* JavaScript HTML DOM

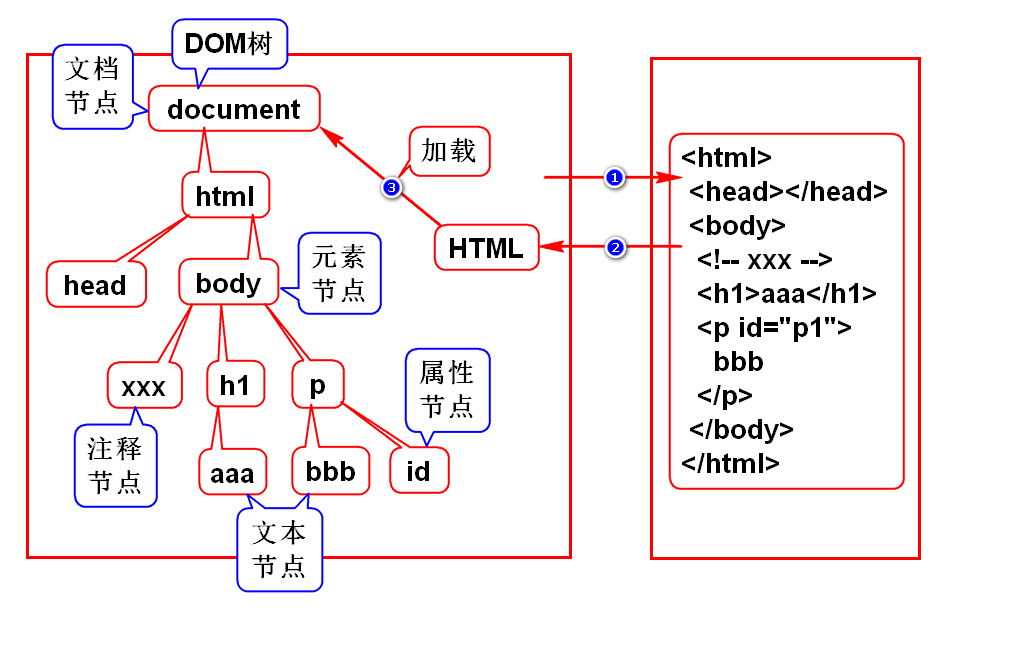
通过 HTML DOM，可访问 JavaScript HTML 文档的所有元素。

HTML DOM （文档对象模型）

当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model）。

HTML DOM 模型被构造为对象的树。

**1.HTML DOM 树**



通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML。

•JavaScript 能够改变页面中的所有 HTML 元素

•JavaScript 能够改变页面中的所有 HTML 属性

•JavaScript 能够改变页面中的所有 CSS 样式

•JavaScript 能够对页面中的所有事件做出反应

**2.查找 HTML 元素**

通常，通过 JavaScript，您需要操作 HTML 元素。

为了做到这件事情，您必须首先找到该元素。有三种方法来做这件事：

•通过 id 找到 HTML 元素

•通过标签名找到 HTML 元素

•通过类名找到 HTML 元素

1)在 DOM 中查找 HTML 元素的最简单的方法，是通过使用元素的 id。

实例本例查找 id="intro" 元素：

var x=document.getElementById("intro");

如果找到该元素，则该方法将以对象（在 x 中）的形式返回该元素。

如果未找到该元素，则 x 将包含 null。

2)通过标签名查找 HTML 元素

本例查找 id="main" 的元素，然后查找 "main" 中的所有 <p> 元素：

var x=document.getElementById("main");

var y=x.getElementsByTagName("p");

提示：通过类名查找 HTML 元素在 IE 5,6,7,8 中无效。

* JavaScript HTML DOM - 改变 HTML

HTML DOM 允许 JavaScript 改变 HTML 元素的内容。

**1.改变 HTML 输出流**

JavaScript 能够创建动态的 HTML 内容：

今天的日期是： Wed May 10 2017 20:09:28 GMT+0800 (中国标准时间)

在 JavaScript 中，document.write() 可用于直接向 HTML 输出流写内容。

<script>

document.write(Date());

</script>

提示：绝不要使用在文档加载之后使用 document.write()。这会覆盖该文档。

**2.改变 HTML 内容**

修改 HTML 内容的最简单的方法时使用 innerHTML 属性。

如需改变 HTML 元素的内容，请使用这个语法：

document.getElementById(id).innerHTML=new HTML

本例改变了 <h1> 元素的内容：

<h1 id="header">Old Header</h1>

<script>

var element=document.getElementById("header");

element.innerHTML="New Header";

</script>

**3.改变 HTML 属性**

如需改变 HTML 元素的属性，请使用这个语法：

document.getElementById(id).attribute=new value

本例改变了 <img> 元素的 src 属性：

<img id="image" src="smiley.gif">

<script>

document.getElementById("image").src="landscape.jpg";

</script>

* JavaScript HTML DOM - 改变 CSS

HTML DOM 允许 JavaScript 改变 HTML 元素的样式。

改变 HTML 样式

如需改变 HTML 元素的样式，请使用这个语法：

document.getElementById(id).style.property=new style

本例改变了 id="id1" 的 HTML 元素的样式，当用户点击按钮时：

<h1 id="id1">My Heading 1</h1>

<button type="button" onclick="document.getElementById('id1').style.color='red'">点击这里</button>

Visibility使元素不可见

* JavaScript HTML DOM 事件

HTML DOM 使 JavaScript 有能力对 HTML 事件做出反应。

我们可以在事件发生时执行 JavaScript，比如当用户在 HTML 元素上点击时。

如需在用户点击某个元素时执行代码，请向一个 HTML 事件属性添加 JavaScript 代码：

onclick=JavaScript

**1.HTML 事件的例子：**

•当用户点击鼠标时

•当网页已加载时

•当图像已加载时

•当鼠标移动到元素上时

•当输入字段被改变时

•当提交 HTML 表单时

•当用户触发按键时

在本例中，当用户在 <h1> 元素上点击时，会改变其内容：

<h1 onclick="this.innerHTML='谢谢!'">请点击该文本</h1>

本例从事件处理器调用一个函数：

<script>

function changetext(id){id.innerHTML="谢谢!";}

</script>

<body>

<h1 onclick="changetext(this)">请点击该文本</h1>

</body>

**2.HTML 事件属性**

如需向 HTML 元素分配 事件，您可以使用事件属性。

实例

向 button 元素分配 onclick 事件：

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

使用 HTML DOM 来分配事件

HTML DOM 允许您通过使用 JavaScript 来向 HTML 元素分配事件：

向 button 元素分配 onclick 事件：

<script>

document.getElementById("myBtn").onclick=function(){displayDate()};

</script>

在上面的例子中，名为 displayDate 的函数被分配给 id=myButn" 的 HTML 元素。

当按钮被点击时，会执行该函数。

**3.onload 和 onunload 事件**

onload 和 onunload 事件会在用户进入或离开页面时被触发。

onload 事件可用于检测访问者的浏览器类型和浏览器版本，并基于这些信息来加载网页的正确版本。

onload 和 onunload 事件可用于处理 cookie。

实例

<body onload="checkCookies()">

**4.onchange 事件**

onchange 事件常结合对输入字段的验证来使用。

下面是一个如何使用 onchange 的例子。当用户改变输入字段的内容时，会调用 upperCase() 函数。

<input type="text" id="fname" onchange="upperCase()">

**5.onmouseover 和 onmouseout 事件**

onmouseover 和 onmouseout 事件可用于在用户的鼠标移至 HTML 元素上方或移出元素时触发函数。

一个简单的 onmouseover-onmouseout 实例：

把鼠标移到上面

onmousedown、onmouseup 以及 onclick 事件

onmousedown, onmouseup 以及 onclick 构成了鼠标点击事件的所有部分。首先当点击鼠标按钮时，会触发 onmousedown 事件，当释放鼠标按钮时，会触发 onmouseup 事件，最后，当完成鼠标点击时，会触发 onclick 事件。

* JavaScript HTML DOM 元素（节点）

添加和删除节点（HTML 元素）。

**1.创建新的 HTML 元素**

如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

<div id="div1">

<p id="p1">这是一个段落</p>

<p id="p2">这是另一个段落</p>

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("这是新段落。");

para.appendChild(node);

var element=document.getElementById("div1");

element.appendChild(para);

</script>

例子解释：

这段代码创建新的 <p> 元素：

var para=document.createElement("p");

如需向 <p> 元素添加文本，您必须首先创建文本节点。这段代码创建了一个文本节点：

var node=document.createTextNode("这是新段落。");

然后您必须向 <p> 元素追加这个文本节点：

para.appendChild(node);

最后您必须向一个已有的元素追加这个新元素。

这段代码找到一个已有的元素：

var element=document.getElementById("div1");

这段代码向这个已有的元素追加新元素：

element.appendChild(para);

**2.删除已有的 HTML 元素**

如需删除 HTML 元素，您必须首先获得该元素的父元素：

实例

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另一个段落。</p>

</div>

<script>

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.removeChild(child);

</script>

例子解释：

这个 HTML 文档含有拥有两个子节点（两个 <p> 元素）的 <div> 元素：

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另一个段落。</p>

</div>

找到 id="div1" 的元素：

var parent=document.getElementById("div1");

找到 id="p1" 的 <p> 元素：

var child=document.getElementById("p1");

从父元素中删除子元素：

parent.removeChild(child);

提示：如果能够在不引用父元素的情况下删除某个元素，就太好了。

不过很遗憾。DOM 需要清楚您需要删除的元素，以及它的父元素。

这是常用的解决方案：找到您希望删除的子元素，然后使用其 parentNode 属性来找到父元素：

var child=document.getElementById("p1");

child.parentNode.removeChild(child);

* **示例1:**

DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script>

window.onload = function() {

//查询节点

//1.根据ID查询节点

//2.根据标签名查询节点

//3.根据NAME查询节点

var radios =

document.getElementsByName("sex");

console.log(radios);

//4.根据层次(关系)查询节点

var gz = document.getElementById("gz");

//1)查询某节点的父亲

var ul = gz.parentNode;

console.log(ul);

//2)查询某节点的孩子(带文本)

var lis = ul.childNodes;

console.log(lis);

//3)查询某节点的孩子(不带文本)

lis = ul.getElementsByTagName("li");

console.log(lis);

//4)节点.父亲.孩子们[i]

var sh =

gz.parentNode.getElementsByTagName("li")[1];

console.log(sh);

}

function f1() {

//创建新节点

var li = document.createElement("li");

//修改此li

li.innerHTML = "天津";

//追加新节点

var ul = document.getElementById("city");

ul.appendChild(li);

}

function f2() {

//创建节点

var li = document.createElement("li");

//修改节点

li.innerHTML = "石家庄";

//插入节点

var ul = document.getElementById("city");

var gz = document.getElementById("gz");

ul.insertBefore(li,gz);

}

function f3() {

//通过父亲删除孩子

var ul = document.getElementById("city");

var gz = document.getElementById("gz");

ul.removeChild(gz);

}

</script>

</head>

<body>

<p>

<input type="radio" name="sex"/>男

<input type="radio" name="sex"/>女

</p>

<ul id="city">

<li>北京</li>

<li>上海</li>

<li id="gz">广州</li>

<li>深圳</li>

<li>杭州</li>

</ul>

<p>

<input type="button" value="增加"

onclick="f1();"/>

<input type="button" value="插入"

onclick="f2();"/>

<input type="button" value="删除"

onclick="f3();"/>

</p>

</body>

</html>

* **示例2:省市联动**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script>

var cities;

window.onload = function(){

//查询所有的城市(模拟)

cities = [

["石家庄","廊坊","保定"],

["济南","青岛","烟台"],

["南京","苏州","无锡"]

];

}

function chg() {

console.log(1);

//获取省份(下标)

var p =

document.getElementById("province").value;

console.log(p);

//获取本省份的城市

var pcities = cities[p];

//先删除旧的城市

var select =

document.getElementById("city");

//var options =

// select.getElementsByTagName("option");

//for(var i=options.length-1;i>0;i--) {

// select.removeChild(options[i]);

//}

select.innerHTML = "<option>请选择</option>";

//再追加新的城市

if(pcities) {

for(var i=0;i<pcities.length;i++) {

var option =

document.createElement("option");

option.innerHTML = pcities[i];

select.appendChild(option);

}

}

}

</script>

</head>

<body>

省:

<select id="province"

onchange="chg();">

<option value="-1">请选择</option>

<option value="0">河北省</option>

<option value="1">山东省</option>

<option value="2">江苏省</option>

</select>

市:

<select id="city">

<option>请选择</option>

</select>

</body>

</html>

* **示例3:创建对象演示**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script>

//采用直接量方式创建对象

function f1() {

var stu = {

"name":"zhangsan",

"age":25,

"work":function(){

alert("我学Java");

}

};

alert(stu.name);

alert(stu.age);

stu.work();

}

//采用内置构造器创建对象

function f2() {

var tch = new Object();

tch.name = "cang";

tch.age = 20;

tch.work = function(){

alert("我教Java");

}

alert(tch.name);

alert(tch.age);

tch.work();

}

//采用自定义构造器创建对象

function Coder(name,age,work) {

this.name = name;

this.age = age;

//this指代当前对象

//this.job给当前对象加job属性

//=work是将参数work赋值给此属性

this.job = work;

}

function f3() {

var coder = new Coder(

"wangwu",

28,

function(){

alert("我写Java");

}

);

alert(coder.name);

alert(coder.age);

coder.job();

}

</script>

</head>

<body>

<input type="button" value="按钮1"

onclick="f1();"/>

<input type="button" value="按钮2"

onclick="f2();"/>

<input type="button" value="按钮3"

onclick="f3();"/>

</body>

</html>

* **示例4:冒泡机制**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<style>

div {

width: 200px;

border: 1px solid red;

padding: 30px;

}

p {

border: 1px solid red;

text-align: center;

padding: 30px;

}

</style>

<script>

function f1(e) {

alert("button");

//取消冒泡

//IE:e={cancelBubble:false}

//其他:e={stopPropagation:function(){}}

if(e.stopPropagation) {

e.stopPropagation();

} else {

e.cancelBubble = true;

}

}

</script>

</head>

<body>

<div onclick="alert('div');">

<p onclick="alert('p');">

<input type="button" value="按钮1"

onclick="f1(event);"/>

</p>

</div>

</body>

</html>

* **示例5:计算器**

<!DOCTYPE html>

<html>

<head>

<title>计算器</title>

<meta charset="utf-8" />

<style type="text/css">

.panel {

border: 4px solid #ddd;

width: 192px;

margin: 100px auto;

/\*border-radius: 6px;\*/

}

.panel p, .panel input {

font-family: "微软雅黑";

font-size: 20px;

margin: 4px;

float: left;

/\*border-radius: 4px;\*/

}

.panel p {

width: 122px;

height: 26px;

border: 1px solid #ddd;

padding: 6px;

overflow: hidden;

}

.panel input {

width: 40px;

height: 40px;

border:1px solid #ddd;

}

</style>

<script>

window.onload = function() {

var div =

document.getElementById("jsq");

//给div动态绑定单击事件

div.onclick=function(e){

//获取事件源(INPUT/P/DIV)

var obj = e.srcElement || e.target;

//只处理按钮INPUT

if(obj.nodeName=="INPUT") {

var p =

document.getElementById("screen");

if(obj.value=="=") {

try {

p.innerHTML = eval(p.innerHTML);

} catch(ex) {

p.innerHTML = "Error";

}

} else if(obj.value=="C") {

p.innerHTML = "";

} else {

p.innerHTML += obj.value;

}

}

}

}

</script>

</head>

<body>

<div class="panel" id="jsq">

<div>

<p id="screen"></p>

<input type="button" value="C">

<div style="clear:both"></div>

</div>

<div>

<input type="button" value="7">

<input type="button" value="8">

<input type="button" value="9">

<input type="button" value="/">

<input type="button" value="4">

<input type="button" value="5">

<input type="button" value="6">

<input type="button" value="\*">

<input type="button" value="1">

<input type="button" value="2">

<input type="button" value="3">

<input type="button" value="-">

<input type="button" value="0">

<input type="button" value=".">

<input type="button" value="=">

<input type="button" value="+">

<div style="clear:both"></div>

</div>

</div>

</body>

</html>