МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Т.В. Зудилова, М.Л. Буркова

Web-программирование JavaScript

Учебное пособие

Санкт-Петербург

2012

Оглавление

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.

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 некими 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.

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 обратится 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 отображенного 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>
```

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

Для присваивания переменным значений основных типов применяются литералы – буквальные значения данных соответсвующих 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
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения на единицу
--	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
= =	Равно	&&	логическое И
! =	Не равно	| |	логическое ИЛИ
>=	Больше или равно	!	логическое НЕ

```
<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 соответсвующего 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 их 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

```
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 невыполненными все последующиевыражения, 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>
```

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.

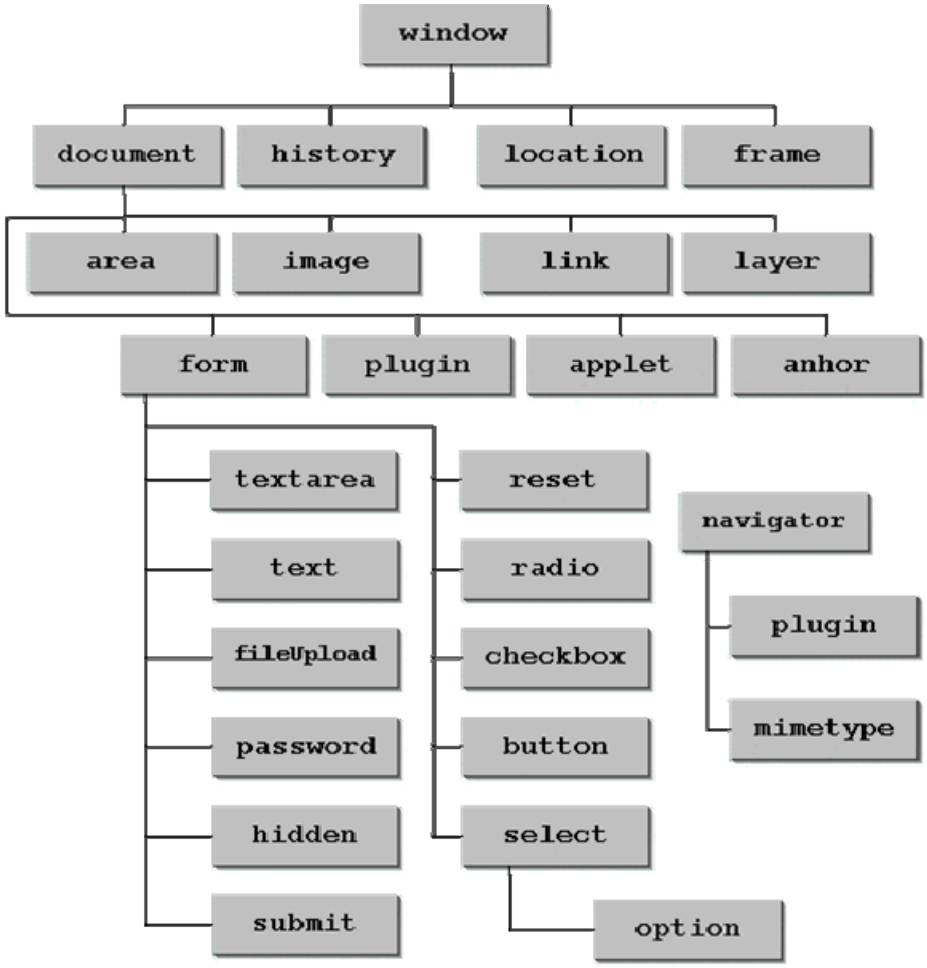
1.5. Объекты клиента

При интерпритации страницы HTML браузером создаются объекты JavaScript, свойтсва 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

```
    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().

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 ОК и определенным программистом текстом
- метод confirm() – выводит диалоговое окно с кнопками ОК 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:

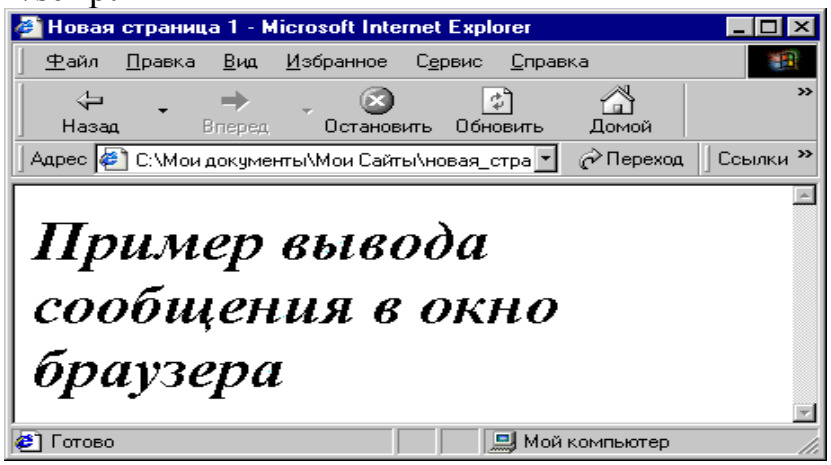
document.images.name.src

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

Таблица 6.

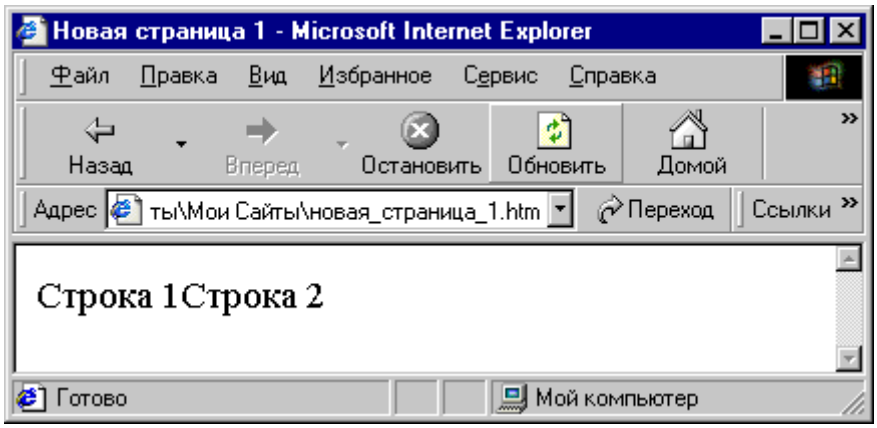
Свойства	Назначение
bgColor	Устанавливает цвет фона текущего документа. Этот цвет может иметь шестнадцатеричное представление #rrggbb или соответствующее название. Синтаксис: document.bgColor="#e7e6d8"
fgColor	Устанавливает цвет текста документа. Аналогичен по функциям свойству bgColor
referrer	Указывает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

<script language="JavaScript">
document.write('<h1>< i>Пример вывода сообщения в окно браузера</i></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
, который включается в состав выводимой строки;

2. используя метод document.writeln().

Следующие примеры приведут к одинаковому результату:

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

Таблица 9.

Метод	Описание
assign(url)	Загружает документ, адрес которого передан в качестве параметра. Поддерживаетс только IE начиная с 4.0
reload()	Перезагружает документ с Web-сервера.
replace(url)	Загружает документ, адерс 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;

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:

- . . .
- <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()">
```

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:

документ page1

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:

документ page1

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:

 Подтверждение активизации гиперсвязи

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

```
</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

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<hl id="head1">Доброо 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 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:

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, text-decoration 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>3десь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.
    Здесь много интересной информации.</p>
</body>
</html>
```

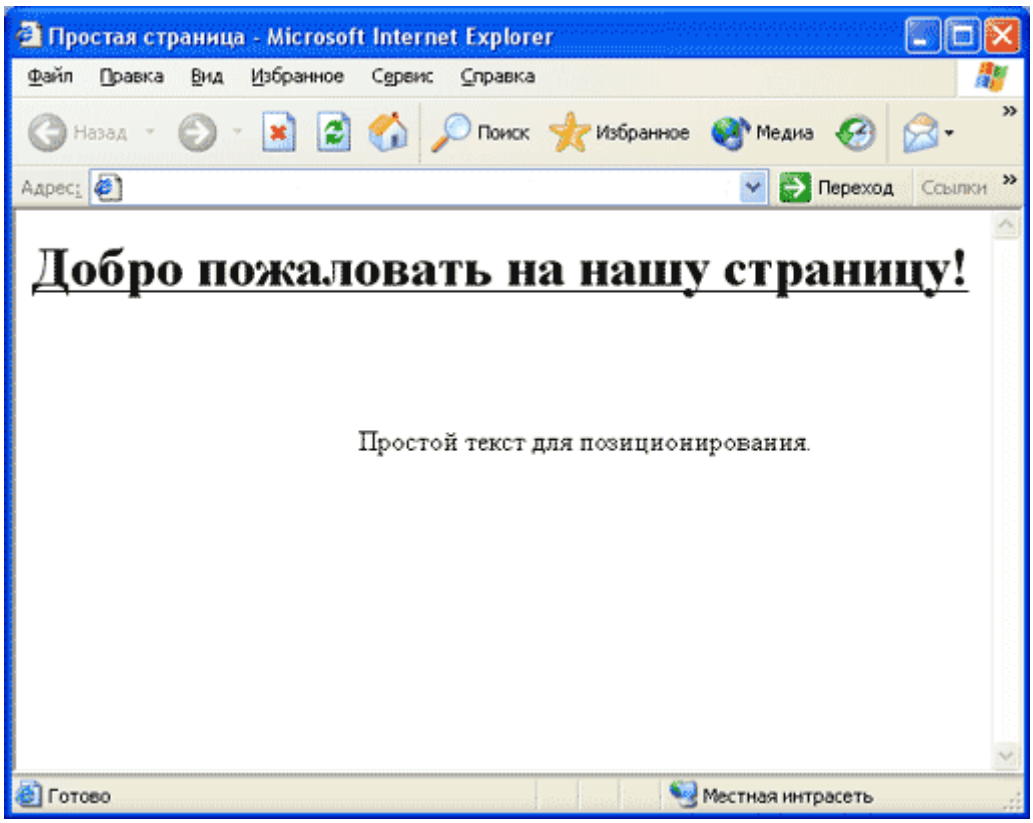
Необходимо обратить внимание на прописную букву в слове textDecoration — если все слово набрать в нижнем регистре, сценарий выполняться не будет. Теперь при наведении курсора заголовок станет подчеркнутым, а затем, если убрать курсор, вернется в прежнее состояние.

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



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 , ограничивающем текст, добавить идентификатор id.

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

```
<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 < 500)
{
anim.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
</script>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

</body>
</html>

1.9. Слои

Позиционирование слоя

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

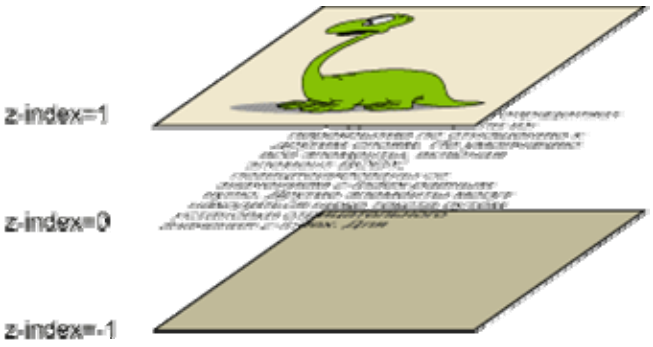
В Таблице 11 перечислены наиболее важные атрибуты.

Таблица 11.

id	Имя слоя, используемое для указания его в <script>
left	Позиция слоя по x координате
top	Позиция слоя по y координате
position	Задает относительную или абсолютную позицию относительно других объектов
z-index	Позиция слоя при наложении нескольких объектов друг на друга
width	Ширина слоя в пикселах или %
height	Высота слоя в пикселах или %
bgColor	Цвет фона слоя
background	Картинка фона
src	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>
```



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:

Спрятанный слой

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

Чтобы на странице не оставалось пустого блока, соотвествующего 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:

...

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

```
</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>
```

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 приветсвенного сообщения в диалоговом окне, задав необходимые команды внутри тэга <script>.
4. Добавьте команду вывода аналогичного приветсвенного 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 приветсвенного сообщения в диалоговом окне, задав необходимые команды JavaScript во внешем файле. Для этого:
   - создайте новый текстовый файл,
   - поместите в него код JavaScript,

```
}
</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**.**

наук. Для создания и вывода в окно браузера новых массивов используйте метод slice(…) и wtite(…) 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,

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 рабой 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:

Отправить Очистить

* - необходимые для заполнения поля

```
{
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 . 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.

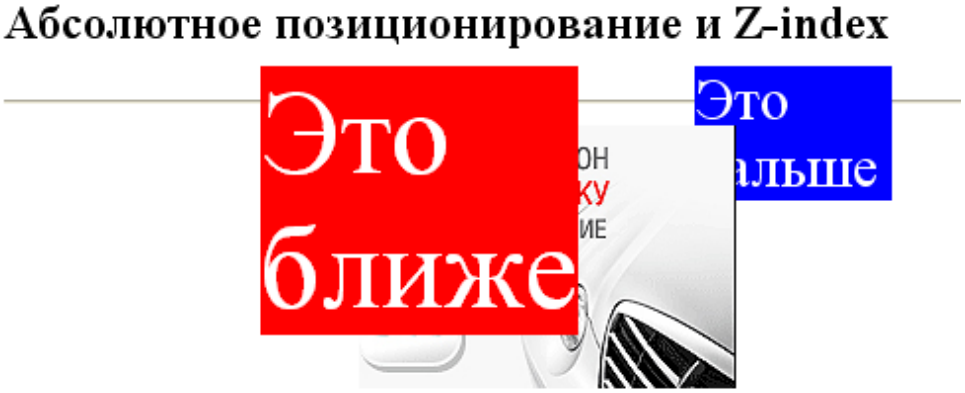
1. Рассмотрите скрипт:

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

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 меду ними должна проплывать любая картинка слева направо.



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.

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.

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 IT- 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 IT- специалистов на кафедре ПС неоднократно были удостоены **высокими наградами «Компьютерная Элита» в номинации лучший учебный центр России**.

Партнеры:

1. **Microsof**t Certified Learning Solutions;
2. **Novell** Authorized Education Center;
3. **Cisco** Networking Academy;

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