1. JavaScript的简介
2. 概述

一门客户端脚本语言，所谓的脚本语言指的是不需要编译的语言，可以直接被浏览器解析执行。

1. 功能

增强用户与html页面的交互过程，控制html元素，让页面有动态的效果。

1. JavaScript与HTML的结合方式
2. 在html内部定义js

在html中定义<script>标签，标签体内容就是js代码。

1. 定义js文件，导入HTML中

定义js文件，在HTML中用<script>标签的src属性导入js文件。

1. JavaScript的数据类型
2. 基本数据类型
3. number

数字类型，分为整数，小数，NaN（非数字）

1. string

字符串类型，用双引号引起来。

1. Boolean

布尔类型，true和false。

1. null

一个对象为空的占位符

1. undefined

如果一个变量没有被初始化，则被默认赋值为undefined。

1. JavaScript变量
2. 强类型语言

强类型语言在开辟变量存储空间时，定义了空间将来存储数据的数据类型，只能存储固定类型的数据。如Java

1. 弱类型语言

在开辟变量存储空间时，不定义空间要存储数据的数据类型，可以存放任意数据类型

1. 变量语法

var 变量名 = 初始化值；

1. 函数对象
2. 创建函数

Function 方法名称(形参){

方法体

}

1. 特点
2. 方法定义的形参类型不用写，返回值类型也不用写
3. 方法是一个对象，如果定义名称相同的方法，前面的方法会被覆盖
4. 在方法声明中有一个内置数组arguments存储所有的实参
5. DOM
6. 概述

Document Object Model，文档对象模型。将标记语言文档的各个组成部分，封装为对象。可以使用这些对象，对标记语言文档进行操作。

（二）DOM三个标准

1. 核心DOM

针对任何结构化文档的标准模型

Document :文档对象

Element:元素对象

Attribute:属性对象

Text:文本对象

Comment:注释对象

Node:节点对象，其他5个对象的父对象

2.XML DOM

针对XML文档的标准模型

1. HTML DOM

针对HTML文档的标准模型

（三）核心DOM模型

1. Document文档对象
2. 创建

在 HTML DOM 模型中可以使用window对象来获取

window.document

document

1. 方法

获取Element对象：

getElementById():根据id属性获取元素对象

getElementsByTagName():根据元素名称获取元素对象们，返回值是数组

getElementsByClassName():根据class属性值获取元素对象们

getElementsByName():根据name属性值获取元素对象们

创建其他DOM对象：

createAttribute(name)

crreateComment()

createElement()

createTextNode()

1. Node节点对象

（1）概述

所有的DOM对象都可以被认为是一个节点。

（2）方法

appendChild()：向节点的子节点列表的结尾添加新的子节点

removeChild():删除（并返回）当前节点的指定子节点

replaceChild()：用新节点替换一个子节点

1. HTML DOM
2. innerHTML

innerHTML用于标签体的设置和获取，直接使用innerHTML就可以修改元素的标签的设置。

如：将div通过innerHTML替换为文本框

<div id=”div”></div>

<script>

var div = document.getELementById(“div”);

var innerHTML = div.innerHTML;

div.innerHTML = “<input type=”text”>”;

</script>

如：通过innerHTML追加文本框

div.innerHTML += “<input type=”text”>”;

（五） 使用核心DOM的动态表格示例

1. 添加表格
2. 为添加按钮绑定点击事件
3. 获取输入文本框的内容
4. 创建表格的td标签，设置td标签的内容为文本框的内容
5. 创建tr标签
6. 将td作为tr的子标签
7. 获取table标签，将tr作为子标签添加到table中
8. 删除表格

通过点击删除超链接删除该行数据，首先获取通过查找该超链接的元素，然后调用删除函数，删除该行。

1. 代码实现

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>动态表格</title>  
 <style>  
 table{  
 border: 1px solid;  
 margin: auto;  
 width: 500px;  
 }  
  
 td, th{  
 text-align: center;  
 border: 1px solid;  
 }  
  
 div{  
 text-align: center;  
 margin: 50px;  
 }  
 </style>  
</head>  
<body>  
 <div>  
 <input type="text" id="id" placeholder="请输入编号">  
 <input type="text" id="name" placeholder="请输入姓名">  
 <input type="text" id="gender" placeholder="请输入性别">  
 <input type="button" value="添加" id="btn\_add">  
 </div>  
  
 <table id="table">  
 <caption>  
 学生信息表  
 </caption>  
 <tr>  
 <th>编号</th>  
 <th>姓名</th>  
 <th>性别</th>  
 <th>操作</th>  
 </tr>  
  
 <tr>  
 <td>1</td>  
 <td>a</td>  
 <td>男</td>  
 <td><a href="javascript:void(0);" onclick="del(this);">删除</a></td>  
 </tr>  
  
 <tr>  
 <td>2</td>  
 <td>b</td>  
 <td>男</td>  
 <td><a href="javascript:void(0);" onclick="del(this);">删除</a></td>  
 </tr>  
  
 <tr>  
 <td>3</td>  
 <td>c</td>  
 <td>女</td>  
 <td><a href="javascript:void(0);" onclick="del(this);">删除</a></td>  
 </tr>  
  
 </table>  
  
 <script>  
 var ***btn\_add*** = ***document***.getElementById("btn\_add");  
 ***btn\_add***.onclick = function () {  
 var id = ***document***.getElementById("id").value;  
 var name = ***document***.getElementById("name").value;  
 var gender = ***document***.getElementById("gender").value;  
  
 //获取表格  
 var table = ***document***.getElementById("table");  
  
 //创建tr  
 var tr = ***document***.createElement("tr");  
  
 //创建id的td  
 var td\_id = ***document***.createElement("td");  
 //创建id文本的文本节点  
 var text\_id = ***document***.createTextNode(id);  
 td\_id.appendChild(text\_id);  
 tr.appendChild(td\_id);  
  
 //创建name的td  
 var td\_name = ***document***.createElement("td");  
 var text\_name = ***document***.createTextNode(name);  
 td\_name.appendChild(text\_name);  
 tr.appendChild(td\_name);  
  
 //创建性别的td  
 var td\_gender = ***document***.createElement("td");  
 var text\_gender = ***document***.createTextNode(gender);  
 td\_gender.appendChild(text\_gender);  
 tr.appendChild(td\_gender);  
  
 //创建超链接a的td  
 var td\_option = ***document***.createElement("td");  
  
 //创建超链接a  
 var a = ***document***.createElement("a");  
 //给超链接添加属性  
 a.setAttribute("href", "javascript:void(0);");  
 a.setAttribute("onclick", "del(this)");  
 var text\_a = ***document***.createTextNode("删除");  
 a.appendChild(text\_a);  
 td\_option.appendChild(a);  
 tr.appendChild(td\_option);  
  
 table.appendChild(tr);  
 }  
  
 //删除函数  
 function del(obj){  
 //通过obj向上找到table,tr  
 var table = obj.parentNode.parentNode.parentNode;  
 var tr = obj.parentNode.parentNode;  
 //通过table删除子节点tr，删除一行  
 table.removeChild(tr);  
 }  
  
 </script>  
</body>  
</html>

（六）使用innerHTML添加动态表格

直接获取table的元素，通过innerHTML直接添加一行。

//通过innerHTML添加  
***document***.getElementById("btn\_add").onclick = function () {  
 var id = ***document***.getElementById("id").value;  
 var name = ***document***.getElementById("name").value;  
 var gender = ***document***.getElementById("gender").value;  
  
 //获取table  
 var table = ***document***.getElementsByTagName("table")[0];  
 //追加一行  
 table.innerHTML += "<tr>\n" +  
 "<td>" + id + "</td>\n" +  
 "<td>" + name + "</td>\n" +  
 "<td>" + gender+ "</td>\n" +  
 "<td><a href=\'javascript:void(0);\' onclick='del(this);\'>删除</a></td>\n" +  
 "</tr>"  
}

1. 事件

（一）概述

事件指的是通过鼠标，键盘等对页面进行操作，当执行了这些操作以后，就能够触发某些代码的执行。

1. 事件：某些操作，如单击，双击，按下键盘，移动鼠标
2. 事件源：组件，如：按钮，文本输入框
3. 监听器：执行的代码
4. 注册监听：将事件，事件源，监听器结合在一起，当事件源发生了某个事件时，则触发执行某个监听器代码。
5. 常见的事件
6. 点击事件

onclick:单击事件

ondbclick:双击事件

1. 焦点事件

onblur:失去焦点

onfocus:元素获得焦点

1. 加载事件

onload：一张页面或衣服图像完成加载

1. 鼠标事件

onmousedown:鼠标被按下

onmouseup:鼠标被松开

onmousemove:鼠标被移动

onmouseout:鼠标从某元素移开

onmouseover:鼠标移到某元素之上

5．键盘事件

onkeydown:某个键盘按键被按下

onkeyup:某个键盘按键被松开

onkeypress:某个键盘按键被按下并松开

6.选中和改变

onchange:域的内容被改变

onselect:文本被选中

7表单事件

onsubmit:确认按钮被点击

onreset:重置按钮被点击

（三）将操作绑定事件

1. 直接在html中，获取元素对象，指定事件属性，属性就是要执行的操作。
2. 通过js获取元素对象，指定事件属性，将操作设置为一个函数，传入该函数执行事件的操作。
3. BOM
4. 概述

Browser Object Model，浏览器对象模型。该模型将浏览器的各个组成部分封装成对象。

1. BOM的组成
2. Navigator

浏览器对象，表示整个浏览器

1. Window

窗口对象，表示每个页面的窗口

1. Screen

显示器屏幕对象

1. History

历史记录对象

1. Location

地址栏对象，记录了地址。

1. Window窗口对象
2. 创建
3. 方法
4. 弹出框方法

Alert()：显示带有一段信息和一个确认按钮的警告框

Confirm()：显示一段信息和确认，取消按钮的对话框。

如果用户选择确定，则返回true

如果用户选择取消，则返回false

Prompt()：显示可提示用户输入的对话框

返回值：获取用户的输入

1. 打开关闭窗口

Open()：打开一个新的浏览器窗口

返回新的Window对象

Close()：关闭调用的浏览器窗口

1. 定时器

setTimeout()：在指定毫秒后调用函数或者表达式

参数：js代码或者函数，毫秒值

返回值：唯一标识，用于取消定时器

clearTimeout()：取消setTimeout()设置的定时器

setInterval()：按照指定的周期调用函数

clearInterval()：取消setInterval()设置的定时器

1. 属性
2. 获取其他BOM对象

history,location,Navigator,Screen对象都可通过Window对象获取，如：

window.history;

1. 获取DOM对象

通过window对象可以获取到document对象

1. 特点
2. window对象不需要创建，可以直接使用，如

window.方法名();

1. window引用可以省略，直接写方法名。
2. 轮播图示例

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
 //显示图片  
 <img src="./img/banner\_1.jpg" alt="兵马俑" id="img">  
  
 <script>  
 var ***num*** = 1;  
 function f() {  
 ***num***++;  
 if(***num*** > 3){  
 ***num*** = 1;  
 }  
 //获取img对象  
 var img = ***document***.getElementById("img");  
 img.src = "./img/banner\_" + ***num*** + ".jpg";  
 }  
  
 //定义定时器  
 setInterval(f, 2000);  
 </script>  
</body>  
</html>

1. Location地址栏对象
2. 获取location对象

window.location

location

1. 方法

reload() 重新加载当前文档

1. 属性

href

设置或返回完整的URL

1. 地址栏示例

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
 <input type="button" id="btn" value="refresh">  
 <input type="button" id="goBaidu" value="baidu">  
 <script>  
 //reload方法  
 var ***btn*** = ***document***.getElementById("btn");  
 ***btn***.onclick = function () {  
 //刷新页面  
 ***location***.reload();  
 }  
  
 //获取地址的href  
 var ***href*** = ***location***.href;  
 var ***goBaidu*** = ***document***.getElementById("goBaidu");  
 ***goBaidu***.onclick = function(){  
 ***location***.href = "https://www.baidu.com";  
 }  
  
 </script>  
</body>  
</html>

1. History历史记录对象
2. 创建

window.history

history

1. 方法

back() 加载history列表中的前一个URL

forward() 加载history列表中的下一个URL

go() 加载history 列表中的某个具体页面

参数：

正数：前进几个历史记录

负数：后退几个历史记录

1. 属性

length 返回当前窗口历史列表中的URL数量

1. history示例

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>History</title>  
</head>  
<body>  
 <input type="button" id="btn" value="history">  
 <a href="./轮播图.html">轮播图</a>  
 <input type="button" id="forward" value="forward">  
 <input type="button" id="back" value="back">  
 <script>  
 var ***btn*** = ***document***.getElementById("btn");  
 ***btn***.onclick = function(){  
 var length = ***history***.length;  
 alert(length);  
 }  
  
 var ***forward*** = ***document***.getElementById("forward");  
 ***forward***.onclick = function () {  
 ***history***.forward();  
 }  
  
 var ***back*** = ***document***.getElementById("back");  
 ***back***.onclick = function(){  
 ***history***.back();  
 }  
  
 </script>  
</body>  
</html>