Практическое занятие № 15
" Внедрение JavaScript-кода в HTML-страницу "

Цель работы:

1. Изучение особенностей внедрения JavaScript-кода в HTML-страницу.
2. Получение практических навыков в создании HTML-документов с применением сценариев.

Оборудование и программное обеспечение:

1. ПК, локальная сеть.
2. ОС MS Windows 7, офисный пакет MS Office, браузер MS Internet Explorer.

План занятия:

1. Освоение теоретической части занятия в соответствии с разделами 1 - 5.
2. Выполнение практической части занятия в соответствии с заданиями.
3. Повторение и закрепление основных изученных понятий и терминов.
4. Оформление отчета.

1. Основные практические советы из предшествующего материала

Для того чтобы интегрировать сценарий на HTML страницу, существует специальный тег: <SCRIPT> код сценария ...</SCRIPT>.

Этот тег является контейнером и его можно помещать в любую часть кода HTML страницы. Обычно его помещают в заглавную часть HTML страниц, т.е. между тегами <HEAD> …. </HEAD>.

При внедрении скрипта в HTML страницы важно помнить правила:

- закрывающий тег </script> обязателен;
- в языке javascript, как и в java, различаются прописные и заглавные буквы, так что надписи JavaScript и Javascript - это совершенно разные вещи;
- после каждой строчки следует указывать символ «;».

В остальном браузер обрабатывает сценарий построчно, сам же сценарий будет выглядеть вот так:

```
<script language=”JavaScript”>
alert(“Hello world”);
</script>
```

Если теперь Вы поместите этот код на Вашу HTML страницу, Вы получите сообщение на экран “Hello world”. Параметр language=”javasript” является необязательным (лучше его указать, чтобы Браузер все понял правильно), так же можете указать параметр type=”text/javascript”. И теперь у Вас должно получиться следующее:

```
<script language=”JavaScript” type=”text/javascript”>
alert(“Hello world”);
</script>
```

2. Размещение текста сценария

Текст сценария необходимо помещать вот в такую конструкцию:

```
<script language=”JavaScript” type=”text/javascript”>
<!--
alert(“Hello world”);
//-->
</script>
```

Тогда все браузеры, которые по каким-то причинам не понимают этот сценарий, будут считать, что это обычный комментарий.

Интеграция исходного сценария на HTML страницу не самый лучший вариант, если Ваш сценарий используется в нескольких страницах, то имеет смысл поместить его в отдельный файл и подключать как внешний файл с расширением .js (другие языки имеют другие расширения). Это делается при помощи атрибута SRC все у того же тега <SCRIPT>.

```
<script language=”JavaScript” type=”text/javascript” src=”main.js”>
</script>
```

где **main.js** - это URL адрес файла, содержащего исходный текст самого сценария.

3. Сценарий "КАЛЬКУЛЯТОР" с формой и атрибутом обработчика результата.

```
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Windows-1251">
<META HTTP-EQUIV="Translator" CONTENT="Promt98 Translation System">
<SCRIPT LANGUAGE="JavaScript">
function compute(form) {
  if (confirm("Вы уверены?"))
    form.result.value = eval(form.expr.value)
```

```
  else
    alert("Please come back again.")
}
</SCRIPT>
 </HEAD>
<BODY>
<FORM>  Введите выражение:
<INPUT TYPE="text" NAME="expr" SIZE=15 >
<INPUT TYPE="button" VALUE="Calculate" ONCLICK="compute(this.form)">
 <BR>
Результат:
<INPUT TYPE="text" NAME="result" SIZE=15 >
 <BR>
</FORM>
</BODY>
```

4. Сценарий с формой и атрибутом обработчика результата внутри тега BODY.

```
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Windows-1251">
<SCRIPT LANGUAGE="JavaScript">
<!--- hide script from old browsers
function checkNum(str, min, max) {
   if (str == "") {
      alert("Enter a number in the field, please.")
      return false
   }
   for (var i = 0; i < str.length; i++) {
      var ch = str.substring(i, i + 1)
      if (ch < "0" || ch > "9") {
         alert("Попробуйте еще, пожайлуста.")
         return false
      }
   }
   var val = parseInt(str, 10)
   if ((val < min) || (val > max)) {
      alert("Введите число от 1 до 10.")
      return false
   }
   return true
}
function thanks() {
   alert("Спасибо.")
}
// end hiding from old browsers -->
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="ex5">  Пожалуйста введите маленькое число:
<INPUT NAME="num" onChange="if (!checkNum(this.value, 1, 10)) {this.focus();this.select();}
else {thanks()}">
</FORM>
</BODY>
```

5. Создание Массивов

Массив - упорядоченное множество значений, на которые вы ссылаетесь через имя массива и индекс. Например, у вас есть массив, с именем emp, который содержит имена служащих, индексированные их номером служащего. Так emp [1] будет служащий номер один, emp [2] служащий номер два, и так далее.

JavaScript не имеет явный тип данных массива, но из-за близкой связи между массивами и объектами свойств. Вы можете определять тип объекта массива, следующим образом:

```
function MakeArray(n) {
  this.length = n;
  for (var i = 1; i <= n; i++) {
    this[i] = 0 }
    return this
    }
}
```

Здесь определяется массив такой, что первое свойство, длина, (с индексом ноля), представляет число элементов в массиве. Оставшиеся свойства имеют индекс целого числа один или больше, и инициализированы к нолю.

Вы можете создавать массив вызывая new с именем массива, определяя число элементов, которые имеет. Например:

```
 emp = new MakeArray(20);
```

Здесь создается массив, с именем emp с 20 элементами, и элементы инициализируется к нолю.

Начальная загрузка Массива

Вы можете заполнять массив, просто присваивая значения к ее элементам, например:

```
emp[1] = "Casey Jones"
emp[2] = "Phil Lesh"
emp[3] = "August West"
```

И так далее.

Вы можете также создавать массивы объектов. Например, пусть вы определяете тип объекта, именованный Employees, следующим образом:

```
function Employee(empno, name, dept) {
  this.empno = empno;
  this.name = name;
  this.dept = dept;
}
```

Затем следующие утверждения определяет массив этих объектов:

```
emp = new MakeArray(3)
emp[1] = new Employee(1, "Casey Jones", "Engineering")
emp[2] = new Employee(2, "Phil Lesh", "Music")
emp[3] = new Employee(3, "August West", "Admin")
```

Затем Вы можете легко показывать объекты в этом массиве, используя функцию show_props (определенную в разделе Объектная Модель JavaScript) следующим образом:

```
 for (var n =1; n <= 3; n++) {
  document.write(show_props(emp[n], "emp") + "
");
}
```

6. Выполнение практического занятия:

Вариант 1: Выполните размещение текста сценария JavaScript по условиям, описанным в разделе 2 методического пособия. Реализуйте пример сценария "калькулятор" с формой и атрибутом обработчика результата.

Вариант 2: Выполните размещение текста сценария JavaScript по условиям, описанным в разделе 2 методического пособия. Реализуйте пример сценария с формой и атрибутом обработчика результата внутри тега BODY.

7. Оформление отчета.

Контрольные вопросы к практической работе № 15:

1. Перечислите особенности размещения сценария JavaScript в HTML-странице.
2. Разъясните принцип, на котором основан сценария «калькулятор».
3. Разъясните пример сценария с формой и атрибутом обработчика результата внутри тега BODY.
4. Дайте определение массива