如何获取该变量(对象)是不是属于什么类型的?

变量 Instanceof 类型的名字 布尔类型 true就是这种类型 false就不是这种类型

alert 弹框

prompt 弹框 可以输入内容

confirm 弹框 返回true 或 false

页面关闭后触发的事件 window.onunload

页面关闭前触发的事件window.onbeforeload

console.air(“”) 打印对象

鼠标点击事件函数

onmouseover 鼠标进入

onmouseout 鼠标离开

getElementsByName 获取name属性的名字 伪数组

getElementsByTagName 获取标签元素 伪数组

getElementsByClassName 获取类属性 伪数组

, querySelector 根据选择器的方式获取单个元素 返回一个元素

querySelector 根据选择器的方式获取多个元素 返回多个元素

onfocus 注册获取焦点事件

onblur 注册失去焦点事件

textContent 设置标签中的文本内容

innerText 设置标签中的文本内容

如果浏览器不支持这个属性,那么这个属性的类型就是undefined

兼容代码

设置任意的标签中间的任意文本内容

Function setInnerText(element, text) {

//判断浏览器是否支持这个属性

If(typeof element.textContent == “undefined”) { //不支持

Element.innerText=text;

} else {

Element.textContent = text;

}

}

获取任意标签的文本内容

Function getInnerText(element) {

If (typeof element.textContent === “undefined”) {

return element.innerText;

} else {

return element.textContent;

}

}

innerText主要是设置文本的,设置标签内容,是没有标签效果的

innerHTML 可以设置文本内容, 主要作用是在标签中设置新的html标签内容,室友标签效果的,

想要设置标签内容,使用innerText或者textContent,或者innerHTML, 推荐使用innerHtml

innerText可以获取标签中的文本内容,但是标签中如果还有标签,那么最里面的标签的文本内容也能获取 ---- 获取不到标签, 文本可以获取

innerHTML 才是真正的获取标签中间的所有内容的

getAttribute(“自定义属性的名字”)

在HTML标签中添加的自定义属性,如果想要获取这个属性的值,需要使用 getAttribute(“自定义属性的名字”) 才能获取这个属性的值

设置自定义属性: setAttribute (“属性的名字”, “属性的值”);

获取自定义属性的值: getAttribute(“属性的名字”);

移除自定义属性 removeAttribute(“属性名”)

阻止超链接默认的跳转事件 return false

标签的父级节点

parentNode

标签的父级元素

parentElement

nodeType 节点类型

如果是1 是标签

2.属性

3.文本

nodeName 节点名字

大写的标签 ---标签

小写的属性名---属性

#text---------文本

nodeValue 节点数值

标签----null

属性---属性的值

文本---文本的内容

childNodes 子节点

children 子元素

firstChild 第一个子节点

firstElementChild 第一个子元素

lastChild 最后一个子节点

lastElementChild 最后一个子元素

perviousSibling 前一个兄弟的节点

previousElementSibling 前一个兄弟的元素

nextSibling 后一个兄弟节点

nextElementSibling 后一个兄弟元素

getAttributeNode(“属性的名字”) 返回的是一个节点

//element.firstChild -🡪 谷歌和火狐获取的是第一个子节点

//element.firstChild🡪 IE8获取的是第一个子元素

//element.firstElementChild --- 谷歌和火狐是第一个子元素,IE8不支持

// 获取任意一个父级元素的第一个子集元素

function getFirstElementChild(element) {

if (element.firstElementChild) {

return element.firstElementChild;

} else {

var node = element.firstChild; //第一个节点

while (node && node.nodeType !=1) {

node = node.nextSibling;

}

return node;

}

}

// 获取任意一个父级元素的最后一个子集元素

function getFirstElementChild(element) {

if (element.firstElementChild) {

return element.firstElementChild;

} else {

var node = element.firstChild; //第一个节点

while (node && node.nodeType !=1) {

node = node.previousSibling;

}

return node;

}

}

document.body.style.backgroundImage = “url’+this.src+’”; 背景颜色拼接

元素创建的三种方式

1. document.write(标签的代码及内容”)
2. 对象.innerHTML = “标签及代码”;
3. document.createElement(“标签的名字”);

document.write()创建元素缺陷: 如果是在页面加载完毕后,此时通过这种方式创建元素,那么页面上存在的所有内容都会被干掉

页面加载时创建的,其他元素都有

var 变量 = document.createElement(“元素名”) 创建元素

setInnerText(变量, “内容”); 给元素赋值

appendChild(变量名) 把创建后的子元素追加到父元素中

元素.insertBefore(新元素, 要操作的元素)

document.getElement.insertBefore(obj, dv.firstElementChild) 把新元素插入到第一个子元素的前面

replaceChild() 替换新元素

removeChild() 移除元素

dv.removeChild(dv.firstElementChild) 移除第一个子元素

addEventListener (事件 函数 布尔) 为同一个元素绑定多个相同的事件

参数1: 事件的类型---事件的名字,没有on

参数2: 事件处理函数—函数(命名函数,匿名函数)

参数3: 布尔类型,目前就写false---

my$(“id”).addEventListener(“click”, function() {}, false )

为任意元素,绑定任意的事件,任意的元素,事件的类型,事件处理函

function addEventListener(element, type, fn) {

//判断浏览器是否支持这个方法

if (element.addEventListener) {

element.addEventListener(type, fn, false);

} else if (element.attachEvent) {

element.attachEvent(“on”+type, fn);

} else {

element[“on”+type]=fn;

}

}

总结绑定事件的区别:

addEventListener();

attachEvent();

相同点: 都可以为元素绑定方法

不同点:

1. 方法名不一样
2. 参数的个数不一样addEventListener三个参数, attachEvent 两个参数
3. addEventListener 谷歌,火狐,IE11支持, IE8不支持

attachEvent谷歌火狐不支持,IE11不支持,IE8支持

1. this不同, addEventListener 中的this是当前绑定事件的对象

attachEvent 中的this是window

1. addEventListener 中的事件的类型(事件名字)没有on

attachEvent 中的事件的类型(事件的名字)有on

解绑事件:

注意: 用什么方式绑定事件,就应该用对应的方式解绑事件

1. 解绑事件

对象.on事件名字=事件处理函数-🡪绑定事件

对象.on事件名字=null;

1. 解绑事件

对象.addEventListener(“没有on 的事件类型”, 命名函数, false); ---绑定事件

对象.removeEventListener(“没有on的事件类型”, 函数名字, false)

1. 解绑事件

对象.attachEvent(“on事件类型”, 命名函数); ---绑定事件

对象.detachEvent(“on事件类型”, 函数名字);

1. 点击第二个按钮,把第一个按钮的第一个点击事件解绑

解绑事件的时候,需要在绑定事件的时候,使用命名函数

removeEventListener(点击事件, 函数, 布尔);

阻止事件冒泡排序

window.event.cancelBubble=true; ie特有 谷歌支持, 火狐不支持

e.stopPropagation(); 谷歌和火狐支持 ie8不支持

获取文本框注册键盘抬起事件

onkeyup

onkeydown

e.keyCode 键盘码

e.preventDefault 取消键盘默认行为

**location对象的属性和方法**

**地址栏上#及后面的内容**

**window.location.hash**

**主机名及端口号**

**window.location.host**

**主机名**

**window.location.hostname**

**文件的路径—相对路径**

**window.location.pathname**

**端口号**

**window.location.port**

**协议**

**window.location.protocol**

**搜索的内容**

**window.location.search**

**onload=function() {**

**设置跳转的网页地址**

**location.href=”**[**http://www.jd.com**](http://www.jd.com)**” 属性 -----重点**

**location.assign(“http://www.id.com”) 方法**

**location.reload(); 重新加载—刷新**

**location.replace(“http://www.jd.com”) ; 没有历史记录 不能反回**

**}**

**window.history.forward() 前进**

**window.history.back() 后退**

**window.navigator.userAgent 通过userAgent可以判断用户浏览器的类型**

**window.navrAgent.platform 通过platform可以判断浏览器所在的系统平台类型**

**定时器**

**参数1 : 函数**

**参数2 : 时间—毫秒—1000毫秒—1秒**

**//执行过程: 页面加载完毕后,过了1秒,执行函数代码,又过了一秒,执行函数代码…..**

**返回值就是定时器的id值**

**var timeId = setInterval(function() {**

**alert(“hello”);**

**}, 1000);**

**document.getElementById(“btn”).onclick = function() {**

**//点击按钮,停止计时器**

**//参数: 要清理的定时的id的值**

**window.clearInterval(timeId);**

**}**

**onload 页面加载完成之后触发**

**setTimeout 一次性计时器, 延时定时器**

**clearTimeout 取消定时器**

**opacity 透明度 css3属性**

**offsetleft 获取left当前属性的位置**

**cloneNode 克隆**

**ulObj.appendChild(ulObj.children[0].cloneNode(true));**

**onmousemove 鼠标移动**

**可视区域的横坐标clientX**

**可视区域的纵坐标 clientY**

**e = e || window.event 处理时间对象浏览器的兼容性**

**e.enentPhase 事件阶段 1: 捕获阶段 2 目标阶段 3.冒泡阶段**

**target = e.target ||**

**e.target 获取真正触发事件的对象**

**e.currentTarget 事件处理函数所属的对象 就是this**

**重点属性**

**e.pageX 鼠标在页面的x轴坐标**

**epageY 鼠标在页面的Y轴坐标**

**e.clientX 鼠标在可视区的X轴坐标**

**e.clientY 鼠标在可视区的Y轴坐标**

**兼容性处理**

**有的浏览器支持body 有的支持html**

**document.body.scrollTop**

**document.documentElement.scorllTop**

**offsetParent: 距离自身最近的,并且带有定位的父元素**

**offsetWidth 元素的宽 有边框**

**offsetHeight 元素的高 有边框**

**scrollWidth: 元素中内容的实际的宽 没有边框 如果没有内容,就是元素的宽**

**scrollHeight: 元素中内容的实际的高 没有边框 如果没有内容,就是元素的宽**

**scrollLeft 向左卷曲的距离**

**scrollTop 向上卷曲的距离**

**内容的宽高,元素自身的宽高,并且溢出的部分的内容宽高,**

**clientLeft 获取左边框的宽度 + 左滚动条的距离**

**clientTop 获取上边框的宽度 + 顶部滚动条的距离**

**clentWidth / clientHeight : 元素的宽高, 元素自身的宽高, + padding 不包括border**

**window.pageYoffset 向上卷曲出去的距离**

**window.getComputedStyle (my$(dv), null).left 获取任意类型属性的值 (谷歌支持)**

**currentStyle.left 获取任意类型属性的值 (IE8支持)**

**dirction**

**onmouseleave 鼠标离开,不会触发冒泡**

**onmoseenter 鼠标进入**

**设置鼠标移动的时候,文字不被选择**

**window.getSelection ? window.getSelection().removeAllRanges() : document.selection.empty();**

**scrollTo(x, y) 绝对滚动,跳转到目标位置**

**scrollBy(x, y) 相对滚动,相对于当前位置跳转一个步进值**

**onmousewheel 鼠标滚轮事件**