JS基础简介及入门

- JavaScript 是脚本语言
- JavaScript 是一种轻量级的编程语言。
- JavaScript 是可插入 HTML 页面的编程代码。
- JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。
- JavaScript 是大小写敏感的

废多看码:

- `HTML` 中的脚本必须位于 `<script>` 与 `</script>` 标签之间。
- 脚本可被放置在 `HTML` 页面的 `<body>` 和 `<head>` 部分中。
- 通常的做法是把函数放入 `<head>` 部分中，或者放在页面底部。这样就可以把它们安置到同一处位置，
不会干扰页面的内容。

在 `HTML` 页面中嵌入 `JavaScript` 需要使用 `<script>` 标签。`<script>` 和 `</script>` 来告诉 `JavaScript` 的开始和结束。之间的代码就是 `JavaScript`：

```
<script>
alert("Hello World!");
</script>
```

或者也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。扩展名是 `.js`。
如需使用外部文件，请在 `<script>` 标签的 `src` 属性中设置该 `.js` 文件：

```
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Note:有些代码中 `<script>` 标签中使用 `type="text/javascript"`。现在已经不必这样做了，`JavaScript`

是所有现代浏览器以及`HTML5`中的默认脚本语言。

- 输出内容

  ```
  document.write("are you ok?");
  ```

  document是当前的窗口对象，大学老师好像是这么教的，我还有点印象。

- 响应事件

  ```
  <button type="button" onclick="alert('Are you ok?')">点我试试?
  </button>
  ```

  `type`属性指明这是一个`buttton`

  `onclick`指定事件的相应

  `alert()`方法就是弹出对话框。

- 变量
  使用`var`关键词来声明变量

  ```
  x = 5;
  或
  var x = 5;
  或
  var x = "James";
  ```

  一条语句声明多个变量，中间用逗号分割。

  ```
  var name="Gates",
  age=56,
  job="CEO";
  ```

如果只生命了一个变量，但是没有复制，这时候它的值是`undefined`；

```
var name; // 这时候变量name的值将是 undefined
```

```
var carname="Volvo";
var carname;
```

如果重新声明 `JavaScript` 变量，该变量的值不会丢失
在以上两条语句执行后，变量 `carname` 的值依然是 `Volvo`

- Undefined 和 Null
  `Undefined` 这个值表示变量不含有值。
  可以通过将变量的值设置为 `null` 来清空变量。
  JS 一声明变量就有内存分配。

  ```
  x = null;
  ```

- `JavaScript` 是弱类型的,这意味这相同的变量可用作不同类型:

  ```
  var x // x 为 undefined
  var x = 6; // x 为数字
  var x = 6.00;
  var x = "Bill"; // x 为字符串
  ```

- 布尔

  ```
  var x = true;
  var y = true;
  ```

- 数组

  ```
  var array = new Array();
  array[0] = "are";
  array[1] = "you";
  array[2] = "ok";
  ```

  或

```
var array = new Array("are", "you", "ok")
```

或

```
var array = ["are", "you", "ok"];
```

- 声明变量类型
  当您声明新变量时,可以使用关键词 `new` 来声明其类型

```
var carname=new String;
var x= new Number;
var y= new Boolean;
var cars= new Array;
var person= new Object;
```

`JavaScript` 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

- 对象
  对象由花括号分割，在括号内部，对象的属性以名称和值的形式出现，属性之间由逗号分割:

```
var person={
firstname : "Bill",
lastname : "Gates",
id : 5566
};
```

属性调用:

```
name = person.firstname;
```

或

```
name = person["firstname"];
```

- 创建对象
  创建 `person` 对象，并为其添加四个属性:

```
person=new Object();
```

```
person.firstname="Bill";
person.lastname="Gates";
person.age=56;
person.eyecolor="blue";
```

- 访问对象的属性
  访问对象属性的语法是：
  `objectName.propertyName`
  本例使用 `String` 对象的 `length` 属性来查找字符串的长度：

```
var message="Hello World!";
var x=message.length;
```

- 访问对象的方法
  您可以通过下面的语法调用方法：

  `objectName.methodName()`
  这个例子使用 `String` 对象的 `toUpperCase()` 方法来把文本转换为大写：

```
var message="Hello world!";
var x=message.toUpperCase();
```

- 方法

```
function myFunction(name, job) {
    document.getElementById("name").innerHTML = name;
    document.getElementById("job").innerHTML = job;
    // 如果需要返回值，直接加上下面的return
    var x = 5;
    return x;
}
```

- 注释
  - 单行注释 //

    - 多行注释 /* */

- 局部变量及全局变量

    - 在`JavaScript`函数内部声明的变量(使用`var`)是局部变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。
    - 在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问它。

  变量的生命期从它们被声明的时间开始
  - 局部变量会在函数运行以后被删除。
  - 全局变量会在页面关闭后被删除。

- 向未声明的变量来分配值
  如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

  ```
  name = "james";
  ```

  将声明一个全局变量`name`,即使它在函数内执行。

- 运算符

    - +
    - -
    - *
    - /
    - %
    - ++ - --
    - =
    - +=
    - -=
    - *=
    - /=
    - %=
    - ==
    - === 全等 类型和值都一样
    - !=
    - >
    - <
    - >=

    - <=
    - &&
    - ||
    - ! 非
    - `+`运算符用于把数字或者字符串加起来(连接起来).

- 三元运算符

```
variablename=(condition)?value1:value2
```

- 条件语句

```
if (条件) {

} else if (条件) {

} else {

}
```

- Switch语句

```
var day=new Date().getDay();
switch (day) {
case 6:
  x="Today it's Saturday";
  break;
case 0:
  x="Today it's Sunday";
  break;
default:
  x="Looking forward to the Weekend";
}
```

- 循环

```
for (var i=0; i<5; i++) {
    x=x + "The number is " + i + "<br>";
}
```

或

```
var person={fname:"John",lname:"Doe",age:25};
for (x in person) {
    txt=txt + person[x];
}
```

或

```
while (条件) {
    需要执行的代码
}
```

或

```
do {
    需要执行的代码
}
while (条件);
```

循环过程中也都支持 `break` 和 `continue`， 这和 `java` 类似。

- 标签

  ```
  cars=["BMW","Volvo","Saab","Ford"];
  list: {
  document.write(cars[0] + "<br>");
  document.write(cars[1] + "<br>");
  document.write(cars[2] + "<br>");
  break list;
  document.write(cars[3] + "<br>");
  document.write(cars[4] + "<br>");
  document.write(cars[5] + "<br>");
  }
  ```

  `list` 就是标签或者别名。

- 异常的处理

  ```
  try {
  //在这里运行代码
  } catch(err) {
  //在这里处理错误
  }
  ```

  如:

```
<script>
function myFunction()
{
try
  {
  var x=document.getElementById("demo").value;
  if(x=="")    throw "empty";
  if(isNaN(x)) throw "not a number";
  if(x>10)     throw "too high";
  if(x<5)      throw "too low";
  }
catch(err)
  {
  var y=document.getElementById("mess");
  y.innerHTML="Error: " + err + ".";
  }
}
</script>

<h1>My First JavaScript</h1>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="mess"></p>
```

- 对话框

三种弹出框:警告(alert)、确认(confirm)以及提问(prompt)。

```
alert("我是菜鸟我怕谁");
```

```
var r = confirm("你是菜鸟吗");
if (r == true)
{
    document.write("彼此彼此");
} else
{
    document.write("佩服佩服");
}
```

`prompt` 和 `confirm` 类似，不过它允许访客随意输入答案。

```
var score;
score = prompt("你的分数是多少？")
```

- 五种原始类型:
    - Undefined
    - Null
    - Boolean
    - Number
    - String
- typeof运算符

```
alert (typeof sTemp);    //输出 "string"
alert (typeof 86);    //输出 "number"
```

对变量或值调用 typeof 运算符将返回下列值之一：

- undefined - 如果变量是 Undefined 类型的
- boolean - 如果变量是 Boolean 类型的
- number - 如果变量是 Number 类型的
- string - 如果变量是 String 类型的
- object - 如果变量是一种引用类型或 Null 类型的

- instanceof 运算符 在使用 `typeof` 运算符时采用引用类型存储值会出现一个问题，无论引用的是什么类型的对象， 它都返回 `object` 。引入了另一个 `Java` 运算符 `instanceof` 来解决这个问题。

最后看下这个小`demo`，为下面的`DOM`作铺垫。

```
<!DOCTYPE html>
<html>
<body>

<h1>我的第一段 JavaScript</h1>

<p id="demo">
JavaScript 能改变 HTML 元素的内容。
</p>

<script>
function myFunction()
{
x=document.getElementById("demo");  // 找到元素
x.innerHTML="Hello JavaScript!";    // 改变内容
}
</script>

<button type="button" onclick="myFunction()">点击这里</button>

</body>
</html>
```

- 邮箱： zhu_tian_cheng@126.com

- Good Luck!