javaScript

# 引入js

Js代码最好写在html的head中

## 第一种引入方式

|  |
| --- |
| <script type=*"text/javascript"*>  </script> |

Js代码必须写在脚步标签之内。

## 第二种引入方式

引入外部的js文件

|  |
| --- |
| <script type=*"text/javascript"* src=*"js/js\_demo1.js"*></script> |

如果引入了外部的js文件就不能在这个js的脚本标签内些js代码了，整个工程的编码使用UTF-8

# 2.JavaScript的语法

* 区分大小写：与Java一样。变量、函数名、运算符以及其他一切东西都是区分大小写的。
* 变量是弱类型的：与Java不一样。JavaScript中的变量无特定的类型，定义变量时只用var 运算符，可以将它初始化为任意的类型。变量的类型取决于给变量赋值的内容。
* 每行结尾的分号可有可无：与Java不一样。如果一条语句的结尾没有分号表明结束的话，前提是这样没有破坏代码的语义。最好的编写代码习惯是加入分号。
* 注释：与Java一样。支持”//”、”/\*\*/”这几种常见的注释方式

{}括号表明代码块：与Java一样。

# 变量

变量的定义：var 变量名字 = 变量的值；

Java中局部变量必须赋给初始化，但是在js中不是必须的。

Js在定义的时候不指定具体的数据类型，而是根据具体赋给的值来决定是什么类型。

|  |
| --- |
| <script type=*"text/javascript"*>  **var** age = 10;  **var** name = 'renliang';  **var** isMarry = **false**;  document.write(age + "数据类型是:"+(**typeof** age)+"<br>");  document.write(name + "数据类型是:"+(**typeof** name)+"<br>");  document.write(isMarry + "数据类型是:"+(**typeof** isMarry)+"<br>");  </script> |

Js变量的特点

|  |
| --- |
| <script type=*"text/javascript"*>  **var** age = 10;  **var** name = 'renliang';  **var** isMarry = **false**;  document.write(age + "数据类型是:"+(**typeof** age)+"<br>");  document.write(name + "数据类型是:"+(**typeof** name)+"<br>");  document.write(isMarry + "数据类型是:"+(**typeof** isMarry)+"<br>");  document.write("<hr>");  //js的变量的数据类型随着值的改变而改变  age = **true**;  document.write(age + "数据类型是:"+(**typeof** age)+"<br>");  //js中变量可以重复，下面的变量会把上面重复的变量给覆盖  **var** name = '拓薪教育';  document.write(name + "数据类型是:"+(**typeof** name)+"<br>");  </script> |

# 数据类型

* 原始值，是存储在栈中的简单数据段，也就是说，它们的值直接存储在变量访问的地址。
* 引用值，是存储在栈中的对象，也就是说，存储在变量出的值是一个指针，指向存储对象的内存处。

## 原始值

五种原始类型，即Undefined、Null、Boolean、Number和String。

Number：代表是数值类型，包括java中的6中数值类型，不管是整数还是小数都是使用number类型。

|  |
| --- |
| <script type=*"text/javascript"*>  **var** price = 100.12;  document.write("price:"+price+" 数据类型："+(**typeof** price) +"<br>");  //定义数值类型的包装类  **var** age = **new** Number(20);  document.write("age:"+age+" 数据类型："+(**typeof** age)+"<br>");  //包装类可以获取最大值和最小值，相当于我们java中的访问static的变量，  //使用对象虽然在eclipse中能通过.来提示但是不能访问  document.write(" 数值类型的最大值："+Number.MAX\_VALUE);  document.write(" 数值类型的最小值："+Number.MIN\_VALUE);  </script> |

String类型：字符串类型，在js中没有字符类型，字符串类型可以使用‘’也可以使用“”，建议使用‘’

|  |
| --- |
| <script type=*"text/javascript"*>  **var** name = 'www.txjava.cn';  document.write("name:"+name+" 数据类型："+(**typeof** name) +"<br>");  **var** name1 = **new** String('tuoxin');  document.write("name1:"+name1+" 数据类型："+(**typeof** name1) +"<br>");  **var** xindex = name.indexOf("x");  document.write(" x的索引："+xindex+"<br>");  **var** strsub = name.substring(3, 8);  document.write(" 切割的字符串："+strsub+"<br>");  document.write(" 字符串的长度："+name.length+"<br>");  </script> |

Boolean类型：一共两个值true和false和java一样的。

**Undefined类型**

Undefined类型只有一个值，即undefined。当声明的变量未初始化时，该变量的默认值是undefined

当函数无明确返回值时，返回的也是值undefined

* **Null类型**

Null类型也只有一个值null，即它的字面量。值undefined实际上是从值null派生来的，因此把它们定义为相等的。

alert(null == undefined) //输出 true

尽管这两个值相等，但它们的含义不同。undefined是声明了变量但未对其初始化，null则用于表示尚未存在的对象，比如无法获得文档中的对象。

|  |
| --- |
| alert(**null** == undefined);  **var** doc = document.getElementById("div2");  alert(doc)  </script>  </head>  <body>  <div id=*"div1"*></div> |

页面加载的顺序是从上向下的，所以js是不能使用在他下面的html的。

## 引用值

**Math**

**Math** 对象不能用 **new** 运算符创建，如果试图这样做则给出错误。在装载脚本引擎时由该引擎创建该对象。其所有方法和属性在脚本中总是可用。

**Date**

|  |
| --- |
| //创建时间类型 对象  **var** birthday = **new** Date();  //日期设置  birthday.setDate(22);  //日期获取  document.write(birthday.getFullYear()+"年");  document.write((birthday.getMonth() + 1)+"月");  document.write(birthday.getDate()+"日");  document.write(birthday.getHours()+"时");  document.write(birthday.getMinutes()+"分");  document.write(birthday.getSeconds()+"秒");  document.write(birthday.getMilliseconds()+"毫秒"); |

定义数组

数组对象用来在单独的变量名中存储一系列的值。

我们使用关键词 new 来创建数组对象。下面的代码定义了一个名为 myArray 的数组对象：

注意：数组的长度是不固定的，可以自动扩展，而且我们不需要指定数组的长度。

|  |
| --- |
| <script type=*"text/javascript"*>  **var** arr = **new** Array();  arr[0] = 1;  arr[1] = 23;  arr[2] = 55;  **for**(**var** i = 0; i < arr.length; i++){  document.write(arr[i]+"<br>");  }  //在js中数组的长度不固定，和java中的集合一样  **var** arr = **new** Array('a', 'hgh', 'ghh');  arr[3] = 'ab';  arr[4] = 'abc';  **for**(**var** i = 0; i < arr.length; i++){  document.write(arr[i]+"<br>");  }  </script> |

#### Global

#### isNaN():判断是否不是一个数字，是一个能转成数字的类型则返回false，如果是一个不能转换成数字的是true

#### parseInt():把数据变成整数

#### parseFloat():把数据变成小数

|  |
| --- |
| <script type=*"text/javascript"*>  **var** num = '60.2';  //isNaN false是不是数值，true是数值,只要是能转换成数值类型的变量的值都认为是数值。  alert(isNaN(num));  //把字符串转换成数值  **var** numb = **new** Number(num);  alert(numb + 10);  //使用global来做数值转换，如果是小数那么会自动取整  numb = parseInt(num);  alert(numb + 10);  //把字符串类型的小数转换成数值类型的小数  numb = parseFloat(num);  alert(numb + 10);  </script> |

# js的方法定义

**语法:**

**function 方法名(参数...){**

**return xxx；//可选**

**}**

**var 方法名 = function (参数...){**

**return xxx；//可选**

**}**

**方法同名不管是哪种语法都会把上面的覆盖掉**

|  |
| --- |
| <script type=*"text/javascript"*>  /\*\*  \* 方法的定义  \*/  **function** showInfo(){  alert("www.txjava.cn");  }  **function** showInfo(){  alert("JavaScript method");  }  /\*\*  \* 变量的名字就是方法的名字  \*/  **var** showInfo1 = **function**(){  alert("拓薪教育");  }  **var** showInfo1 = **function**(){  alert("大家好");  }  //alert(typeof showInfo1)  //方法的调用  //showInfo();  </script> |

方法中没有重载

js中没有重载，如果方法名字相同，后面的方法会把前面的方法覆盖掉,我们在调用方法的时候，我们传递的参数可以少于形参，从左向右赋值，如果调用时实参多于形参，也是从左向右赋值，所出来的实参就没有用

|  |
| --- |
| <script type=*"text/javascript"*>  /\*\*  \* 方法的定义,方法的参数不需要指定类型，如果方法有返回值使用return加上返回值  \*/  **function** showInfo(a, b){  alert("方法1");  **return** a + b;  }  /\*\*  \* js中没有重载，如果方法名字相同，后面的方法会把前面的方法覆盖掉,我们在调用方法的时候，我们传递的参数可以少于形参，从左向右复制  \*/  **function** showInfo(a, b, c){  alert("方法2"+ c);  **return** a + b + c;  }  /\* var showInfo = function(a, b){  return a + b;  }  var showInfo = function(a, b, c){  return a + b + c;  } \*/  **var** result = showInfo(5, 5);  alert(result);  </script> |

Function对象

|  |
| --- |
| //方法对象是每一个方法定义的的底层写法，第一个参数是字符串类型，方法的每一个参数用逗号分隔，第二个参数是方法体  **var** fun = **new** Function("a, b", "return a + b");  **var** result1 = fun(2, showInfo(4, 6));  alert(result1) |

# Object对象

创建对象的第一种方式：

|  |
| --- |
| <script type=*"text/javascript"*>  /\*\*  \* 根据object来创建一个对象  \*/  **function** createPerson(username, password){  **var** obj = **new** Object();  //定义对象的属性  obj.username = username;  obj.password = password;  //定义对象的方法  obj.showInfo = **function**(){  alert(**this**.username + " "+ **this**.password);  }  **return** obj;  }  **var** person = createPerson("renliang", "123");  //alert(person.username + " " + person.password);  //调用对象的方法  person.showInfo();  </script> |

第二种方式：

|  |
| --- |
| <script type=*"text/javascript"*>  //创建以键值对的方式的对象，我们也管它叫json对象  **var** person = {username : "zhangsan", age:30, address:'北京'};  //alert(typeof person);  alert(person.address + " "+person.username);  </script> |

第三种方式：

|  |
| --- |
| <script type=*"text/javascript"*>  **function** showInfo(){  alert(**this**.username + " "+ **this**.password);  }  /\*\*  \* 根据object来创建一个对象  \*/  **function** createPerson(username, password){  **var** obj = **new** Object();  //定义对象的属性  obj.username = username;  obj.password = password;  //引入外部的方法作为对象方法  obj.showInfo = showInfo;  //定义对象的方法    **return** obj;  }  **var** person = createPerson("renliang", "123");  //alert(person.username + " " + person.password);  //调用对象的方法  person.showInfo();  </script> |

创建对象第四种方式：

|  |
| --- |
| <script type=*"text/javascript"*>  **function** Person(username , password){  //通过this来直接指定属性，不需要创建Object  **this**.username = username;  **this**.password = password;  **this**.showInfo = **function**(){  alert(**this**.username + " " + **this**.password);  }  //不需要return，默认隐含了return  }  **var** person = **new** Person("lisi", "323");  person.showInfo()  </script> |

创建对象第五种方式：推荐使用

|  |
| --- |
| <script type=*"text/javascript"*>  **function** Person(username, age){  **this**.username = username;  **this**.age = age;  }  /\* function User(){    } \*/  //以原型方式来赋值方法  Person.prototype.showInfo = **function**(){  alert(**this**.username + " " + **this**.age);  }  /\* function showInfo(){  alert(this.username + " " + this.age);  } \*/  **var** person = **new** Person("zhangsan", 23);  person.showInfo();  **var** user = **new** User();  user.showInfo()  </script> |

# dom

当标记型文档被DOM解析器解析后，会按照标签的层次关系，产生一颗dom树结构。

这个树的每个分支被称为一个节点。

我们就是研究如何获取每个节点，并对节点进行操作的。

C：而浏览器本身具备了DOM解析器,所以可以对标记型文本进行DOM解析.

Window 对象代表一个浏览器窗口或一个框架。

控制浏览器的地址：

|  |
| --- |
| <script type=*"text/javascript"*>  //当前浏览器窗口的地址  **var** url = window.location.href;  **function** turn(){  //修改浏览器的地址  window.location.href = "http://www.txjava.cn";  }  </script> |

## dom获得

文档对象的加载顺序是从上到下的，所以我们js想要获得dom对象，一定要等到文档对象加载完毕在获得

### 根据元素的id获得dom对象

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <script type=*"text/javascript"*>    **function** myclick(){  //根据文档对象的id来获得dom对象  **var** obj = document.getElementById("username");    alert(obj);  //获得用户名的值  **var** username = obj.value;  alert(username);  //给input设置值  obj.value = "拓薪教育";  //设置元素的样式  obj.style.background = "yellow";  //获得password的对象  **var** obj1 = document.getElementById("password");  alert(obj1.value);  }  </script>  </head>  <body>  用户名：<input id=*"username"* type=*"text"* name=*"username"* value=*"renliang"*><br>  密码：<input id=*"password"* type=*"password"* name=*"password"* value=*"123"*>  <input type=*"button"* value=*"点击"* onclick="myclick()">  </body>  </html> |

### 2.根据元素的name来获得dom对象

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <script type=*"text/javascript"*>    **function** myclick(){  //根据元素的name来获得元素，返回的是数组  **var** favors = document.getElementsByName("favor");  **for**(**var** i = 0; i < favors.length; i++){  //获得数组中的每一个元素  **var** f = favors[i];  //alert(f.value);  //alert(f.checked);  /\* 原生的js中获得多选的的checked属性是true而不是checked  if(f.checked == "checked"){  alert(f.value);  }  \*/    **if**(f.checked){  alert(f.value);  }  }  }  **function** chkAll(){  **var** favors = document.getElementsByName("favor");  **for**(**var** i = 0; i < favors.length; i++){  //获得数组中的每一个元素  **var** f = favors[i];  f.checked = **true**;  }  }  **function** reverChkAll(){  **var** favors = document.getElementsByName("favor");  **for**(**var** i = 0; i < favors.length; i++){  //获得数组中的每一个元素  **var** f = favors[i];  **if**(f.checked){  f.checked = **false**;  }**else**{  f.checked = **true**;  }  }  }  </script>  </head>  <body>  爱好：<input type=*"checkbox"* name=*"favor"* value=*"1"*>看电影  <input type=*"checkbox"* name=*"favor"* value=*"2"*>打台球  <input type=*"checkbox"* name=*"favor"* value=*"3"*>处对象  <input type=*"button"* value=*"点击"* onclick="myclick()">  <input type=*"button"* value=*"全选"* onclick="chkAll()">  <input type=*"button"* value=*"反选"* onclick="reverChkAll()">  </body>  </html> |

### 3.根据元素标签名获得dom对象

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <script type=*"text/javascript"*>    **function** myclick(){  //根据元素的标签名获得dom对象，返回的是数组  **var** objs = document.getElementsByTagName("input");  **for**(**var** i = 0; i < objs.length; i++){  alert(objs[i].value);  }  }  </script>  </head>  <body>  用户名：<input id=*"username"* type=*"text"* name=*"username"* value=*"renliang"*><br>  密码：<input id=*"password"* type=*"password"* name=*"password"* value=*"123"*>  <input type=*"button"* value=*"点击"* onclick="myclick()">  </body>  </html> |

## innerText和innerHTML

innerText和innerHtml的设置

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <style type=*"text/css"*>  *.myclass*{  border: *1px solid black*;  width: *100px*;  height: *100px*;  }  </style>  <script type=*"text/javascript"*>  **function** myclick(){  document.getElementById("div1").innerText = "我是div1";  //innerText是指的元素中的 内容，如果有html不会被浏览器解析  document.getElementById("div2").innerText = "<font color='red'>我是div1</font>";  //innerText是指的元素中的 内容，如果有html会被浏览器解析  document.getElementById("div3").innerHTML = "<font color='red'>我是div1</font>";      }  </script>  </head>  <body id=*"mybody"*>  <div id=*"div0"*>  <div id=*"div1"* class=*"myclass"*></div>  <div id=*"div2"* class=*"myclass"*></div>  <div id=*"div3"* class=*"myclass"*></div>  </div>  <input type=*"button"* value=*"点击"* onclick="myclick()">  </body>  </html> |

innerText和innerHtml的获得

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <style type=*"text/css"*>  *.myclass*{  border: *1px solid black*;  width: *100px*;  height: *100px*;  }  </style>  <script type=*"text/javascript"*>  **function** myclick(){  //innerText获取的时候只能拿到元素内的除了html元素外的文本内容  **var** t1 = document.getElementById("div2").innerText;  alert(t1);  //获得到元素内的所有内容  **var** t2 = document.getElementById("div3").innerHTML;  alert(t2);    }  </script>  </head>  <body id=*"mybody"*>  <div id=*"div0"*>  <div id=*"div1"* class=*"myclass"*></div>  <div id=*"div2"* class=*"myclass"*>aaa<font color=*'red'*>我是div1</font></div>  <div id=*"div3"* class=*"myclass"*>aaa<font color=*'red'*>我是div2</font></div>  </div>  <input type=*"button"* value=*"点击"* onclick="myclick()">  </body>  </html> |

## 删除一行

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <style type=*"text/css"*>  *.mytable*{  width: *300px*;  border-collapse: *collapse*;  }    **tr**{  height: *25px*;    }  </style>  <script type=*"text/javascript"*>  **function** addrow(){  //获得表体的dom对象  **var** tbObj = document.getElementById("tb");  //创建 tr dom对象  **var** trObj = document.createElement("tr");  //创建td dom对象  **var** td0 = document.createElement("td");  //创建一个input类型的多选  **var** ck = document.createElement("input");  //设置属性  ck.setAttribute("type", "checkbox");  //把ck的dom对象追加到td0内部  td0.appendChild(ck);    **var** td1 = document.createElement("td");  **var** td2 = document.createElement("td");  **var** td3 = document.createElement("td");  //把3个td追加到tr里面  trObj.appendChild(td0);  trObj.appendChild(td1);  trObj.appendChild(td2);  trObj.appendChild(td3);  //把tr对象追加到tb里面  tbObj.appendChild(trObj);    }  **function** deleteRow(){  //获得表体的dom对象  **var** tbObj = document.getElementById("tb");  //获得所有的表体的内部的input  **var** is = tbObj.getElementsByTagName("input");  **for**(**var** i = is.length-1; i > 0; i--){  //获得被选中的多选  **if**(is[i].checked){  //删除被选中的元素，获得父元素  **var** trObj = is[i].parentNode.parentNode;  tbObj.removeChild(trObj);  }  }      }  </script>  </head>  <body id=*"mybody"*>  <input type=*"button"* value=*"添加一行"* onclick="addrow()">  <input type=*"button"* value=*"删除"* onclick="deleteRow()">  <table border=*"1"* class=*"mytable"*>  <thead>  <tr>    <td></td>  <td>姓名</td>  <td>年龄</td>  <td>性别</td>  </tr>  </thead>  <tbody id=*"tb"*>  <tr>  <td><input type=*"checkbox"*></td>  <td>任亮</td>  <td>18</td>  <td>男</td>  </tr>  </tbody>  </table>  </body>  </html> |

## 4替换

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <style type=*"text/css"*>  *.myclass*{  border: *1px solid black*;  width: *100px*;  height: *100px*;  }  </style>  <script type=*"text/javascript"*>  **function** myclick(){    **var** parentDom = document.getElementById("div0");    //获得div3的dom  **var** div1Dom = document.getElementById("div1");  **var** div3Dom = document.getElementById("div3");    parentDom.replaceChild(div3Dom, div1Dom);  }  </script>  </head>  <body id=*"mybody"*>  <div id=*"div0"*>  <div id=*"div1"* class=*"myclass"*>div1</div>  <div id=*"div2"* class=*"myclass"*>aaa<font color=*'red'*>我是div2</font></div>  <div id=*"div3"* class=*"myclass"*>aaa<font color=*'red'*>我是div3</font></div>  </div>  <input type=*"button"* value=*"替换"* onclick="myclick()">  </body>  </html> |

## 5克隆

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <style type=*"text/css"*>  *.myclass*{  border: *1px solid black*;  width: *100px*;  height: *100px*;  }  </style>  <script type=*"text/javascript"*>  **function** myclick(){    **var** parentDom = document.getElementById("div0");    //获得div3的dom  **var** div1Dom = document.getElementById("div1");  //复制一个dom对象  **var** cn = div1Dom.cloneNode(**true**);  parentDom.appendChild(cn);  }  </script>  </head>  <body id=*"mybody"*>  <div id=*"div0"*>  <div id=*"div1"* class=*"myclass"*>div1</div>  <div id=*"div2"* class=*"myclass"*>aaa<font color=*'red'*>我是div2</font></div>  <div id=*"div3"* class=*"myclass"*>aaa<font color=*'red'*>我是div3</font></div>  </div>  <input type=*"button"* value=*"克隆"* onclick="myclick()">  </body>  </html> |

# 事件

## 如何给dom对象赋予事件

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Insert title here</title>  <script type=*"text/javascript"*>    **function** myclick(){  alert(0);  }  </script>  </head>  <body>  <input type=*"button"* value=*"点击"* onclick="myclick()">  <input id=*"b1"* type=*"button"* value=*"点击1"*>  </body>  <script type=*"text/javascript"*>  document.getElementById("b1").onclick = **function**(){  alert(1);  }  </script>  </html> |