УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите теоретические сведения к лабораторной работе, выполните пример и задание согласно вашему варианту.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: igor-gricenko-95@mail.ru **в тчении ТРЕХ дней**

Требования к отчету:

Отчет предоставляется преподавателю в электронном варианте и должен содержать:

- **название работы, постановку цели, вывод;**
- **ответы на контрольные вопросы, указанные преподавателем.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)132-63-42,

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лабораторная работа №11.

Тема: Создание документов с помощью JavaScript

JavaScript был разработан совместно компаниями Sun Microsystems и Netscape. За синтаксическую основу нового языка был взят язык Java, в свое время разработанный компанией Sun Microsystems. В последнее время популярность JS очень возросла в результате выхода в свет браузеров, поддерживающих данный язык. JS - интерпретатор с элементами объектно-ориентированной модели. Хотя он и лишен возможностей создания собственных классов, но он оперирует стандартными объектами. Так как обработчик находится на компьютере пользователя, JS, будучи интерпретатором, использует методы и свойства объектов обозревателя на пользовательском компьютере. JS имеет возможность написания пользовательских функций, имеет ряд операторов, но работает с объектами, их методами, свойствами и событиями. Также имеется иерархия наследования свойств объектов. Сложность составляет и то, что JS встраивается в html документ и взаимодействует с ним.

Microsoft выпустила похожие версии языка под названием JScript, поэтому под названием "JavaScript" часто понимается любая версия языка, в том числе и Microsoft JScript.

В большинстве случаев при упоминании JavaScript подразумевается так называемый клиентский JavaScript, интерпретатор которого встроен в Web-браузеры. Однако JavaScript изначально был разработан как универсальный язык

программирования для встраивания в любое приложение и обеспечения возможности написания в нем сценариев. Например, ActionScript, язык сценариев, доступный в Macromedia Flash, также смоделирован в соответствии со стандартом ECMAScript.

Рассмотрим способы использования JavaScript внутри HTML-документа.

Способ первый

На первом этапе мы составим программу JavaScript, которая вставляет слова "Hello, world!" непосредственно в документ HTML. Программы или сценарии JavaScript встраиваются в документ HTML. Взгляните на листинг 1, в которой приведен исходныйтекст документа с программой, составленной на JavaScript.

Листинг 1.

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("Hello, world!");
// -->
</SCRIPT>
</BODY>
</HTML>
```

Как и подобает любому документу HTML, он ограничен операторами <HTML>, </HTML> и состоит из двух разделов. Раздел заголовка выделяется операторами <HEAD> и

</HEAD>, а раздел тела документа - операторами <BODY> и </BODY>. ПрограммаJavaScript этом примере встроена в тело документа HTML при помощи операторов

<SCRIPT> и </SCRIPT>.

С помощью оператора <SCRIPT> можно встроить в документ сценарий, составленный на языке JavaScript или VBScript. Язык указывается с помощью

параметра LANGUAGE. Текст сценария оформлен как комментарий с применением операторов <!-- и -->. Это сделано для того, чтобы сценарий не вызывал проблем у пользователей, браузеры которых не могут работать с JavaScript. Такие браузеры просто проигнорируют сценарий. Обратите внимание на строку, которой завершается комментарий: // --> Перед символами --> записаны два символа /. Это позволяет обеспечить работоспособность сценария в различных браузерах. Некоторые из них (например, Netscape Navigator) в сценариях JavaScript рассматривают строку --> как ошибочную. Символы // используются в JavaScript для выделения комментариев и предписывают браузерам игнорировать символы, записанные после них (в том числе и -->). Для обозначения комментариев можно использовать также конструкцию /*...*/. Этот способ удобен, если комментарий содержит несколько строк.

Наша первая программа весьма проста и содержит только одну строку:

```
document.write("Hello, world!");
```

Здесь для объекта с именем document вызывается метод write. В качестве параметра ему передается текстовая строка "Hello, world!". Строка закрывается символом точка с запятой, хотя этот символ может и отсутствовать. Объект document - это документ HTML,загруженный в окно браузера. Он содержит в себе объекты, свойства и методы, предназначенные для работы с элементами этого документа HTML, а также для взаимодействия с другими объектами. Метод write записывает в тело документа HTML приветственную строку "Hello, world!". При этом документ будет выглядеть так, как будтоэта строка находится в нем на месте сценария:

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>Hello, world!
</BODY>
</HTML>
```

Способ второй

Исходный текст любого сценария должен включаться в документы HTML. Однако, есть техническая возможность оформлять программы на JavaScript в отдельных файлах, а в страницах HTML указывать на эти файлы ссылки. Браузер, загружая оформленные подобным образом HTML документы, загружает оформленные в отдельных файлах сценарии и подставляет их вместо

соответствующих ссылок. Такой способ включения JavaScript сценариев удобен, если один и тот же сценарий должен быть включен во множество документов HTML, или же если есть необходимость скрыть исходный код от просмотра пользователями (через просмотр источника).

Ссылки на файлы с подгружаемыми скриптами оформляются с помощью параметра SRC тега <SCRIPT>, допускающего указывать адрес URL файла сценария. Следующий пример демонстрирует использование параметра SRC. В листинге 2 находится исходный текст документа HTML, содержащий ссылку на файл сценария hello.js.

Листинг 2.

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
<SCRIPT LANGUAGE="JavaScript" SRC="hello.js">
</SCRIPT>
</BODY>
</HTML>
```

Ссылка оформлена с применением операторов <SCRIPT> и </SCRIPT>, однако между этими операторами нет ни одной строчки исходного текста. Этот текст перенесен в файлhello.js (листинг 3).

Листинг 3. Файл hello.js

```
document.write("<HR>"); document.write("Hello, world!"); document.write("<HR>");
```

В параметре SRC вышеприведенного примера задано только имя файла, так как он находится в том же каталоге, что и ссылающийся на него файл документа HTML. Однако можно указать и относительный путь, и полный адрес URL, например: <SCRIPT LANGUAGE="JavaScript" SRC="http://www.myserver.ru/scripts/hello.js"> Существенно, чтобы файл, в котором находится исходный текст сценария JavaScript, имел тип js. В противном случае сценарий работать не будет.

Способ третий

В сценариях JavaScript активно применяют функции и переменные.

Приведем исходный текст простой программы, в которой используется переменная и функция (листинг 4).

Листинг 4.

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
var szMsg = "Hello, world!";function printHello()
{
document.write(szMsg);
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
<SCRIPT LANGUAGE="JavaScript">
<!--
printHello();
// -->
</SCRIPT>
</BODY>
</HTML>
```

Прежде всего, обратите внимание на область заголовка документа, выделенную операторами <HEAD> и </HEAD>. В этой области расположено определение переменной и функции, оформленное с применением операторов <SCRIPT> и </SCRIPT>. Кроме того, в теле документа HTML есть еще один раздел сценариев, выделенный аналогичным образом.

Переменная с именем szMsg определяется при помощи оператора var, причем ей сразу же присваивается начальное значение - текстовая строка "Hello, world!".

Язык JavaScript не является типизированным. Это означает, что программист не может указать явным образом тип создаваемых им переменных. Этот тип определяется интерпретатором JavaScript автоматически, когда переменной в первый раз присваивается какое-либо значение. В дальнейшем

можно легко изменить тип переменной, просто присвоив ей значение другого типа. Отсутствие строгой типизации упрощает создание сценариев, особенно для непрофессиональных программистов, однако может привести к ошибкам. Поэтому необходимо внимательно следить за тем, какие типы данных применяются. Этому способствует использование префиксов имен, по которым можно судить о типе переменной. Например, текстовые строки можно начинать с префикса sz, а численные значения - с префикса n.

Помимо переменной szHelloMsg, в области заголовка документа HTML с помощью ключевого слова function определена функция с именем printHello. Эта функция вызывается из сценария, расположенного в теле документа и выводит в документ HTML значение переменной szHelloMsg.

Интерпретация документа HTML и встроенных в него сценариев происходит по мере загрузки документа. Поэтому если в сценарии одни функции вызывает другие или используют определенные в документе переменные, то их определения (вызываемых функций и переменных) необходимо разместить выше вызывающих. Размещение определений переменных и функций в разделе заголовка документа гарантирует, что они будут загружены до момента загрузки тела документа.

Язык JavaScript имеет встроенные средства для отображения простейших диалоговых панелей, таких как панель сообщений. В листинге 5 приведен исходный текст сценария JavaScript, в котором вызывается функция alert, предназначенная для отображения диалоговых панелей с сообщениями.

Листинг 5.

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function printHello()
{
alert("Hello, world!");
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
<SCRIPT LANGUAGE="JavaScript">
```

```
<!--
printHello();
// -->
</SCRIPT>
</BODY>
</HTML>
```

Помимо представленной в этом примере диалоговой панели сценарии JavaScript могут выводить на экран и более сложные. В них пользователь может делать, например, выбор из двух альтернатив или даже вводить какую-либо информацию.

Рассмотрим еще несколько простейших примеров использования JavaScript.

Например, интересной возможностью Javascript является использование диалоговой панели ввода информации. Введенная в диалоговой панели текстовая строка выводится вокне браузера.

Листинг 6.

```
<HTML>
<HEAD>
<TITLE>Hello, world!</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function printHello()
{
szHelloStr=prompt("Введите приветственное сообщение:", "");
document.write(szHelloStr);
}
// -->
</SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
<SCRIPT LANGUAGE="JavaScript">
<!--
printHello();
// -->
</SCRIPT>
</BODY>
</HTML>
```

Диалоговая панель ввода информации вызывается с помощью функции prompt. В качестве параметров функции передается вводное сообщение для пользователя и начальное значение запрашиваемой текстовой строки (в приведенном примере - пустое).

Еще один пример. В языке JavaScript есть удобные средства обработки событий. В следующем примере когда пользователь пытается выбрать ссылку "Select me!", разместив над ней курсор мыши, на экране появляется диалоговая панель с сообщением "Hello, world!". Исходный текст соответствующего документа HTML с встроенным в него сценарием представлен в листинге 7.

Листинг 7.

```
<HTML>
<HEAD>
<TITLE>Hello world!</TITLE>
</HEAD>
<BODY>
<H1>JavaScript Test</H1>
<A HREF="" onMouseover="alert('Hello, world!');">Select me!</A>
</BODY>
</HTML>
```

Здесь для нас интересна строка оператора <A>. Напомним, что этот оператор обычно применяется для организации ссылок на другие документы HTML или файлы различных объектов. В данном случае поле ссылки параметра HREF пустое, однако дополнительно в оператор <A> включена следующая конструкция:

```
onMouseover="alert('Hello, world!');"
```

Она указывает, что при возникновении события onMouseover (наведение мыши) должна выполняться следующая строка программы JavaScript:

```
alert('Hello, world!');
```

Обратите внимание, что строка задана не в двойных кавычках, а в одинарных. В сценариях JavaScript допустимо использовать и те, и другие кавычки, однако закрывающая кавычка должна быть такой же, что и открывающая. Внутренняя пара кавычек должна отличаться от внешней.

Можно устанавливать обработчики самых разных событий, таких, как: загрузка страницы, щелчок по ссылке или кнопке формы, выбор ссылки или поля формы и др. Мы рассмотрим это несколько позже.

Задания:

1. Измените последний пример так, чтобы диалоговая панель возникала не принаведении курсора мыши, а при выборе ссылки (событие onClick).
2. Измените пример так, чтобы при наведении курсора мыши на ссылку, выполнялась бы процедура, выводящая в окно браузера фразу "Hello, word!".
3. Напишите скрипт, который запрашивает у пользователя информацию, а затем выводит ее в диалоговом окне.
4. Составьте документ так, чтобы диалоговое окно для ввода информации предлагалось только после наведения курсора мыши на ссылку, и введенная пользователем текстовая строка выводилась бы в виде диалогового окна, или вокно браузера.