой пример сложного типа данных – объекты.

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

Существование программных объектов самих по себе не имеет никакого смысла. Они дадут преимущества при программировании тогда, когда можно организовать их взаимодействие.

1. Объекты – это сложные сущности, позволяющие хранить сразу несколько значений разных типов данных, они представляют собой блоки, из которых строится JavaScript. Применяются для возвращения значений и изменения состояния форм, страниц, браузера и определенных программистом переменных. Объекты можно сопоставить с существительными. Кошка, автомобиль, дом, компьютер, форма – все это существительные, они могут быть представлены как объекты.
2. Экземпляры объекта – сущности, хранящие реальные данные и созданные на основе этого объекта. То есть конкретный, реально существующий дом, находящийся по заданному адресу можно рассматривать, как экземпляр объекта типа дом.
3. Свойства – набор внутренних параметров объекта. Используются для того, чтобы различать экземпляры одного объекта – например, все экземпляры типа дом. Свойства сравнимы с прилагательными и ссылаются на уникальные для каждого экземпляра объекта особенности. Один и тот же объект может обладать многими свойствами: дом может быть большим и маленьким, синим и красным. Разные объекты могут обладать одинаковыми свойствами: дерево, так же, как и дом, может быть большим и маленьким, синим и красным... Большинство свойств объекта мы можем изменять, воздействуя на них через методы.
4. Методы - это действие или способ, при помощи которого мы можем изменять определенные свойства объекта, то есть управлять этими объектами, а также в некоторых случаях менять их содержимое.
5. События – это очень важное в программировании на JavaScript понятие. События главным образом порождаются пользователем, являются следствиями его действий. Если пользователь нажимает кнопку мыши, то происходит событие, которое называется Click. Если экранный указатель мыши движется по ссылке HTML-документа, происходит событие MouseOver. Существует несколько различных событий.
6. Оператор - это команда, инструкция для компьютера. Встретив в программе тот или иной оператор, машина четко его выполняет.
7. Функция - это определенная последовательность операторов, то есть набор команд, последовательное выполнение которых приводит к какому-то результату. Например, выполнение кем-то заданной Вами функции (процедуры) "возьми стакан, открой кран, набери в него воды и принеси мне" приведет к результату: Вы получите стакан воды из-под крана.
8. Переменная - в языках программирования переменные используются для хранения данных определенного типа, например параметров свойств объекта. Каждая переменная имеет свое имя (идентификатор) и хранит только одно значение, которое может

меняться в ходе выполнения программы. Данные могут быть разных типов: целое число, десятичная дробь, логическая константа, текстовая строка.

### *Понятие объектной модели применительно к JavaScript*

При загрузке HTML-страницы в браузер интерпретатор языка создает объекты со свойствами, определенными значениями тэгов страницы. Для правильного использования объектных моделей следует четко понимать, как браузер компонует страницы и, тем самым, создает иерархию объектов. При загрузке страницы просматриваются сверху вниз, тем самым последовательно происходит компоновка страницы и ее отображение в окне браузера. А это означает, что и объектная модель страницы также формируется последовательно, по мере ее обработки. Поэтому невозможно обратиться из сценария, расположенного ранее какой-либо формы на странице, к элементам этой формы. Всегда следует помнить о том, что браузер последовательно сверху вниз интерпретирует содержимое HTML-страницы.

Еще один аспект работы с объектами языков сценариев заключается в том, что нельзя изменить свойства объектов. Браузер обрабатывает страницу только один раз, компонуя и отображая ее. Поэтому попытка в сценарии изменить свойство отображеного элемента страницы, обречена на провал. Только повторная загрузка страницы приведет к желаемому результату.

### *Размещение операторов языка JavaScript на странице*

Встроить сценарий JavaScript в HTML-страницу можно несколькими способами.

1. Задание операторов языка внутри тэга `<script>` языка HTML.

Для внедрения в HTML-страницу сценария JavaScript в спецификацию языка HTML был введен тэг-контейнер `<script>...</script>`, внутри которого могут располагаться операторы языка JavaScript. Обычно браузеры, не поддерживающие какие-нибудь тэги HTML, просто их игнорируют, анализируя, однако, содержимое пропускаемых тэгов с точки зрения синтаксиса языка HTML, что может приводить к ошибкам при отображении страницы. Во избежание подобной ситуации следует помещать операторы языка JavaScript в контейнер комментария `<!-- ... -->`, как показано ниже

```
<script (language="javascript")>
<!--
операторы javascript
//-->
</script>
```

Параметр `language` задает используемый язык сценариев. В случае языка JavaScript его значение задавать не обязательно, так как этот язык используется браузерами по умолчанию.

Примечание:

символы `//` перед закрывающим тэгом комментария `-->` являются оператором комментария JavaScript. Он необходим для правильной работы интерпретатора.

Документ может содержать несколько тэгов `<script>`, расположенных в любом месте документа. Все они последовательно обрабатываются интерпретатором JavaScript по мере отображения частей документа в окне браузера. В связи с этим ссылка на переменную, определенную в сценарии, размещенном в конце документа, может привести к генерации ошибки интерпретатора при обращении к такой переменной из сценария в начале документа.

## 2. Задание файла с кодом JavaScript.

Тэг `<script>` имеет параметр `src`, позволяющий связать встраиваемый сценарий с внешним файлом, содержащим программный код на языке JavaScript. В качестве значения параметра задается полный или относительный URL-адрес ресурса. Задание закрывающего тэга `</script>` обязательно, независимо от того, заданы или нет операторы внутри тэга. Следующий фрагмент кода связывает документ HTML с файлом-источником, содержащим некоторый набор функций:

```
<script language="JavaScript" src="http://url/file.js">
    операторы javascript
</script>
```

Примечание:

связываемый внешний файл не должен содержать тэгов HTML и должен иметь расширение `.js`.

## 3. Использование выражений JavaScript в качестве значений параметров тэгов HTML.

Переменные и выражения JavaScript можно использовать в качестве значений параметров тэгов HTML. Например:

```
<a href="javascript: window.open('name.htm', '_self')">

</a>
```

## 4. Определение обработчика событий в тэге HTML.

## 1.2. Язык ядра JavaScript

### *Синтаксис языка*

Язык JavaScript чувствителен к регистру.

Приложение JavaScript представляет собой набор операторов языка (команд), последовательно обрабатываемых встроенным в браузер интерпретатором. Каждый оператор можно располагать в отдельной строке. В этом случае разделитель ‘;’, отделяющий один оператор от другого, не обязателен. Его используют только в случае задания нескольких операторов на одной строке. Любой оператор можно расположить в нескольких строках без всякого символа продолжения. Например, следующие два вызова функции alert эквивалентны:

```
...
alert("Подсказка");
alert(
"Подсказка"
);
...
```

Нельзя перемещать на другую строку единый строковый литерал - он должен располагаться полностью на одной строке текста программы или разбит на два строковых литерала, соединенных операцией конкатенации ‘+’:

```
...
alert("Подсказка");// правильно
alert("Под
сказка"); // не правильно
alert("Под" +
"сказка"); // правильно (но браузер выведет текст одной строкой!)
...
```

Пробельные символы в тексте программы являются незначащими, если только они не используются в строковых литералах.

В JavaScript строковые литералы можно задавать двумя равноправными способами - последовательность символов, заключенная в двойные или одинарные кавычки:

```
"Анна"
'Анна'
```

В строковых литералах можно использовать ESC-последовательности, которые начинаются с символа обратной наклонной черты, за которой следует обычный символ. Некоторые подобные комбинации трактуются как один специальный символ.

Таблица 1.

Esc-последовательности	Символ
\b	Возврат на один символ
\f	Переход на новую страницу
\n	Переход на новую строку
\r	Возврат каретки
\t	Горизонтальная табуляция Ctrl-I

\'	Апостроф
\\"	Двойные кавычки
\\"\\	Обратная наклонная черта

ESC-последовательности форматирования используются при отображении информации в диалоговых окнах, отображаемых функциями `alert()`, `prompt()` и `confirm()`, а также, если методом `document.write()` записывается содержимое элемента `pre`.

Комментарии в программе JavaScript двух видов - односторонние и многострочные:

```
// комментарий, расположенный на одной строке.  
/*  
   комментарий, расположенный  
   на нескольких строках.  
*/
```

Ссылка на объект осуществляется по имени, заданному параметром `name` тэга HTML, с использованием точечной нотации. Например, пусть в документе задана форма с двумя полями ввода:

```
<form name="form1">  
Фамилия: <input type = "text" name = "student" size = 20>  
Курс: <input type = "text" name = "course" size = 2>  
</form>
```

Для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript следует использовать ссылку `document.form.student.value`, а чтобы определить курс, на котором обучается студент, необходимо использовать ссылку `document.form.course.value`.

### *Переменные и литералы в JavaScript*

В JavaScript все переменные вводятся с помощью одного ключевого слова `var`. Синтаксическая конструкция для ввода в программе новой переменной с именем `name1` выглядит следующим образом:

```
var name1;
```

Объявленная таким образом переменная `name1` имеет значение `'undefined'` до тех пор, пока ей не будет присвоено какое-либо другое значение, которое можно присвоить и при ее объявлении:

```
var name1 = 5;  
var name1 = "новая строковая переменная";
```

JavaScript поддерживает четыре простых типа данных:

- Целый
- Вещественный
- Строковый
- Логический (булевый)

Для присваивания переменным значений основных типов применяются литералы – буквальные значения данных соответствующих типов.

### *Выражения JavaScript*

Выражение – комбинация переменных, литералов и операторов, в результате вычисления которой получается одно единственное значение. Переменные в выражениях должны быть инициализированы.

#### 1. Присваивание

Оператор присваивания (`=`) рассматривается как выражение присваивания, которое вычисляется равным выражению правой части, и в то же время он присваивает вычисленное значение выражения переменной заданной в левой части:

```
var name2=10;
```

#### 2. Арифметическое выражение

Вычисляемым значением арифметического выражения является число. Создаются с помощью арифметических операторов.

Таблица 2.

Оператор	Действие
<code>+</code>	Сложение
<code>-</code>	Вычитание
<code>*</code>	Умножение
<code>/</code>	Деление
<code>%</code>	Остаток от деления целых чисел
<code>++</code>	Увеличение значения на единицу
<code>--</code>	Уменьшение значения на единицу

#### 3. Логическое выражение

Вычисляемым значением логического выражения может быть `true` или `false`. Для создания используются операторы сравнения или логические операторы, применяемые к переменным любого типа.

Таблица 3.

Операторы сравнения	Значение	Логические Операторы	Значение
<code>==</code>	Равно	<code>&amp;&amp;</code>	логическое И
<code>!=</code>	Не равно	<code>  </code>	логическое ИЛИ
<code>&gt;=</code>	Больше или равно	<code>!</code>	логическое НЕ

$\leq$	Меньше или равно		
$>$	Строго больше		
$<$	Строго меньше		

#### 4. Строковые выражения

Вычисляемым значением строкового выражения является число. В JavaScript существует только один строковый оператор – оператор конкатенации (сложения) строк:

```
string1 = "Моя " + "строка"
```

### 1.3. Управляющие конструкции языка JavaScript

#### *Операторы JavaScript*

Операторы служат для управления потоком команд в JavaScript. Блоки операторов должны быть заключены в фигурные скобки.

##### 1. Операторы выбора

- условный оператор if

Эта управляющая структура используется, когда необходимо выполнить некий программный код в зависимости от определенных условий. Также предусмотрена конструкция if-else (если-тогда-иначе).

```
if (условие_1)
{
    оператор_1;      // эти операторы выполняются, если условие_1
верно
    оператор_2;
}
else
{
    оператор_3; // эти операторы выполняются, если условие_1 ложно
    оператор_4;
}
```

Условие для проверки (вопрос компьютеру) записывается сразу после слова if в круглых скобках. После этого в фигурных скобках пишется то, что будет предприниматься в случае выполнения условия. Далее else и снова в фигурных скобках то, что выполнится в случае, если условие не сработает. Количество различных действий между фигурными скобками неограниченно, фактически можно выполнить две различные программы. При сравнении можно использовать логические выражения. Например:

```
<script language="JavaScript">
var x = 5;
var y = 10;
if (x>y) {
    alert('x - максимальное число')
}
else
{
    alert('y - максимальное число')
}
</script>
```

- оператор выбора switch

Это фактически несколько условных операторов, объединенных в одном. В данном операторе вычисляется одно выражение и сравнивается со значениями, заданными в блоках case. В случае совпадения выполняются операторы соответствующего блока case.

```
switch (выражение) {
    case значение1:
        оператор_1;
        break;
    case значение2:
        оператор_2;
        break;
    ....
    default:
        оператор;
}
```

Если значение выражения не равняется ни одному из значений, заданных в блоках case, то вычисляется группа операторов блока default, если этот блок задан, иначе происходит выход из оператора switch. Необязательный оператор break, задаваемый в блоках case, выполняет безусловный выход из оператора switch.

## 2. Операторы цикла

Оператор цикла повторно выполняет последовательность операторов JavaScript, определенных в его теле, пока не выполниться некоторое заданное условие.

- цикл for (цикл со счетчиком)

```
for (i=1; i<10; i++){
    <тело цикла>
}
```

Первый параметр (i=1) определяет счетчик и указывает его начальное значение. Этот параметр называется начальным выражением, поскольку в нем задается начальное значение счетчика (начальное значение в данном

случае равно единице). Это выражение инициализации выполняется самым первым и всего один раз.

Второй параметр ( $i < 10$ ) - это условие, которое должно быть истинным, чтобы цикл выполнялся, как только условие цикла становится ложным, работа цикла завершается. Он называется условием цикла. Проверка условия цикла осуществляется на каждом шаге; если условие истинно, то выполняется тело цикла (операторы в теле цикла). Цикл в данном случае выполнится только девять раз так как задано условие  $i < 10$ .

Третий параметр ( $i++$ ) - это оператор, который выполняется при каждом последовательном прохождении цикла. Он называется выражением инкремента, поскольку в нем задается приращение счетчика (приращение счетчика в данном случае равно единице). Пример автоматической прорисовки нескольких линий с помощью цикла for.

```
<script language="JavaScript" type="text/JavaScript">
for (var i=1; i<10; i++){
    document.write("<hr align='center' width='100'>");
}
</script>
```

- цикл while (цикл с предусловием)  
while (условие){  
 {  
 <тело цикла>  
 }

Пока значение условия - true (истинно), выполняется тело цикла. Тело цикла может быть представлено простым или составным оператором.

Оператор while содержит в скобках все необходимые параметры условия цикла (логическое выражение). После определения всех параметров цикла вводится открывающая фигурная скобка, символизирующая начало тела цикла. Закрывающая фигурная скобка вводится в конце тела цикла. Все операторы, введенные в скобках, выполняются при каждом прохождении цикла.

```
<script language="JavaScript">
var i=1;
while(i<=10){
    document.write('число=' + i + '<br>');
    i=i+2;
}
</script>
```

- прерывание и перезапуск цикла

Оператор прерывания break позволяет прервать выполнение цикла и перейти к следующему за ним выражению:

```
a = 10;
i = 1;
while (a<100){
```

```
a = a * i;  
if (i>4) break;  
++i;  
}
```

Если значение *i* превысит 4, то прерывается выполнение цикла.

Оператор перезапуска *continue* позволяет перезапустить цикл, т.е. оставить невыполненным все последующие выражения, входящие в тело цикла, и запустить выполнение цикла с самого начала.

```
a = 10;  
i = 1;  
while (a<100){  
    ++i;  
    if (i>2 && i<11) continue;  
    a = a * i;  
}
```

### *Создание и вызов функций в JavaScript*

В JavaScript функцией называется именованная часть программного кода, которая выполняется только при обращении к ней посредством указания ее имени. Функции создаются с помощью ключевого слова *function*. Обычно функции располагают в секции *<head>*. Такое расположение функций в HTML-документе гарантирует их полную загрузку до того момента, когда их можно будет вызвать из секции *<body>*.

После названия функции (*func\_name*) ставятся двойные круглые скобки, программный код при этом заключается в фигурные скобки:

```
<script language="JavaScript">  
function func_name()  
{  
    программный код функции (тело функции)  
}  
</script>
```

Для того, чтобы вызвать функцию в нужном месте, необходимо просто указать ее имя в тексте:

```
<script language="JavaScript">  
func_name();  
</script>
```

Второй вариант вызова функции непосредственно в HTML теге:

```
<a href="javascript:func_name()">Текст ссылки</a>
```

Ниже приведен код страницы HTML, после загрузки которой каждые три секунды будет появляться сообщение, генерируемое вызовом функции *myMessage()*:

```
<script>
```

```

function myMessage()
{
    alert("My Message")
}
</script>
<body onload='setTimeout ("myMessage()",3000)'>
<p>Каждые три секунды будет появляться сообщение</p>
</body>

```

Метод `setTimeout()` запускает выполнение кода JavaScript, задаваемого первым строковым параметром, через определенный промежуток времени после выполнения метода.

Интервал задается в миллисекундах (1000 соответствует 1 секунде).

#### **1.4. Стандартные объекты и функции ядра JavaScript**

##### *Объект Array*

Массив - упорядоченный набор однородных данных, к элементам которого можно обращаться по имени и индексу. Язык JavaScript не имеет встроенного типа данных для создания массивов, поэтому для решения используется объект `Array` и его методы.

Для создания объекта `Array` вызывается оператор `new` и конструктор массива - системная функция (ее имя совпадает с именем объекта), инициализирующая элементы массива:

`m=new Array();`

Заполнение массива происходит позже. Например:

```

<script language="JavaScript">
//создание нового массива
m=new Array();
//заполнение массива
m[0]=1;
m[1]=2;
m[2]=4;
m[3]=56;
</script>

```

В приведенном выше примере с помощью команды `new` создается массив `m`, а затем происходит его заполнение - каждому элементу присваивается определенное значение.

`m=new Array(1,2,4,56);`

Вызывается команда `new` и сразу задаются значения всех элементов массива.

```

<script language="JavaScript">
//создание нового массива и его заполнение

```

```
m=new Array(1,2,4,56)
</script>
```

Объявление строковых массивов проводится тем же способом, что и объявление числовых массивов.

Таблица 4.

Методы объекта Array	Действие
join()	Объединяет все элементы массива в одну строку с указанием разделителя.
reverse()	Изменяет порядок элементов в массиве - первый элемент становится последним, последний - первым
sort()	Выполняет сортировку элементов массива
split()	Разделяет строку на составные части
concat()	Объединяет два массива в один
slice()	Выделяет часть массивы
toString()	Возвращает строку - результат конкатенации всех элементов массива. Элементы массива в строке разделены запятой.
Свойство length	Возвращает длину массива (число элементов в нем).

Пусть определены два массива:

```
array1 = new Array("Первый","Второй","Третий");
```

```
array2 = new Array("Один","Два","Три");
```

Тогда метод join() первого массива array1.join() возвратит строку:  
"Первый,Второй,Третий"

Метод sort() первого массива array1.sort() упорядочит элементы массива array1 (переставив их местами непосредственно в самом массиве array1) в алфавитном порядке:

```
array1[0] = "Второй";
array1[1] = "Первый";
array1[2] = "Третий";
```

Поскольку некоторые методы массива возвращают массив, то к нему можно сразу же применить какой-либо метод, продолжив "точечную" нотацию. Например, array1.concat(array2).sort() объединит два массива в один новый и отсортирует его.

### *Объект Date*

Используется для представления дат в программах JavaScript. Время храниться в виде числа миллисекунд, прошедших от 1 января 1970 года.

Данный объект создается также, как и любой объект в JavaScript – с помощью оператора new и конструктора, в данном случае Date():

```
date1 = new Date(); // значением переменной date1 будет текущая дата
```

Параметром конструктора может быть строка, в которой записана нужная дата:

```
date1 = new Date("january 14, 2000, 12:00:00");
```

Можно задать список параметров:

```
date1 = new Date(2000, 1, 14, 12, 0, 0);
```

### *Объект Math*

В свойствах данного объекта хранятся основные математические константы, а его методы вычисляют основные математические функции. При обращении к данному объекту, создавать его не надо, но необходимо явно указывать его имя Math. Например:

```
p = Math.PI; // хранится значение числа пи.
```

### *Объект String*

Можно явно создавать строковый объект, используя оператор new и конструктор:

```
myString = new String("Hello!");
```

Данный объект имеет единственное свойство length, хранящее длину строки, содержащейся в строковом объекте, и два типа методов: одни непосредственно влияют на содержание самой строки, вторые возвращают отформатированный HTML-вариант строки.

### *Стандартные функции верхнего уровня*

В JavaScript существуют несколько функций, для вызова которых не надо создавать никакого объекта, она находятся вне иерархии объектов.

Функция parseFloat(parameter) анализирует значение переданного ей строкового параметра на соответствие представлению вещественного числа.

Функция parseInt(parameter, base) пытается возвратить целое число по основанию, заданному вторым параметром.

Эти функции полезны при анализе введенных пользователем данных в полях формы до их передачи на сервер.

Функции Number(object) и String(object) преобразуют объект, заданный в качестве его параметра в число или строку.

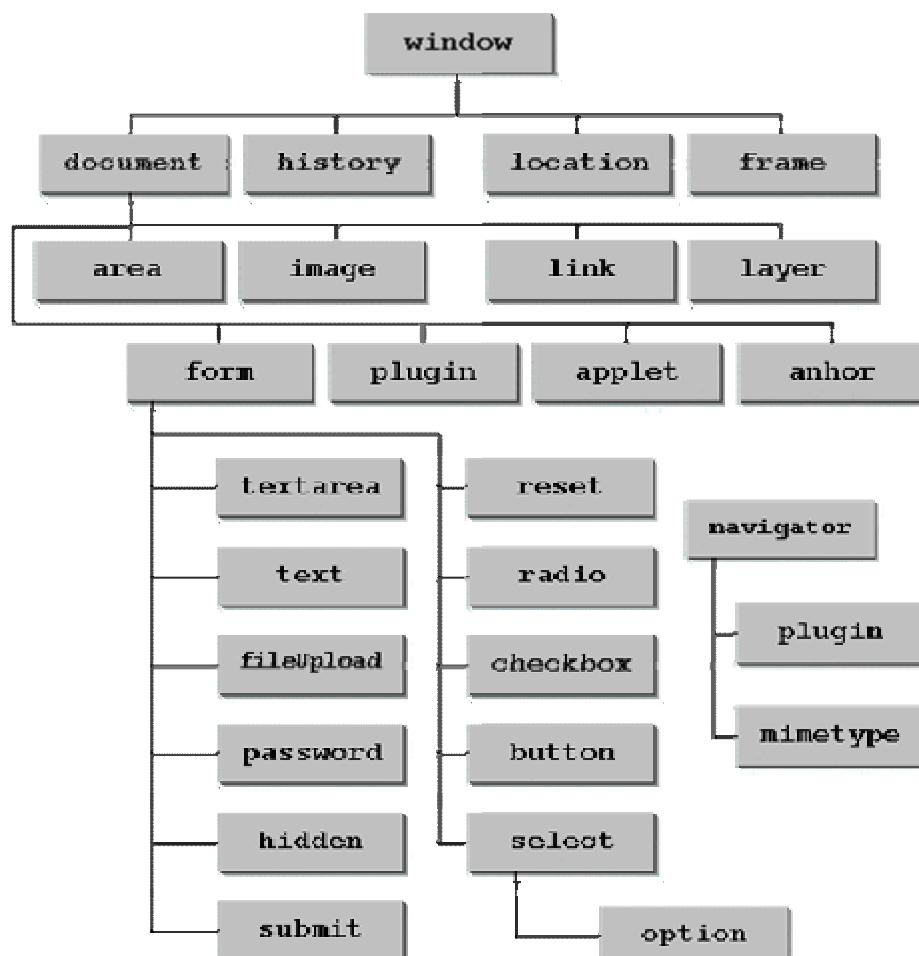
## 1.5. Объекты клиента

При интерпритации страницы HTML браузером создаются объекты JavaScript, свойства которых представляют значения параметров тэгов языка HTML.

### Иерархия объектов

Созданные объекты существуют в виде иерархической структуры, отражающей структуру самой HTML-страницы. На верхнем уровне расположен объект window, представляющий собой активное окно браузера. Далее вниз по иерархической лестнице следуют объекты frame, document, location и history и т.д.

Значения свойств объектов отражают значения соответствующих параметров тэгов страницы или установленных системных параметров. На рисунке показана структура объектов клиента (браузера).



Особняком стоит объект navigator с двумя дочерними (подчиненными) объектами. Он относится к самому браузеру, и его

свойства позволяют определить характеристики программы просмотра. Каждая страница в добавление к объекту `navigator` обязательно имеет еще четыре объекта:

- `window` — объект верхнего уровня, свойства которого применяются ко всему окну, в котором отображается документ;
- `document` — свойства которого определяются содержимым самого документа: связи, цвет фона, формы и т. д.;
- `location` — свойства которого связаны с url-адресом отображаемого документа;
- `history` — представляет адреса ранее загружавшихся HTML-страниц.

Кроме указанных объектов страница может иметь дополнительные объекты, зависящие от ее содержимого, которые являются дочерними объектами объекта `document`. Если на странице расположена форма, то все ее элементы являются дочерними объектами этой формы. Для задания точного имени объекта используется точечная нотация с полным указанием всей цепочки наследования объекта. Наиболее общий объект высшего уровня находится слева в выражении, и слева направо происходит переход к более частным объектам, являющимся при этом наследниками высших в иерархии объектов.

Кроме этих классов объектов пользователь может создавать и свои собственные. Но обычно большинство программ используют эту систему классов и не создают новых.

### *Объект navigator*

Этот объект применяется для получения информации о версиях.

Синтаксис:

`navigator.name_properties`

Методы и события, догадаться не определены для этого объекта. Да и свойства только для чтения, так как ресурс с информацией о версии недоступен для редактирования.

Свойства

- `appCodeName` - кодовое имя браузера;
- `appName` - название браузера;
- `appVersion` - информация о версии браузера;
- `userAgent` - кодовое имя и версия браузера;

Ниже приведен пример использования объекта `navigator`.

```
<html>
<head>
<title> navigator </title>
<script language="javascript">
var firstn = window.prompt("Введите ваше имя: ","ваше имя")
function welcome(){
```

```

var appname=navigator.appName
var appver=navigator.appVersion
window.alert("Привет, " + firstn + ". Вы используете " +
appname + ". Версия " + appver + ". Спасибо за визит.");
}
function writename(){
document.write(firstn + ".");
}
</script>
</head>
<body onLoad="welcome()">
Добро пожаловать,
<script language="JavaScript">
writename();
</script>
</body>
</html>

```

### *Объект window*

Объект `window` создается автоматически при запуске браузера, так как для отображения документа необходимо окно. Одно из назначений объекта окна - это создание нового окна. Новое окно браузера создается с помощью метода `window.open()`. Метод `window.open()` имеет ряд дополнительных аргументов, которые позволяют задать местоположение окна, его размер и тип, а также указывают, должно ли окно иметь полосы прокрутки, полосу команд и т. п. Помимо этого можно задавать и имя окна.

В общем виде данный метод можно представить следующим образом:

```
window.open('url', 'name', 'parameters')
```

Рассмотрим синтаксис более подробно:

- Первый параметр метода `window.open()` - это url документа, загружаемого в окне. Если его не заполнить, то окно останется пустым.
- Второй параметр определяет название окна (`name`). Это имя может использоваться для обращения к созданному окну.
- Третий параметр представляет список необязательных опций, разделенных запятой. С их помощью Вы определяете вид нового окна: наличие в нем панелей инструментов, строки состояния и других элементов.

Приведем таблицу с описанием параметров нового окна, задаваемого третьим параметром (`parameters`) метода `open()`.

Таблица 5.

Параметр	Значение		Описание
fullscreen	yes 1	no 0	указывает, показывается ли новое окно на полный экран или как обычное окно. По умолчанию показывается обычное окно
channelmode	yes 1	no 0	позволяет указать, отображается ли полоса каналов
toolbar	yes 1	no 0	позволяет указать, отображается ли полоса кнопок
location	yes 1	no 0	позволяет указать, отображается ли полоса для ввода адреса
directories	yes 1	no 0	позволяет указать, отображается ли полоса кнопок для выбора каталогов
status	yes 1	no 0	позволяет указать, отображается ли полоса статуса
menubar	yes 1	no 0	позволяет указать, отображается ли полоса меню
scrollbars	yes 1	no 0	задает отображение горизонтальной и вертикальной полос прокрутки
resizable	yes 1	no 0	позволяет указать, может ли окно изменять свой размер
width	yes 1	no 0	задает ширину окна в пикселях. Минимальное значение - 100
height	yes 1	no 0	задает высоту окна в пикселях. Минимальное значение - 100
top	yes 1	no 0	задает вертикальную координату левого верхнего угла окна
left	yes 1	no 0	задает горизонтальную координату левого верхнего угла окна

Объект window использует три метода отображения сообщений:

- метод `prompt()` – выводит диалоговое окно с полем ввода, куда пользователь может ввести информацию
- метод `alert()` – выводит на экран окно - сообщение с кнопкой OK и определенным программистом текстом
- метод `confirm()` – выводит диалоговое окно с кнопками OK и Cancel. Дает возможность пользователю продолжить или отменить предложенную операцию.

Сообщение, которое вы хотите вывести на экран, набирается в кавычках внутри круглых скобок.

Данный скрипт запрашивает имя посетителя и выдает приветствие с введенным именем.

```
<script language="JavaScript">
name=window.prompt ("Введите, пожалуйста, свое имя", "Ваше имя");
window.alert ("Вас зовут, " + name);
</script>
```

В этом фрагменте кода метод `prompt` имеет следующие параметры: текст запроса и значение, заполняющее поле ввода по умолчанию; переменная `name` - имя переменной, куда сохраняется введенная информация (имя может быть любым).

### *Объект document*

Объект `document` имеет дело прежде всего с телом HTML-страницы. Он имеет несколько дочерних объектов (коллекций): `all`, `images`, `link`, `anchor` и `form`. Пользуясь объектной моделью построения документа можно, например, обратиться к любой картинке на странице через следующий синтаксис:

```
document.images.name.src
```

Для `document` не существует никаких событий. Некоторые свойства и методы перечислены в таблице, из методов наиболее употребимы `write` и `writeln`.

Таблица 6.

Свойства	Назначение
<code>bgColor</code>	Устанавливает цвет фона текущего документа. Этот цвет может иметь шестнадцатеричное представление <code>#rrggbb</code> или соответствующее название. Синтаксис: <code>document.bgColor="#e7e6d8"</code>
<code>fgColor</code>	Устанавливает цвет текста документа. Аналогичен по функциям свойству <code>bgColor</code>
<code>referrer</code>	Указывает url документа, на который ссылается пользователь в настоящее время. Например, если кто-то обратился по адресу: <code>http://www.nm.org/welcome.htm</code> с сервера

	http://www.someplace.com, то свойством referrer будет: http://www.someplace.com, если это свойство было в странице вышеупомянутого расположения; в противном случае оно обращается в null
location	Соответствует адресу url текущего документа

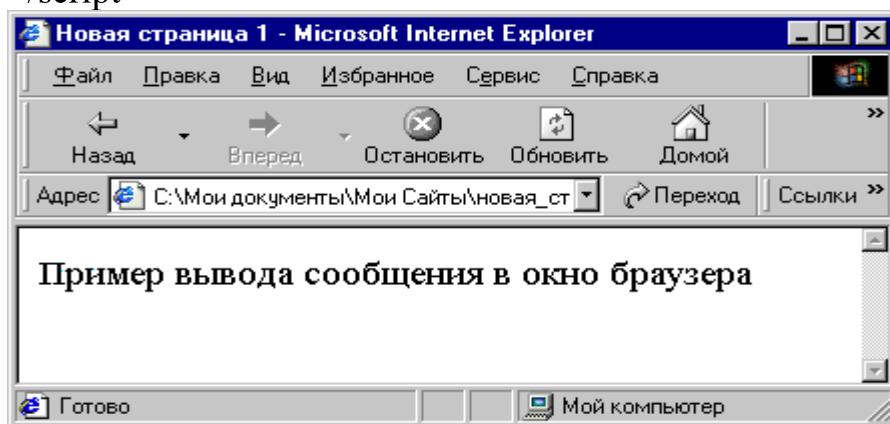
Таблица 7.

Методы	Назначение
write() writeln()	Записывает HTML-текст в текущий документ и должен вызываться, когда документ открывается для записи. Синтаксис: document.write('somestring'), где somestring может быть одной строкой, переменной или же несколькими связанными строками в формате HTML, которые выводятся на экран
lastModified()	Показывает дату последней модификации документа: date1 = document.lastModified
open()	Открывает документ для записи дополнительных строк в формате HTML: document.open()
close()	Закрывает документ, который вызывался методом document.open(): document.close().

Методы write() / writeln().

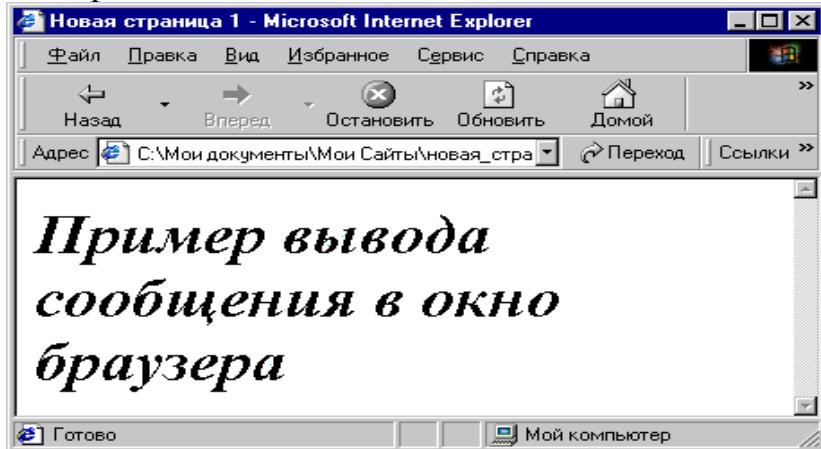
Вызов метода document.write() с указанием определенных параметров приводит к отображению текста в окне браузера. В качестве параметра при вызове метода document.write() мы указываем строку, которую хотели бы увидеть на экране.

```
<script language="JavaScript">
document.write('Пример вывода сообщения в окно браузера')
</script>
```



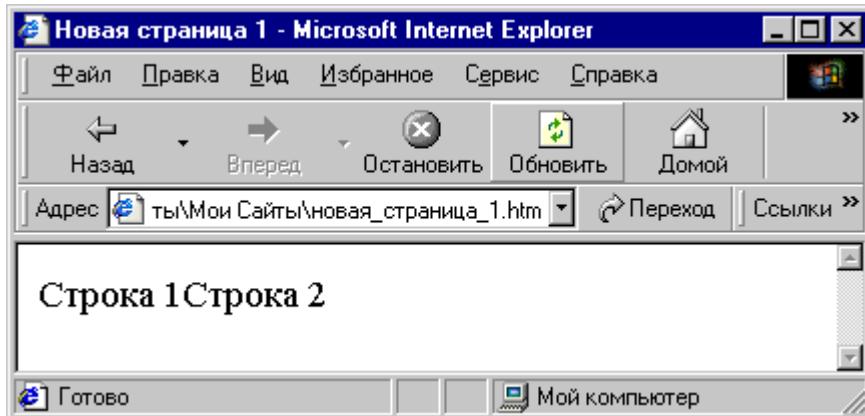
Выводимая строка может содержать и тэги языка HTML. В этом случае браузер выведет данную строку точно так же, как если бы она была размещена непосредственно в HTML документе.

```
<script language="JavaScript">
    document.write('<h1><b><i>Пример вывода сообщения в окно
браузера</i></b></h1>')
</script>
```



При написании скрипта, содержащего несколько команд `document.write()` подряд, при выводе в браузер текст окажется на одной строке

```
<script language="JavaScript">
    document.write('Строка 1');
    document.write('Строка 2');
</script>
```



Для размещения каждого куска текста в новом абзаце можно использовать 2 способа:

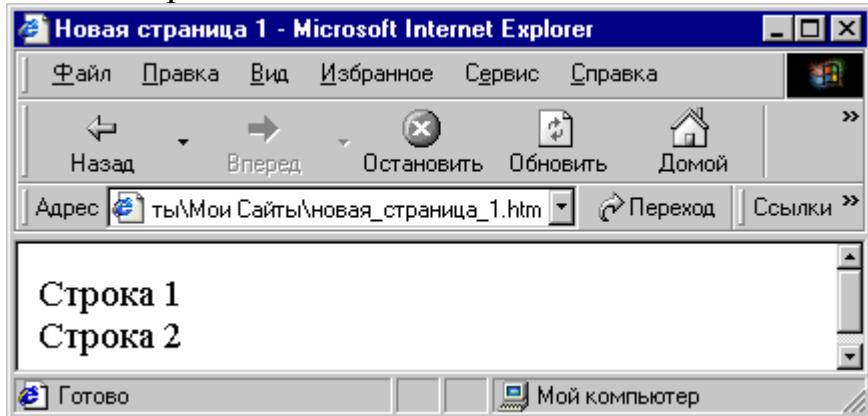
1. либо тег `<p>`, либо тег `<br>`, который включается в состав выводимой строки;
2. используя метод `document.writeln()`.

Следующие примеры приведут к одинаковому результату:

```
<script language="JavaScript">
    document.write('Строка 1<br>');
    document.write('Строка 2<br>');
</script>
```

и

```
<script language="JavaScript">  
document.writeln('Строка 1') ;  
document.writeln('Строка 2') ;  
</script>
```



### *Объект location*

Объект *location* содержит информацию о местонахождении текущего документа, т.е. его интернет-адрес. Его также можно использовать для перехода на другой документ и перезагрузки текущего документа.

Таблица 8.

Свойство	Описание
hash	Имя "якоря" в интернет-адресе документа, если оно есть.
host	Имя компьютера в сети интернет вместе с номером порта, если он указан.
hostname	Имя компьютера в сети Интернет.
href	Полный интернет-адрес документа.
pathname	Путь и имя файла, если они есть.
port	Номер порта. Если не указан, возвращает номер 80 - стандартный порт, через который работает протокол http.
protocol	Идентификатор протокола. Если не указан, возвращается "http:".
search	Строка параметров, если она есть.

Таблица 9.

Метод	Описание
assign(url)	Загружает документ, адрес которого передан в качестве параметра. Поддерживает только IE начиная с 4.0
reload()	Перезагружает документ с Web-сервера.
replace(url)	Загружает документ, адрес которого передан в качестве параметра, и заменяет в списке истории Web-обозревателя адрес предыдущего документа адресом нового.

Пользуясь объектом location, можно загрузить другой документ на место текущего. Для этого просто необходимо присвоить значение нового интернет-адреса свойству href.

```
document.location.href = "http://www.---.ru";
```

Если вы хотите полностью заменить текущий документ, чтобы даже адрес его не появлялся в списке истории, воспользуйтесь методом replace:

```
document.location.replace("http://www.--.ru");
```

### *Объект form*

Каждая форма в документе, определенная тегом <form>, создает объект form, порождаемый объектом document. Ссылка на этот объект осуществляется с помощью переменной, определенной в атрибуте name тега <form>. В документе может быть несколько форм, поэтому для удобства ссылок и обработки в объект document введено свойство-массив forms, в котором содержатся ссылки на все формы документа. Ссылка на первую форму задается как document.forms[0], на вторую - document.forms[1] и т.д. Вместо индекса в массиве forms можно указывать имя формы. Например, если в документе присутствует единственная форма со значением атрибута name=form1, то любой из следующих операторов JavaScript содержит ссылку на эту форму:

```
document.forms[0];
document.forms["form1"];
document.form1;
```

Последний оператор возможен в силу того, что объект document порождает объект form (как и все остальные объекты, соответствующие элементам HTML страницы) и ссылку на него можно осуществлять по обычным правилам наследования языка JavaScript.

Все элементы формы порождают соответствующие объекты, подчиненные объекту родительской формы. Таким образом, для ссылки на объект text (с параметром name = text1) формы form1 можно пользоваться любым из нижеприведенных операторов:

```
document.forms[0].text1;
document.forms["form1"].text1;
```

```
document.form1.text1;
```

Кроме имени элементы формы, имеют свойство value, значение которого определяется смыслом атрибута value элемента формы. Например, для элементов text и textarea значением этого свойства будет строка содержимого полей ввода этих элементов; для кнопки подтверждения - надпись на кнопке и т.д. Обратиться к свойству value можно по тому же принципу, например:

```
document.form1.text1.value
```

## 1.6. Обработка событий

Использование языка JavaScript при обработке событий значительно расширило возможности языка HTML. Чаще всего программы создаются для обработки информации, вводимой пользователем в поля форм. Возможности управления элементами форм обеспечиваются главным образом за счет функций обработки событий, которые могут быть заданы для всех элементов формы. События делятся на несколько категорий:

- события, связанные с документами (события документа) - загрузка и выгрузка документов;
- события, связанные с гиперсвязью (события гиперсвязи) - помещение указателя мыши на гиперсвязь и активизация гиперсвязи;
- события, связанные с формой (события формы) –
  - щелчки мыши на кнопках;
  - получение и потеря фокуса ввода и изменение содержимого полей ввода, областей текста и списков;
  - выделение текста в полях ввода и областях текста;

События, связанные с документами, возникают при загрузке и выгрузке документа, в то время как события гиперсвязей возникают при их активизации или при помещении на них указателя мыши. Чтобы обеспечить перехват события, необходимо написать функцию-обработчик события. В качестве обработчиков событий могут быть заданы целые функции языка JavaScript или только группы из одного или нескольких операторов. В таблице перечислены имена событий и условия их возникновения:

Таблица 10.

Имя события	Атрибут HTML	Условие возникновения события
Blur	onBlur	Потеря фокуса ввода элементом формы
Change	onChange	Изменение содержимого поля ввода или области текста, либо выбор нового элемента списка

Click	onClick	Щелчок мыши на элементе формы или гиперсвязи
Focus	onFocus	Получение фокуса ввода элементом формы
Load	onLoad	Завершение загрузки документа
Unload	onUnload	Выгрузка текущего документа и начало загрузки нового
MouseOver	onMouseOver	Помещение указателя мыши на гиперсвязь
MouseOut	onMouseOut	Помещение указателя мыши не на гиперсвязь
Select	onSelect	Выделение текста в поле ввода или области текста
Submit	onSubmit	Передача данных формы

### *Атрибут onClick*

Атрибут onClick может использоваться в следующих тегах HTML:

- <a href="url" onClick="function()"><...></a>
- <input type="checkbox" onClick="function()">
- <input type="radio" onClick="function()">
- <input type="reset" onClick="function()">
- <input type="submit" onClick="function()">
- <input type="button" onClick="function()">

Операторы языка JavaScript, заданные в атрибуте onClick, выполняются при щелчке мыши на таких объектах как гиперсвязь, кнопка перезагрузки формы или контрольный переключатель. Для контрольных переключателей и селекторных кнопок событие Click возникает не только при выборе элемента, но и при разблокировании.

Разберем пример использования атрибута onClick для кнопок, определенных тегами

<input type="button"> в контейнере <form> . . . </form>:

```

...
<script language="JavaScript">
function but1() {
    alert("Вы нажали первую кнопку");
}
function but2() {
    alert("Вы нажали вторую кнопку");
}
</script>
...
<form>
<input type="button" value="Первая кнопка" onClick="but1()">

```

```
<input type="button" value="Вторая кнопка" onClick="but2()">
</form>
...

```

### *Работа с меню*

Список в форме задается с помощью объекта select, обработка событий выполняется с помощью следующих параметров:  
onChange - вызывается при изменении выбора;  
onBlur - вызывается при снятии фокуса с объекта;  
onFocus - вызывается при перемещении фокуса на объект.

Рассмотрим следующий пример:

```
...
<script language="JavaScript">
function selectBlur()
{
    document.myForm7.myText.value="Вы нажали поле вне списка ";
}
function selectFocus()
{
    document.myForm7.myText.value="Вы нажали ту же кнопку ";
}
function selectChange()
{
    document.myForm7.myText.value="Вы нажали другую кнопку ";
}
</script>
...
<form name="myForm7">
<input type="text" name="myText" size=40 value="Город"><br>
<select      name="script"      multiple      onBlur="selectBlur()"
onFocus="selectFocus()" onChange="selectChange()">
    <option value="town1" selected>Париж
    <option value="town2">Лондон
    <option value="town3">Рим
    <option value="town4">Берлин
</select>
</form>
...

```

### *Управление логикой программного кода при помощи событий*

В объектно-ориентированном программировании нет единой структуры управления работой программы. Есть независимые друг от

друга объекты. Когда пользователь щелкает, например, по ссылке на экране, браузер передает событие Click объекту, тега <a>. Для события “щелчок мыши” в этом объекте предусмотрен стандартный обработчик — он загружает в окно новый документ.

Давайте попробуем “перехватить” это событие:

```
<a href="page1.htm" onClick="alert('Хода нет?')">документ page1</a>
```

Если щелкнуть по ссылке, на экране возникнет надпись “Хода нет?”. Событие перехвачено, но, при закрытии окна alert, видим, что браузер по-прежнему грузит документ page1.htm. При помощи атрибута onClick мы установили в объекте, “отвод” на собственный обработчик. Но когда скрипт нашего обработчика выполнен, управление возвращается к стандартному обработчику, и это вызывает загрузку документа page1.htm.

Отключение стандартной обработки кодируется так:

```
<a href=page1.htm onClick="alert('Хода нет!');return false">документ page1</a>
```

Оператор return указывает возвращаемое функцией значение. Если ее операнд true, то документ загружается, если false, нет.

Подтверждение активизации гиперсвязи.

Аналогичный пример управления логикой программного кода при помощи событий рассмотрен и в следующем примере. Гиперссылка обычно всегда срабатывает по клику мыши, но иногда нужно, чтобы пользователь был уверен, что хочет перейти по ссылке в следующий документ. Для этого существует метод confirm(), который отображает на экране окно сообщения с кнопками "Ok" и "Cancel". Для перехвата события в теге <a href= ... > мы применим событие onClick. Рассмотрите пример подтверждения активизации гиперсвязи:

```
<a href="form.htm" onClick="return confirm('Вы действительно хотите  
перейти по ссылке?')">Подтверждение активизации гиперсвязи</a>
```

### *Определение событий формы*

Объект form имеет два обработчика событий: onSubmit и onReset. В эти обработчики событий, задаваемые в пределах дескриптора <form>, добавляется группа операторов JavaScript или функция, управляющая формой.

Если вы добавите оператор (или функцию) в обработчик onSubmit, то он (или она) вызывается до отправки данных в сценарий CGI. Для того чтобы отменить отправку данных на обработку сценарием CGI, обработчик событий onSubmit должен возвратить значение false. Если же он возвращает значение true, то данные отправляются на сервер. В некоторых случаях необходимо добавить в форму кнопку reset, запускающую обработчик событий onReset.

Для формы одним из важных действий на странице является проверка правильности заполнения полей пользователем на машине клиента до

пересылки их на сервер. В следующем примере разъясняется, как выполнять эту процедуру.

Рассмотрим скрипт, который будет проверять правильность заполнения формы. Необходимо проверить нет ли пустых строк и правильно ли введен e-mail:

```
<html>
<head>
<title>пример формы</title>
<script language="JavaScript">
    function doSend(){
        var v=document.user.e.value.indexOf("@",1)
        if(document.user.f.value==""){
            alert('Вы должны заполнить поле ФИО')
            document.user.f.focus()
        }
        if(document.user.a.value==""){
            alert('Вы должны заполнить поле адреса')
            document.user.a.focus()
        }
        if(document.user.e.value==""){
            alert('Вы должны заполнить поле e-mail')
            document.user.e.focus()
        }
        if(v===-1){
            alert('Адрес e-mail указан неверно')
            document.user.e.select()
            document.user.e.focus()
        }
        else
            document.user.submit()
    }
    </script>
</head>
<body>
<p align="center"><font size=6>Данные о пользователе</font>
<form name="user">
<b>Пожалуйста, укажите данные о себе:</b>
<br>
ФИО<input type="text" name="f" size="30"><br>
Адрес<input type="text" name="a" size="35"><br>
e-mai<input type="text" name="e" size="30"><br>
<input type="button" value="Послать" onClick="doSend()">
<input type="reset" value="Отменить">
</form>
```

```
</p>
</body>
</html>
```

### *Вставка звука*

Если вам необходимо озвучить страницу, вот простейшая инструкция:

```
<bgsound src="music/gimn.mid" loop=infinite>
```

С помощью JavaScript можно разнообразить страницы сайта.

Пример скрипта проигрывания музыки при наведении на заголовок текста:

```
...
<script>
function playHome() {
document.all.sound.src = "music/file.mid"
}
</script>
...
<bgsound id=sound>
<h1 onMouseover=playHome()>Заголовок с музыкой</h1>
...
...
```

## **1.7. DHTML**

DHTML (динамический HTML) – это набор средств, которые позволяют создавать более интерактивные Web-страницы без увеличения загрузки сервера. Другими словами, определенные действия посетителя ведут к изменениям внешнего вида и содержания страницы без обращения к серверу.

DHTML построен на объектной модели документа (Document Object Model, DOM), которая расширяет традиционный статический HTML-документ. DOM обеспечивает динамический доступ к содержимому документа, его структуре и стилям. В DOM каждый элемент Web-страницы является объектом, который можно изменять. DOM не определяет новых тэгов и атрибутов, а просто обеспечивает возможность программного управления всеми тэгами, атрибутами и каскадными таблицами стилей (CSS).

Каждой гиперссылке, заголовку или текстовому параграфу можно присвоить имя, атрибуты стиля или цвета текста и указать это имя в сценарии, имеющемся на данной странице. Сценарий может быть написан на любом существующем скриптовом языке, но здесь подразумевается использование языка JavaScript. Элементы страницы впоследствии могут изменяться в результате определенного события, например при наведении курсора мыши, при щелчке, нажатии клавиши на клавиатуре либо после

истечения заданного промежутка времени. Изменяться может не только стиль и цвет текста, но и весь объект, текст или рисунок.

## *Объединение JavaScript и CSS*

### 1. Пример изменения цвета текста.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 style="color:red">Добро пожаловать на нашу страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

Данный пример применения CSS позволяет сделать заголовок красного цвета. Допустим, вы хотите, чтобы текст заголовка только тогда становился красным, когда пользователь наводит на него курсор. Этого можно добиться с помощью CSS и JavaScript.

Шаг 1. Удаление существующей информации о стиле

Это действие может показаться вам шагом назад, но оно действительно необходимо:

```
<html>
<head>
    <title>Простая страница</title>
</head>
<body>
<h1>Добро пожаловать на нашу страницу! </h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

Шаг 2. Добавление идентификатора

Поскольку вам нужно как-то обращаться к элементу, с которым будут производиться манипуляции, необходимо в тэг <h1> добавить атрибут id - это краткое обозначение, позволяющее указать нужный элемент:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 id="head1">Добро пожаловать на нашу страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

### Шаг 3. Добавление обработчика событий

Следующий шаг — добавление обработчика событий. Этому действию соответствует событие onMouseover. Также следует указать имя функции, которая будет вызываться при выполнении события:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 id="head1" onMouseover ="colorchange ()">Добро пожаловать на
нашу
страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>
```

### Шаг 4. Написание сценария JavaScript

Вам потребуется единственная строка, состоящая из следующих частей:

- имя объекта на странице, с которым должен выполняться ваш сценарий - в данном случае head1;
- применяемый аспект JavaScript - в данном случае style;
- атрибут стиля, который будет изменяться - color;
- новое значение, принимаемое атрибутом стиля - red.

Соедините это, и получится следующая строка:

Head1.style.color = "red"

Добавьте ее в функцию и сохраните файл. В окончательном варианте страница должна выглядеть так:

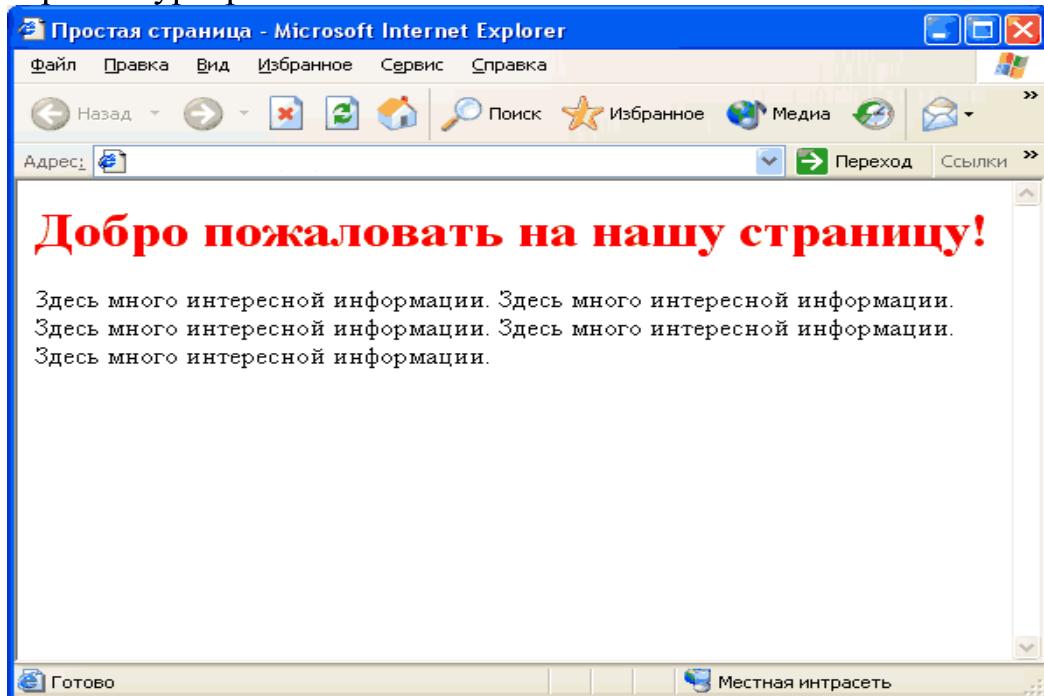
```

<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function colorchange()
{
head1.style.color = "red";
}
</script>
</head>
<body>
<h1 id="head1" onMouseover="colorchange()">Добро пожаловать на
нашу страницу!</h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации. </p>
</body>
</html>

```

Откройте эту страницу в браузере и посмотрите, что происходит, когда вы наводите курсор на заголовок. Если все было сделано правильно, то цвет заголовка изменится.

Но обратите внимание, что цвет текста не становится прежним, когда вы убираете курсор с заголовка.



## 2. Пример использования атрибута text-decoration.

Используя в сценариях JavaScript атрибуты, название которых пишется через дефис, убирайте дефис и пишите оба слова слитно, причем второе слово должно начинаться с заглавной буквы. Таким образом, textDecoration в сценариях должно выглядеть как textDecoration.

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function addunderline()
{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">Добро пожаловать на нашу
страницу! </h1>
<p>Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.</p>
</body>
</html>
```

Необходимо обратить внимание на прописную букву в слове textDecoration — если все слово набрать в нижнем регистре, сценарий выполняться не будет. Теперь при наведении курсора заголовок станет подчеркнутым, а затем, если убрать курсор, вернется в прежнее состояние.

### 3. Пример точного позиционирования текста

Сначала необходимо рассмотреть, каким образом осуществляется позиционирование текста.

Наиболее часто применяются следующие атрибуты позиционирования:

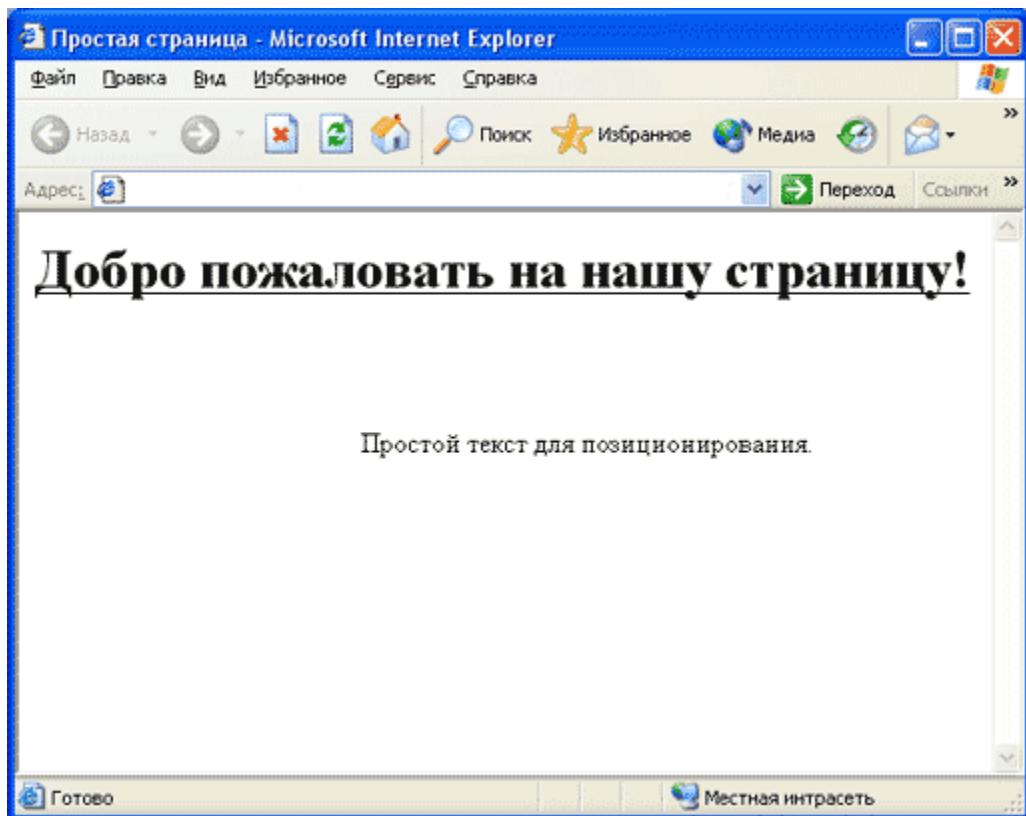
- position - имеет два интересующих нас значения: absolute и relative (по умолчанию значение static). Для значения absolute в качестве точки отсчета используется верхний левый угол окна браузера, и все параметры местоположения отмеряются от него. В свою очередь, для relative точкой отсчета является то место, в котором разместился бы

элемент страницы, если бы не было представлено никакой информации о местоположении;

- top - используется для указания вертикального смещения элемента от точки отсчета. Величина смещения может выражаться в различных единицах (пиксели, дюймы, сантиметры, миллиметры и т.п.). В наших примерах используются пиксели. Положительное значение top соответствует смещению элемента страницы вниз, в то время как отрицательное - по направлению к верхней границе окна браузера;
- left - подобен атрибуту top, но применяется для указания горизонтального направления. Положительное значение соответствует сдвигу элемента вправо, отрицательное - влево.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 style="text-decoration: underline">Добро пожаловать на нашу
страницу!</h1>
<p style="position:absolute; top:125px; left:200px">Простой текст для
позиционирования.</p>
</body>
</html>
```

С помощью CSS текст расположен со значением absolute, то есть его положение отсчитывается от верхнего левого угла окна браузера. Значение атрибута top равно 125px, таким образом, текст будет расположен на 125 пикселов ниже верхнего края страницы. Значение атрибута left равно 200px, то есть текст будет сдвинут на 200 пикселов от левого края окна браузера.



## 1.8. Создание анимационных объектов

Анимация - это процесс «оживления» объекта

Анимация включает в себя две составляющие: расстояние между соседними кадрами, называемое скачком (jump), и временной промежуток между двумя последовательными скачками, называемый интервалом (interval). При больших скачках и длительных интервалах анимация выглядит медленной и грубой. Движение объектов кажется неестественным и воспринимается как мелькание. При малых скачках и кратких интервалах анимация выглядит более плавной, хотя, если чересчур увлечься, движение покажется нарочитым.

### 1. Пример перемещения текста слева направо

Сначала следует ввести текст в тэге <div>, ограничивающем текст, добавить идентификатор id.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<div id="anim">
Текст, шагом марш!
</div>
```

```
</body>
</html>
```

Затем воспользуемся CSS, чтобы поместить текст в начальное положение:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Далее начинаем работать над сценарием JavaScript. Поскольку не нужно, чтобы текст вечно двигался вправо, надо предусмотреть возможность контролирования этого процесса. Чтобы запустить сценарий на выполнение только при условии, если текст находится, например, менее чем в 500 пикселях от левой границы экрана, удобнее всего воспользоваться оператором if. Для этого понадобится атрибут CSS pixelLeft.

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
}
}
</script>
</head>
<body>
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Теперь рассмотрим операторы, управляющие анимацией. Прежде всего нужно задать скачок. Каждый раз текст будет перемещаться вправо на 50 пикселей. Атрибут pixelLeft используется не только для определения положения текста, но и для изменения положения на 50 пикселей:

```

<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
anim.style.pixelLeft +=50;
}
}
</script>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

Далее речь пойдет об интервале. Он задается с помощью метода setTimeout, позволяющего вновь запустить функцию после истечения определенного промежутка времени. Давайте установим интервал до повторного запуска функции moveTxt(), равным 5000 мс:

```

<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
<!-- Маскируемся!
function moveTxt()
{
if (anim.style.pixelLeft &lt; 500)
{
anim.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
&lt;/script&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;div id="anim" style="position:absolute; left:10; top:10"&gt;
Текст, шагом марш!
&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>

```

Процесс будет повторяться до тех пор, пока условие оператора if не станет ложным. Последнее, что нужно сделать, - запустить сценарий на выполнение. Для этого следует воспользоваться событием onLoad:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{
anil.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

## 2. Пример движения текста по диагонали

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop < 500)
{
anim.style.pixelTop += 2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
```

```
</body>  
</html>
```

## 1.9. Слои

### *Позиционирование слоя*

Для создания слоев следует использовать тег `<div>` или `<span>`. Эти теги взаимозаменяемы и различаются лишь внешним видом в браузере. Если требуются отступы до и после текста, следует использовать элемент `<div>`. При размещении текста внутри параграфа применяется тег `<span>`.

В Таблице 11 перечислены наиболее важные атрибуты.

Таблица 11.

<code>id</code>	Имя слоя, используемое для указания его в <code>&lt;script&gt;</code>
<code>left</code>	Позиция слоя по x координате
<code>top</code>	Позиция слоя по y координате
<code>position</code>	Задает относительную или абсолютную позицию относительно других объектов
<code>z-index</code>	Позиция слоя при наложении нескольких объектов друг на друга
<code>width</code>	Ширина слоя в пикселях или %
<code>height</code>	Высота слоя в пикселях или %
<code>bgColor</code>	Цвет фона слоя
<code>background</code>	Картинка фона
<code>src</code>	Внешний html документ, содержащийся в слое

Пример наложения текста:

```
<html>  
<body>  
Слой1 наверху  
<div style="position:relative; font-size:50px; z-index:2; color: navy">  
Слой 1  
</div>  
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px;  
z-index:1">  
Слой 2  
</div>
```

Слой 2 наверху

```
<div style="position:relative; font-size:50px; z-index:3; color: navy">
```

Слой 1

```
</div>
```

```
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px; z-index:4">
```

Слой 2

```
</div>
```

```
</body>
```

```
</html>
```

Тип позиционирования слоя определяется параметром position, положение элемента - двумя координатами top и left.

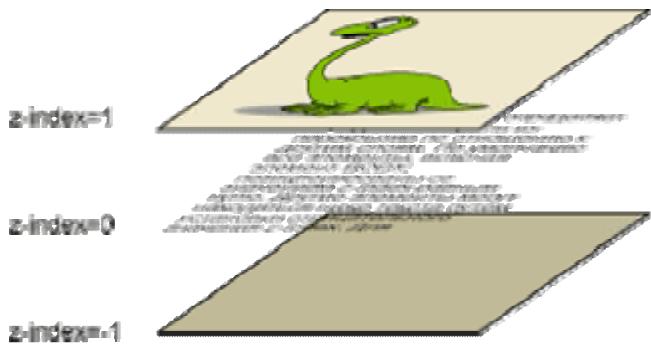
Кроме тегов <div> и <span> абсолютное позиционирование поддерживают следующие элементы: <applet>, <input>, <button>, <object>, <select>, <fieldset>, <iframe>, <table>, <img>, <textarea>.

Параметр position:relative используется для смещения слоя относительно родительского элемента. Установка этого значения не изменяет размещение элемента, но если установлены значения свойств top или left, то слой смещается от своего нормального положения в документе.

В то время как свойство position указывает тип системы координат, параметры top и left определяют точную позицию слоя. Значения этих параметров могут определяться в процентном отношении или пикселях, принимать положительные и отрицательные величины. Это дает возможность размещать контент выше или ниже на странице независимо от физической позиции кода HTML. То есть, в верхней части страницы можно поместить слой, который описан внизу HTML-документа.

### *Свойство z-index*

Свойство z-index определяет порядок слоев, или их перекрытие по отношению к другим слоям. По умолчанию все слои позиционированы со значением z-index равным нулю. Другие слои могут размещаться ниже путем установки отрицательного значения z-index. Для слоев, у которых z-index не установлен, это значение назначается неявно в соответствии с их положением в документе. Поэтому слой, который помещен в документ позже, размещается выше остальных элементов, позиционированных ранее.



### *Свойства visibility и display*

Для отображения или скрытия слоя используется свойство `visibility`. Он может принимать значения `visible`, установленное по умолчанию, для показа слоя, и `hidden`, которое его прячет.

Например, скрытый блок текста можно оформить следующим образом:

```
<div style="visibility: hidden">Спрятанный слой</div>
```

При этом, когда используется данное свойство для скрытия элемента, соответствующий данному элементу блок занимает прежнее положение на странице, но сома содержимо не отображается.

Чтобы на странице не оставалось пустого блока, соответствующего скрываемому элементу, можно использовать свойство `display` со значением `none`. Для отображения элемента `display` равно `block`.

### *Динамическое управление слоями*

Сценарии JavaScript позволяют динамически управлять параметрами установленных слоев. Это позволяет получить такие эффекты, как скрытие и отображение слоя, изменение порядка отображения, перемещение слоя в окне браузера. Все эти эффекты достигаются с помощью изменения соответствующих стилевых параметров установленных слоев.

Для обращения к слоям из сценариев JavaScript, удобнее всего каждому слою дать собственное имя при помощи параметра `id`. Например:

```
<div id="div1">  
...  
</div>
```

Для того, чтобы скрыть отображение слоя `div1`, можно использовать следующую команду:

```
div1.style.visibility='hidden';
```

Для повторного отображения слоя следует выполнить следующее присвоение:

```
div1.style.visibility='visible';
```

Пример динамической смены слоев: в данном примере для отображения некоторого слоя следует нажать на соответствующую

ссылку. Этую идею можно применить и для организации выпадающих меню.

```
<html>
<head>
<style type="text/css">
div {
position: absolute;
top: 20;
left: 160;
visibility: hidden;
}
</style>
<script language="JavaScript">
function show_d(d)
{
div1.style.visibility='hidden';
div2.style.visibility='hidden';
div3.style.visibility='hidden';
div4.style.visibility='hidden';
div5.style.visibility='hidden';
d.style.visibility='visible';
}
</script>
</head>
<body>
<a href="javascript:void(0)" onClick="show_d(div1);">
показать слой 1
</a><br>
<a href="javascript:void(0)" onClick="show_d(div2);">
показать слой 2
</a><br>
<a href="javascript:void(0)" onClick="show_d(div3);">
показать слой 3
</a><br>
<a href="javascript:void(0)" onClick="show_d(div4);">
показать слой 4
</a><br>
<a href="javascript:void(0)" onClick="show_d(div5);">
показать слой 5
</a><br>
<div id="div1">
<h3>Слой номер один</h3>
```

Некоторый текст, на слое расположенный. Его можно скрыть и показать. Текст может содержать несколько строк.

```
</div>
<div id="div2">
<h3>Слой номер два</h3>
Содержит свой текст. Если показывается, то текст на других слоях не
виден.
```

```
</div>
<div id="div3">
<h3>Слой номер три</h3>
```

Тоже текст. При работе со слоями надо следить, чтобы текст одного слоя не "выглядывал" из-под другого слоя при самых различных размерах окна браузера и используемых шрифтах.

```
</div>
<div id="div4">
<h3>Слой номер четыре</h3>
Здесь нет текста.
</div>
<div id="div5">
<h3>Слой номер пять</h3>
И тут тем более нет.
</div>
</body>
</html>
```

### *Динамическое изменение цвета фона ячеек*

Использование стилей и управление ими с помощью JavaScript позволяет менять вид ячейки "на ходу", при выполнении определенных условий, таких как наведение курсора на ссылку или саму ячейку.

Рассмотрим самый простой прием - цвет фона ячейки меняется, когда курсор мыши наводится на нее. Наведение мыши на область отслеживается событием onmouseover, а вывод мыши за ее пределы - событием onmouseout. Поскольку цвет фона меняется у той же самой ячейки, на которую наводим курсор мыши, то изменение стиля делается с помощью метода this.style.background.

```
...
<table width=60% border=1 cellspacing=0 cellpadding=4
bordercolor=#333333 align=center>
<tr>
<td align=center bgcolor="#cccccc"
onMouseover="this.style.background='#ffcc33'"
onmouseout="this.style.background='#cccccc'"><a href="#">Пункт
1</a></td>
<td align=center bgcolor="#cccccc"><a href="#">Пункт 2</a></td>
```

```
</tr>  
</table>
```

```
...
```

## **2. Практика**

### **Постановка задачи**

Необходимо выполнить практические работы №1-№5 и итоговое задание, предложенное в конце.

Для выполнения итогового задания Вам необходимо иметь созданный в курсе «HTML» Web-сайт, состоящий из нескольких (не менее трех) связанных между собой статических HTML-страниц и использующий основные возможности языка HTML.

Для выполнения всех практических работ Вам необходимо иметь текстовый редактор (возможна работа в специальных редакторах Web-документов, например Adobe Dreamweaver), несколько браузеров для просмотра Ваших страниц (Internet Explorer, Mozilla Firefox, Opera и другие).

#### **2.1. Практическая работа №1. Размещение скриптов в HTML-документе.**

##### **Задание 1.**

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды внутри тэга <script>.
4. Добавьте команду вывода аналогичного приветственного сообщения в окно браузера после закрытия диалогового окна.
5. Сохраните документ с именем Ex1.html в рабочей папке.

##### **Задание 2.**

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветственного сообщения в диалоговом окне, задав необходимые команды JavaScript во внешнем файле. Для этого:
  - создайте новый текстовый файл,
  - поместите в него код JavaScript,

- сохраните файл с именем main.js следующим образом: укажите тип файла “Все файлы”, кодировку “UTF-8”.
4. Добавьте ссылку на внешний скриптовый файл из рабочего HTML-документа.
  5. Сохраните документ с именем Ex2.html в рабочей папке.

### **Задание 3.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex3.html в рабочей папке.
3. Добавьте в документ код JavaScript так, чтобы в диалоговом окне появлялось поле с надписью "Введите сюда своё имя" и со значением по умолчанию в поле "Введите имя". Для этого используйте метод prompt(...) объекта window. Для хранения введенного значения заведите новую переменную.
4. Организуйте вывод введенного значения имени в окно браузера в виде: "Ваше имя <.....>".
5. Дополните код, чтобы в новом диалоговом окне появилась надпись "Начать заново? " При положительном ответе появлялось диалоговое окно: "Не надоело? ", при отказе – "Ну и правильно!". Используйте для написания методы alert(...) и confirm(...) объекта window.

## **2.2. Практическая работа №2. Операторы управления, функции. Объекты ядра JavaScript.**

### **Задание 4.**

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>if</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
var x, y;
x=parseInt(prompt("Введите значение x","")); // метод parseInt()
переводит строку в целое
y=parseInt(prompt("Введите значение y","")); // число
if(x<y)
{
alert("Максимальное число - y")
}
else {
alert("Максимальное число - x")
```

переводит строку в целое

```
y=parseInt(prompt("Введите значение y","")); // число
```

```
if(x<y)
```

```
{
```

```
alert("Максимальное число - y")
```

```
}
```

```
else {
```

```
alert("Максимальное число - x")
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Допишите скрипт так, чтобы при введении пользователем одинаковых чисел, открывалось сообщение "Введенные числа равны!".
3. Напишите скрипт, в котором пользователя просят ввести правильный пароль. При вводе правильного пароля, в окне браузера появляется сообщение о том, что пароль верен. При вводе неправильного пароля – выпадает сообщение о неправильно введенном пароле. Для выполнения задания введите переменную password, в которую сохраните верное значение пароля.
4. Сохраните документ с именем Ex4.html в рабочей папке.

### **Задание 5.**

1. Рассмотрите пример скрипта:

```
<html>
```

```
<head>
```

```
<title>for</title>
```

```
</head>
```

```
<body>
```

```
<h1>Пример простой</h1>
```

```
<script language="JavaScript" type="text/JavaScript">
```

```
function line() {
```

```
    document.writeln("<hr align='center' width='100'>");
```

```
}
```

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

```
    line();
```

```
</script>
```

```
</body>
```

```
</html>
```

2. Создайте вариант прорисованных линий со следующим условием:
  - десять линий должны располагаться друг под другом,
  - первая должна быть длинной 10 пикселей,
  - каждая последующая на 10 пикселей больше.
3. Сохраните документ с именем Ex5.html в рабочей папке.

### **Задание 6.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex6.html в рабочей папке.

3. Добавьте в документ код JavaScript так, чтобы в окне браузера была выведена таблица степеней двойки вида:

Степень	Результат
$2^0$	1
$2^1$	2
$2^2$	4
$2^3$	8
$2^4$	16
$2^5$	32

Для этого в сценарии используйте метод write(...) объекта document для формирования содержимого страницы. На каждой итерации цикла for сформируйте очередную строку таблицы, в первую ячейку которой заносится соответствующая степень двойки, а во вторую результат ее возведения в указанную степень. Для выполнения этого действия используется встроенный объект Math и его метод pow(...), возводящий первый параметр в степень, заданную вторым параметром. Обратите внимание, что метод write(...) может вызываться с любым количеством фактических параметров. Результатом его работы в любом случае является вывод в документ строки, полученной конкатенацией всех параметров, переданных в метод.

### Задание 7.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>array</title>
</head>
<body>
<script language="JavaScript">
year=new Array("декабрь","январь","февраль","март","апрель","май",
"июнь","июль","август","сентябрь","октябрь","ноябрь");
summer=new Array(); //летние месяцы
summer=year.slice(6,9);
document.write(summer+"<br>");
</script>
</body>
</html>
```

2. Создайте массив, содержащий названия школьных предметов. Выделите из него два массива. Пусть к первому относятся предметы из раздела точных наук, а ко второму - из раздела гуманитарных

наук. Для создания и вывода в окно браузера новых массивов используйте метод slice(...) и write(...) объекта document. Оформите исполняющий скрипт в виде отдельной функции, описанной в разделе <head> и вызванной в разделе <body>.

3. Сохраните документ с именем Ex7.html в рабочей папке.

### **Задание 8.**

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex8.html в рабочей папке.
3. Добавьте скрипт, на основе которого будут выполняться следующие условия:
  - если на страницу зашел пользователь через браузер Microsoft Internet Explorer, перенаправьте его автоматически на страницу Ex1.html;
  - если на страницу зашел пользователь через любой другой браузер, перенаправьте его на страницу Ex3.html.Для выполнения задания используйте свойство appName объекта navigator.

## **2.3. Практическая работа №3. Объекты клиентских приложений. Обработка событий.**

### **Задание 9.**

1. Рассмотрите скрипт:

```
<html>
<head>
<title>document</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
document.write("Спасибо, что пришли к нам на курсы!");
</script>
</body>
</html>
```

2. Допишите скрипт так, чтобы

- цвет фона документа был #E7E6D8,
- цвет шрифта – красный,

- внизу выводилась дата последней модификации документа, используйте для этого слияние методов `wtime(...)` и `lastModified(...)` объекта `document`.

3. Сохраните документ с именем Ex9.html в рабочей папке.

### **Задание 10.**

1. Рассмотрите пример скрипта открытия нового окна на странице:

```
<html>
<head>
<title>window</title>
</head>
<body>
<h1>Создание нового окна</h1>
<hr>
<script language="JavaScript" type="text/JavaScript">
window.open("http://www.google.com","","toolbar=no,scrollbars=yes,width=250, height=250, resizable=yes, top=100, left=500")
</script>
</body>
</html>
```

2. Измените скрипт так, чтобы выполнялись следующие условия:

- открытие нового окна происходило при нажатии на ссылку с текстом: «Щелкните на ссылке для получения справочной информации»,
- размеры окна - 500x500,
- есть возможность изменения размеров окна.

Для выполнения задания используйте написание функции.

3. Сохраните документ с именем Ex10.html в рабочей папке.

### **Задание 11.**

1. Создайте страницу с переадресацией на другой адрес (redirect).
2. Измените скрипт так, чтобы переадресация на другой адрес была с задержкой 5 секунд.
3. Сохраните документ с именем Ex11.html в рабочей папке.

### **Задание 12.**

1. Создайте HTML-документ, в котором будет 2 ссылки:

- первая ссылка должна ссылаться на PDF файл; при нажатии на нее выпадает сообщение с предупреждением о том, что для загрузки документа требуется программа Acrobat, и

продолжить загрузку или нет; используйте для написания метод confirm(...) для подтверждения загрузки;

- вторая ссылка должна содержать такой код, чтобы при наведении на нее мыши менялся цвет фона документа на красный.

2. Сохраните документ с именем Ex12.html в рабочей папке.

### Задание 13.

1. Создайте HTML-документ, содержащий любую картинку.

2. Добавьте скрипт с условиями:

- при наведении курсора мыши на картинку она увеличивается,
- при отведении курсора мыши – уменьшается до исходного размера.

Постройте скрипт через использование функций и событий MouseOver и MouseOut.

3. Сохраните документ с именем Ex13.html в рабочей папке.

### Задание 14.

1. Создайте HTML-страницу содержащую следующую форму заполнения данных:

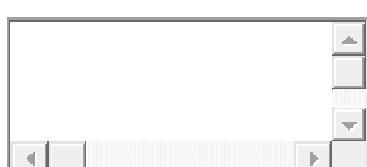
Ваше имя: \*

Пароль \*

Подтверждение пароля\*

Электронный адрес: \*

Тема сообщения:



Сообщение:

\* - необходимые для заполнения поля

2. Добавьте скрипт, проверяющий следующие данные:
  - заполнено ли поле имени,
  - введен ли пароль и содержит ли он больше 4-х символов. Используйте для этого свойство length данного поля,
  - совпадают ли значения, введенные в оба поля для паролей,
  - заполнено ли поле электронного адреса и содержит ли оно символ @,
  - заполнено ли поле сообщения и содержит ли оно больше 10 символов,
3. При несоблюдении условий, курсор должен установиться в то поле, где пользователем введено неверное значение.
4. Сохраните документ с именем Ex15.html в рабочей папке.

## 2.4. Практическая работа №4. Объединение JavaScript и CSS.

### Задание 15.

1. Рассмотрите скрипт:

```
<head>
<title>h1</title>
<script language="JavaScript">
function colorchange()
{
head.style.color = "red";
}
</script>
</head>
<body>
<h1 id="head" onmouseover="colorchange()">Добро пожаловать на
нашу страницу!</h1>
</body>
</html>
```

2. Допишите скрипт страницы таким образом, чтобы красный цвет исчезал после отвода курсора мыши с заголовка.
3. Сохраните документ с именем Ex15.html в рабочей папке.

### Задание 16.

1. Рассмотрите скрипт:

```
<html>
<head>
<title>text decoration</title>
<script language="JavaScript">
function addunderline()
```

```

{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">
Добро пожаловать на нашу страницу!
</h1>
</body>
</html>

```

2. Допишите скрипт страницы таким образом, чтобы на одинарный щелчок мыши появлялось полоса над заголовком, а на двойной щелчок – текст зачеркивался. Используйте события onclick, ondblclick и значения рассматриваемого свойства overline и line-through.

3. Сохраните документ с именем Ex18.html в рабочей папке.

### **Задание 17.**

1. Создайте HTML-документ, содержащий любое изображение.
2. Поместите изображение в тег <div>. Задайте для него абсолютное позиционирование со смещением вниз и влево на 500 пикселей.
3. Сохраните документ с именем Ex17.html в рабочей папке.

## **2.4. Практическая работа №5. Слои. Движущиеся элементы.**

### **Задание 18.**

1. Рассмотрите скрипт:

```

<html>
<head>
<title>simple animation</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{

```

```

anil.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

2. Измените скрипт страницы:

- добейтесь плавного передвижения текста;
- измените направление текста - задайте направление сверху вниз при помощи атрибута pixelTop.

3. Сохраните документ с именем Ex18.html в рабочей папке.

### **Задание 19.**

1. Рассмотрите скрипт:

```

<head>
<title>animal</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop <500)
{
anim.style.pixelTop +=2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>

```

2. Измените направление текста. Задайте направление с верхнего правого угла экрана (приблизительно) по диагонали к середине экрана.

3. Сохраните документ с именем Ex19.html в рабочей папке.

### **Задание 20.**

1. Создайте HTML-страницу, на которой будет три слоя. Верхний и нижней представляют из себя статичные квадраты разного цвета с текстом, а между ними должна проплывать любая картинка слева направо.

### **Абсолютное позиционирование и Z-index**



2. Сохраните документ с именем Ex20.html в рабочей папке.

### **Итоговое задание**

1. Перейдите к Web-сайту, созданному в курсе “Web-программирование: HTML”.
2. Добавьте к странице, содержащей форму, скрипт, осуществляющий проверку введенных в форму данных.
3. Добавьте к остальным страницам скрипты на свое усмотрение.