3. JavaScript

Лекция 4. Основные сведения о JavaScript

4.1. Знакомство с JavaScript

JavaScript – объектно-ориентированный сценарный язык программирования.

Области применения:

- динамическое создание web-страницы;
- проверка достоверности полей формы до передачи на сервер;
- вывод сообщений для пользователей;
- учет особенностей браузера и монитора пользователя для корректного отображения;
- составной элемент технологии AJAX;
- счетчики посещаемости.

Сценарий JavaScript встраивается в HTML-документ с помощью тега **<script>**. При этом скрипт может располагаться в произвольном месте страницы.

Пример. Выводит слово «Привет» в теле документа.

```
<html>
<head>
</head>
<body>
<script>
document.write("Привет!");
</script>
</body>
</html>
```

Здесь объект document – это HTML-документ, загруженный в окно браузера. Метод write записывает в тело HTML-документа строку "Привет!".

Переменные.

Можно объявлять с помощью ключевого слова var, а можно использовать без объявления.

```
var th=12
```

age=12
var dd=”qwe”

4.2. Операторы языка JavaScript

В данном подразделе рассматриваются основные операторы языка JavaScript. При этом полный список операторов не ограничивается приведенными.

4.2.1. *Операторы присваивания*

=	Присваивание
+=	Сложение или слияние строк (x=x+2 аналогично x+=2)
–=	Вычитание (x=x-3; аналогично x-=3)
*=	Умножение
/=	Деление

4.2.2. *Операторы сравнения*

>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно

4.2.3. *Унарные операторы*

++	Увеличение значения переменной на единицу. Может применяться как префикс и как суффикс

--	Уменьшение значения переменной. Может применяться как префикс и как суффикс

Чтобы увидеть разницу между использованием унарных операций как префикс и суффикс, изучите работу двух кодов:

```
a=1                          a=1
document.write(a++)          document.write(++a)
```

В первом случае будет выведено значение 1, во втором – 2.

4.2.5. *Условные операторы*

if-else

Пример. Поиск максимального значения из двух чисел

```
if(a<b) {max=b} else {max=b}
```

Последний пример можно реализовать при помощи такой более компактной конструкции:

```
max=(a<b) ?b:a
```

switch - case

Определяет действия в зависимости от значения переменной

```
switch(variable) {
  case value_1: {
    //блок операторов_1
    break;
  }
  case value_2: {
    //блок операторов_2
    break;
  }
  case value_n: {
    //блок операторов_n
    break;
  }
  default: {
    //блок операторов по умолчанию
  }
}
```

4.2.6. *Операторы цикла*

for

Пример вывода пирамидки из горизонтальных линий.

Для этого достаточно набрать код:

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

document.write("<hr width=\""+i*10+"%\">")
```

Таким образом, первый аргумент – это начальное значение, второй – конечное, третий – изменение. В примере использована конструкция вида \", которая называется экранирование. Цель – поместить внутрь строковой переменной кавычки для вывода на экран.

FOR-IN позволяет получить полный список свойств, методов, событий определенного объекта. Например.

```
for(props in document)

document.write(props+"<br>");
```

WHILE

Пример. Вывести таблицу умножения для введенного числа.

```
a=prompt("Введите число",6)
i=1
while(i<=10)
{document.write(a+"x"+i+"="+a*i+"<br>")
i++}
```

Кроме этих операторов в организации цикла могут участвовать еще два оператора: **break** (выход из цикла) и **continue** (переход на следующий шаг).

4.2.7. *Функции*

Функции определяются так:

```
function F_name(аргументы)
{тело функции}
```

Замечания

1. Операторы JavaScript напоминают общеизвестные операторы языка C++.
2. После оператора надо ставить точку с запятой, но если в строке используется только одна команда, то точку с запятой можно не ставить.
3. Для объединения группы операторов используются фигурные скобки.
4. Однострочные комментарии задаются при помощи двух слешей //
5. Многострочные комментарии при помощи конструкции /* в начале и */ в конце.

4.3. Объекты JavaScript

4.3.1. Объект **Array**

Для создания массива можно воспользоваться одним из трех способов:

abc=new Array()

abc=new Array(10)

abc=new Array("Синий","Красный","Зеленый")

объект Array имеет одно свойство **length** – число элементов массива.

Методы объекта Array

- join( ). Слияние элементов массива в строку. Через параметр передается разделитель элементов. По умолчанию разделителем служит запятая.
- reverse( ). Меняет порядок элементов массива на обратный.

Пример. Создание аналога gif-рисунков.

```
<html>
<head>
<title>Создаем анимацию
</title>
<script>
i=0
function ShowNextImage()
{
        iii.src=i + ".jpg"
        i++
        setTimeout ("ShowNextImage()",1000)
        if ((i>=4)) i=0
}
```

```
</script>
</head>
<body>
 <img src="0.jpg" name="iii">
 <script>
 ShowNextImage()
</script>
</body>
</html>
```

В данном примере заранее заготовлены рисунки с названиями 0.jpg … 4.jpg. Если бы мы хотели использовать произвольные названия, то вместо команды

```
iii.src=i + ".jpg"
```

следовало бы использовать команду

```
iii.src=fname[i]
```

где fname – массив содержащий названия файлов.

4.3.2. Объект **Date**

Предоставляет функции для работы с датой и временем.

Создание объекта

```
newDate= new Date()
```

Методы:

getDate() – возвращает число месяца как целое число.

getDay() – возвращает день недели.

getHours()

getMinutes()

getSeconds()

getYear()

getMonth()

getTime() – возвращает количество миллисекунд прошедших с 0:0:0 1 января 1970 года.

Если вместо **get** указать **set**, то метод устанавливает соответствующее значение в переменной. Например, newDate.getYear(2015) становит значение год, равное 2015.

Следующий пример при обновлении страницы выводит число секунд, которые пользователь был на сайте.

```
<html>
<script>
function person_in()
{
        t1=new Date()
}
function person_out()
{
        t2=new Date()
        t3=t2.getTime()-t1.getTime()
        t3=Math.round(t3/1000)
        alert("Вы были на сайте " + t3+ "c")
}
</script>
<body onLoad="person_in()" onUnLoad="person_out()">
</body>
</html>
```

В данном примере использованы две написанные функции:

- person_in(), которая вызывается при открытии страницы – обработчик onLoad записан в теге body. Данная функция сохраняет значение времени в переменной t1.
- person_out(), которая вызывается при открытии страницы – обработчик onUnLoad. В настоящее время большинство браузеров не поддерживает этот обработчик.

4.3.3. Объект Math

Предназначен для работы с математическими константами и функциями. Приведем некоторые свойства и методы.

Свойства: E, PI, SQRT, SQRT1_2.

Методы: cos(), sin(), exp(), abs() – модуль, round() – округление, ceil() - округление вверх, floor() – округление вниз, pow(число1, число2) – возведение в степень, sqrt() – корень квадратный.

4.3.4. Объект **String**

Предназначен для работы со строковыми значениями.

Создать переменную можно двумя способами:

MyString=”значение”

MyString=new String(”значение”)

Одно свойство – length – длина переменной.

Методы (частично):

big() – возвращает строку с добавлением тегов <big>…</big>;

bold() – возвращает строку с добавлением тегов …;

charAt(позиция) – возвращает символ, стоящий на указанной позиции;

fontcolor(“цвет”) – изменяет цвет;

sub(), sup() – индексы;

substring(позиция1, позиция2) – возвращает подстроку, начинающуюся символом в первой позиции и заканчивающуюся перед второй позицией;

toLowerCase() – преобразует исходную строку в строку со строчными символами;

toUpperCase() – преобразует исходную строку в строку с заглавными символами.

Пример. В текстовом элементе формы по надписи пробегает заглавная буква.

```
<form name="f1">
<input type="text" name="t1" size="100">
</form>
<script>
str="Сыктывкарский государственный университет им. Питирима Сорокина"
i=0
function qwe()
{
        i++
        document.f1.t1.value=str.substring(0,i) + str.charAt(i).toUpperCase() + str.substring(i+1, str.length)
        setTimeout("qwe()",300)
}
qwe()
```

</script>

4.3.5. Некоторые встроенные функции JavaScript

- isNaN(значение) – возвращает истину, если значение не является числом;
- parseFloat(строка) – преобразует строку в число с плавающей точкой;
- parseInt(строка,основание) – преобразует строку в целое число по указанному основанию;
- typeOf(объект) – возвращает тип указанного объекта как строку.

Лекция 5. Формы HTML. Обработка форм с помощью JavaScript

5.1. Основные элементы формы

5.1.1. *Форма* – это элемент HTML, позволяющий передавать информацию на web-сервер, где информация будет обработана. С помощью форм организуются тесты, голосования, опросы. Заметим, что html-формы сами по себе позволяют только организовывать ввод информацию. Для обработки форм необходимо использовать языки программирования, для обработки на стороне клиента можно использовать, например, язык JavaScript, а на стороне сервера – например, PHP, Perl, C.

Форма задается тегами **<form></form>.** Все остальные элементы формы располагаются между этими тегами. Тег <form> может иметь несколько параметров:

- name - имя формы. Необходимо, если на странице несколько форм.
- action - определяет URL-адрес, по которому будет отправлена информация введеная пользователем.
- method - определяет способ отправки информации.
- target - указывает имя окна, в котором будут отображаться результаты обработки отправленной формы.

5.1.2. Рассмотрим элементы, располагающиеся в форме. Начнем с *текстового поля*. Оно задается тегом <**input**>. Пример использования.