JavaScript

Какво е JavaScript?

JavaScript е скриптов език, създаден от екипа на Netscape като първоначално е носел името LiveScript. JavaScript именно е език за писане на скриптове и не трябва да се бърка с езика за програмиране JAVA. Освен съвпадение в част от името, двата езика нямат кой знае какви прилики и дори са разработени от различни корпорации (JAVA е разработена от SUN).

JavaScript е проектиран да добавя интерактивност към HTML страници, чрез всякакви ефекти с изображенията, като предоставя възможности за разпознаване на типа на браузъра, опреационната система, разделителната способност на екрана и още, и още много полезни качества.

Как се използва JavaScript?

Вграждането на JavaScript в HTML документ (подобно на CSS) може да се осъществи по два начина:

- чрез поставяне в body-то на документа с тага <SCRIPT> … </SCRIPT>, като опоменем Langugage="JavaScript" Type="text/JavaScript":

```
<script language="javascript"  type="text/javascript">
        document.write("Здравей!") // изписва „Здравей!"
</script>
```

- или чрез връзка към външен файл поставена в секцията head на документа:

```
<script type="text/javascript  src="script.js"></script>
```

*Забележка:

JavaScript е key sensitive език! Прави разлика между малка и голяма буква.

Променливи, Функции, Масиви?

====

Декларирането на променливи в JavaScript не е задължително, но е препоръчително от гледна точка на прегледност на програмата. То става по следния начин, като може да ги изброяваме и да ги инициализираме (желателно е името на промеливата да започва с малка буква):

```
var x;
var carname;
var y=5;
var busnumber=94;
var a,b;
var w=6,z=66,c;
var seaseon=”summer”;
var t=true;
```

Естествено освен числови променливи имаме и текстови, които се представят като текст заграден в кавички, и булеви променливи.

```
var x=10;
var y=5;
var result1=x+y; // резултатът е 15
var a=”10”
var b=5
var result2=x+y // резултатът е 105
```

====

Функциите са основна градивна единица във всеки един език за програмиране. Те ни позволяват да дефинираме име за дадена секция от код и след това да я извикаме с помощта на това име. Използването на функции ни позволява да преизползваме код, като по този начин се намалява вероятността от грешки и да използваме съответния код, само когато се нуждаем от него и го повикаме. Освен това благодарение на тях може програмата да се раздели на ясно

дефинирани секции. Конструкцията на една функция представлява следното:

```
function name (argument 1, argument 2 , ...)
{
//code ...;
}
```

Function е задължителна служебна дума, а name е името на функцията, което трябва да отговаря на общите правила за наименование на идентификатори. В кръглите скоби – () се записват параметрите на функцията, ако има такива. Но дори и да няма е задължително изписването на самите скоби. И тялото на функцията – кодът, който искаме да изпълним се записва със фигурни скоби – {}.

Самото извикване на функция се извършва чрез изписване на името ѝ (със съответните аргументи) в body-то на документа.

Няколко простички примера:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function firstName() {
name=prompt("What is ypur first name?")
document.write("Hallo, "+name+"!")
}
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE="Javascript">
firstName()
</SCRIPT>
</BODY>
</HTML>
```

В head-а на документа се дефинира функция firstName без аргументи, която присвоява на променлива name, текстът който ще бъде въведен в отговор на въпроса „Как се казвате?“(примерно - Иван) и след което ще бъде изписан поздрав с обръщение към въведения текст: „Здравейте, Иван!“.

Сега да напишем и една функция с параметри:

```
<HTML
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function fontSize(num) {
document.write("<font size="+num+">")
document.write(num)
document.write("</font>")
}
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE="Javascript">
for(num=1;num<=7;num++) {
fontSize(num)
}
</SCRIPT>
</BODY>
</HTML>
```

Тук имаме функция fontSize с аргумент num, който приема стойности от 1 до 7 и със всяка своя стойност се изпълява от функцията. Самата функцияа изписва съответната стойност в еквивалентната за нея големина на шрифта в HTML. Това е осъществено благодарение на <font size=“ ”> (HTML таг с атрибут) на който се задава стойността на num. И съответно при всяко извикване на функцията се изписват последователно числата от 1 до 7 с нарастваща големина на шрифта.

====

Представете си сега, че имате голям брой еднотипна информация, например оценките на студентите от курса по Електронно обучени по дадено задание. Използването на отделни променливи за всяка оценка би затруднило неимоверно много работата с тях. В такива случай се използва структурата от данни – Масив. Той може да се възприема като вид променлива, който съдържа множество еднакви елементи. Тези елементи могат да бъдат достъпвани чрез тяхния пореден номер в масива. Особеното е, че номерацията на масивите започва от 0.

Може да дефинираме масив по няколко начина:

```
var colors=new Array(„червен", „зелен", „син")


var week=new Aray(7)
week[0]= „Понеделник";
week[1]= „Вторник";
week[2]= „Сряда";
week[3]= „Четвъртък";
week[4]= „Петък";
week[5]= „Събота";
week[6]= „Неделя";
```

Тъй като масивът е обект (материал за обект – в бъдеще!), той се задава посредством запазените думи new Array. Представените масиви са едомерни, но могат да се създават и дву-, три- и повече мерни, стига да са адекватни и полезни. Нека демонстрираме и двумерен масив.

```
var red=new Array(„червен", „ябълка");
var green=new Array(„зелен", „грозде");
var yellow=new Array(„жълт", „банан");
var colors=new Array(red, green, yellow);
```

Като достъпа до елементите се извършва съответно с два индекса, като първият показва елементът от colors (например [1] е зелен), а вторият – съответният поделемент (например [2][0] е жълт):

```
document.write(colors[0][1]); // изписва „ябълка"
document.write(colors[1][1]); //изписва „грозде"
```

Тъй като масивът е обект(!!!). Като такъв той предоставя няколко метода за работа с него(примерно *document* е обект, а *write* е метод ... и метода се използва, като се запише с точка след обекта):

```
join(разделител) – връща всички елементи на масива, разделени със съответия разделител
reverse() – обръща реда на елементите на масива в обратен ред
sort() – сортира елементите
length() – измерва дължината на масива
concat() – събира два низа в един, залепва ги!
```

Пример за използване на масиви и техните методи:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/JavaScript">

var array1 = new Array(2,4,6,1,8);
document.write("Първи масив:"+array1.join(",")+"<BR\>");
document.write("Масивът има:"+array1.length+" елемента.<BR\>");

array1.sort();
document.write("Подреденият масив е:"+array1.join(",")+"<BR\>");

array1.reverse();
document.write("Обърнатият масив е:"+array1.join(",")+"<BR\>");

var array2 = new Array(9,10,11,34);
document.write("Втори масив:"+array2.join(",")+"<BR\>");

var array3 = array1.concat(array2);
document.write("Обединеният масив е:"+array3.join(",")+"<BR\>");

</SCRIPT>
</BODY>
</HTML>
```

Тъй като засегнахме (и няма как да не засегнем) термините на обектно-

ориентирано програмиране, то JavaScript не е изцяло такъв вид език, а по-скоро е обектно-базиран такъв, тъй като за него са присъщи някои от важните свойства на ООП, но не всички (не се поддържа наследяване, класифициране, капсулиране, скриване на информация). В общи линии обект се състои от две неща:

- Набор от свойства. Те съдържат данните.
- Методи, които извършват определени дейности.

Нека разгледаме няколко полезни примера:

Обект document предоставя методи за достъп до елементите на страницата. Като особенно полезен е метода .getElementById, който значи „намери ми елемента по зададено ID". Чрез тази конструкция document.getElementById може да се направят много интересни неща, дори и да не се знае много повече от възможностите на JavaScript, като просто се подменят ID-то и/или класа на избрания елемент, което го кара да изглежда по коренно различен начин.

====

Съществен и често използван елемент е alert(). Как работи той?

```
alert("Hallo, World!");
```

Този пример ще изведе съовщението „Hallo, World!" в нов диалогов прозорец, на фокуса на текущия и няма да позволи да продължи работата с браузъра, докато не се отбележи, че съобщението е прочетено.

Това средство е изклчително полезно в комбинация със т.нар. събития в JavaScript – events. Какво представляват те? Ами, най-общо казано изобщо всичко е събитие. JavaScript обработва събитията с така наречените манипулатори на събития. Така например при кликване на мишката ще се извика манипулатора onClick(), при посочване чрез мишката - onMouseover(), когато мишката се отмети от определен обект – onMouseout(). Разбира се има и манипулатори, които не са свързани със събитие на мишката. Например събития с отваряне и затваряне на прозорци. Когато се зареди един прозорец, то подходящия манипулатор е onLoad().

```
<HTML>
<BODY onLoad="alert('Здравейте!')">
</BODY>
</HTML>
```

Веднага след зареждането на страницата, ще се покаже прозорец с поздрав „Здравейте!".

Събитията в JavaScript биват доста видове ... и трудно биха могли да бъдат обхванати само в един урок ... затова ние ще спрем до тук, надявайки сме да сме породили интересът на учащия се ... който да го окрили ... да се запознае със още и още, и още ... от възможностите, предоставени от JavaScript.