# WebApi复习资料

JavaScript分三个部分:

1. ECMAScript标准----JS基本的语法

2. DOM:Document Object Model 文档对象模型

3. BOM:浏览器对象模型

DOM的作用:操作页面的元素

DOM树:把html页面或者是xml文件看成是一个文档,文档就是一个对象,这个文档中所有的标签都是元素,元素也可以看成是对象,标签(元素,对象)有很多,还有嵌套的关系,组成的这种层次结构,可以模拟成树形结构图,简称:树状图 ,就是DOM树

获取元素:

2种方式:

根据id从整个的文档中获取元素---返回的是一个元素对象

document.getElementById("id属性的值")

根据标签的名字获取元素-----返回的是元素对象组成的伪数组

document.getElementsByTagName("标签的名字");

操作基本标签的属性

src,title,alt,href,id属性

操作表单标签的属性

name,value,type,checked,selected,disabled,readonly

元素的样式操作

对象.style.属性=值;

对象.className=值;

为元素添加事件的操作

事件:就是一件事,有事件源,触发和响应

this关键字----如果是在当前的元素的事件中使用this,那么this就是当前的对象

内置对象:系统自带的对象

自定义对象:自己写的对象

浏览器对象:

DOM

DOM对象:------->通过DOM方式获取的元素得到的对象

元素:element:页面中的标签

节点:Node:页面中所有的内容,标签,属性,文本

根元素:html标签

页面中的顶级对象---:document

阻止超链接默认的跳转事件: return false;

获取元素的方式

根据id获取元素

document.getElementById("id属性的值");

根据标签名字获取元素

document.getElementsByTagName("标签的名字");

有的浏览器不支持, IE低版本的浏览器不支持,IE8以下的

根据name属性的值获取元素

document.getElementsByName("name属性的值");

根据类样式的名字获取元素

document.getElementsByClassName("类样式的名字");

根据选择器获取元素

document.querySelector("选择器");返回一个对象

根绝选择器获取元素

document.querySelectorAll("选择器");返回数组,多个元素组成的

设置元素的样式的方式

对象.style.属性=值;

对象.className=值;

innerText和textContent的兼容问题

目前的浏览器都支持innerText,应该是属于ie的标准

textContent本身是火狐支持,IE8不支持

innerText和innerHTML的区别

都可以设置标签的文本内容,如果要设置标签及内容推荐使用innerHTML

如果要获取标签中的文本,innerText,也可以使用innerHTML

如果想要获取的是有标签,也有文本---innerHTML

自定义属性的操作

自定义属性:标签原本没有这个属性,为了存储数据,程序员自己添加的属性

自定义属性无法直接通过DOM对象的方式获取或者设置

对象.getAttribute("自定义属性名字");获取自定义属性的值

对象.setAttribute("属性名字","值");设置自定义属性及值

移除自定义属性

对象.removeAttribute("属性的名字");

节点:

节点:页面中所有的内容都是节点(标签,属性,文本:文字,空格,换行)

文档：document---页面中的顶级对象

元素:页面中所有的标签, 标签---元素--对象(通过DOM的方式来获取这个标签,得到了这个对象,此时这个对象叫DOM对象)

节点的属性:作用:为了将来获取很多节点,得到节点中的标签(元素),识别节点中的标签 元素

节点的类型:1标签节点,2属性节点,3文本节点

nodeType:节点的类型,1---标签节点,2----属性节点,3---文本节点

nodeName:标签节点--大写的标签名字,属性节点---小写的属性名字,文本节点---#text

nodeValue:标签---null,属性---属性的值,文本---文本内容

if(node.nodeType==1&&node.nodeName=="P")

获取节点及元素的代码

//获取当前节点的父级节点

console.log(my$("uu").parentNode);

//获取当前节点的父级元素

console.log(my$("uu").parentElement);

//获取当前节点的子级节点

console.log(my$("uu").childNodes);

//获取当前节点的子级元素

console.log(my$("uu").children);

//获取当前节点的第一个子级节点

console.log(my$("uu").firstChild);

//获取当前节点的第一个子级元素

console.log(my$("uu").firstElementChild);

//获取当前节点的最后一个子级节点

console.log(my$("uu").lastChild);

//获取当前节点的最后一个子级元素

console.log(my$("uu").lastElementChild);

//获取当前节点的前一个兄弟节点

console.log(my$("uu").previousSibling);

//获取当前节点的前一个兄弟元素

console.log(my$("uu").previousElementSibling);

//获取当前节点的后一个兄弟节点

console.log(my$("uu").nextSibling);

//获取当前节点的后一个兄弟元素

console.log(my$("uu").nextElementSibling);

元素的创建

三种元素创建的方式

1. document.write("标签代码及内容"); 如果在页面加载完毕后创建元素.页面中的内容 会被干掉

2. 父级对象.innerHTML="标签代码及内容";

3. document.createElement("标签名字");得到的是一个对象,

父级元素.appendChild(子级元素对象);

父级元素.inerstBefore(新的子级对象,参照的子级对象);

移除子元素

父级元素.removeChild(要干掉的子级元素对象);

事件的绑定:为同一个元素绑定多个相同的事件

三种方式:

1. 对象.on事件名字=事件处理函数 如果是多个相同事件注册用这种方式,最后一 个执行,之前的被覆盖了

my$("btn").onclick=function(){};

2. 对象.addEventListener("没有on的事件名字",事件处理函数,false);

my$("btn").addEventListener("click",function(){},false);

3. 对象.attachEvent("有on的事件名字",事件处理函数);

my$("btn").attachEvent("onclick",function(){});

为元素绑定事件的三种方式:

1.

对象.on事件类型=事件处理函数;

2.IE8不支持

对象.addEventListener("没有on的事件类型",事件处理函数,false);

3.谷歌和火狐不支持

对象.attachEvent("有on的事件类型",事件处理函数);

为元素解绑事件的三种方式:

1.

对象.on事件类型=null;

2.

对象.removeEventListener("没有on的事件类型",事件处理函数名字,false);

3.

对象.detachEvent("有on的事件类型",事件处理函数);

事件冒泡:元素A中有元素B,都有相同的事件,里面的元素的事件触发了,外面元素的事件 也会触发.可以是多个元素嵌套

阻止事件冒泡:

两种

1.

e.stopPropagation(); e---事件参数对象,谷歌和火狐都支持,IE8不支持

IE8和谷歌支持

window.event.cancelBubble=true;

window.event就是事件参数对象----e是一样的

BOM:Browser Object Model 浏览器对象模型---操作浏览器

历史记录的后退和前进 history: back()后退 forward()前进

地址栏上的地址的操作 location href属性跳转页面, assgin()跳转页面 reload()刷新 replace()方法替换地址栏上的地址的,没有历史记录的

获取系统和浏览器的信息的 navigator userAgent属性---获取系统,浏览器的信息的

地址栏上#及后面的内容

console.log(window.location.hash);

主机名及端口号

console.log(window.location.host);

主机名

console.log(window.location.hostname);

文件的路径---相对路径

console.log(window.location.pathname);

端口号

console.log(window.location.port);

协议

console.log(window.location.protocol);

搜索的内容

console.log(window.location.search);

定时器:

该定时器会返回自己的id值

var timeId=window.setInterval(函数,时间);

执行过程:页面加载完毕后,过了一段时间执行函数,反复的,除非清理定时器

window.clearInterval(定时器的id值);清理定时器

轮播图自己的说的要一遍的，写思路

定时器:两个

setInterval(函数,时间);//定时器,隔一段时间就执行一次函数的代码

clearInterval(定时器的id);

setTimeout(函数,时间);//定时器,只执行一次函数的代码-----一次性的定时器

clearTimeout(定时器的id);//

动画函数:

offset系列:获取元素的相关的样式属性的值

offsetWidth:获取元素的宽

offsetHeight:获取元素的高

offsetLeft:获取元素距离左边位置的值

offsetTop:获取元素距离上面位置的值

完整轮播图:

1. 根据ul中li个数(图片的个数)动态的创建li,并且加入到ol中,作为小按钮来使用

为每个ol中的li添加鼠标进入的事件,(添加事件之前,先把索引保存在每个li的自定义属 性中),鼠标进入li的时候做排他功能(背景颜色),移动图片(图片整体移动,移动的ul,每次 移动的是一个图片的宽度,图片的宽度就是向宽的宽度),调用的是动画函数,移动ul,移动 的是索引值图片的宽度,并且把索引值存储在一个全局变量中,pic

2. 创建li之后,1---设置ol中第一个li有默认的背景颜色,2---把ul中的第一个li克隆一下 (无缝连接的效果)追加到ul中所有的li的最后面(无缝效果:页面中的第一个图片和最后 一个图片是同一个图片)

3.左右焦点的div显示和隐藏

4.为左右按钮注册点击事件

每一次点击右边的按钮,先判断pic的值是不是ul中li的length-1(pic的值最大是5),此时 设置pic的值为0,同时设置ul距离左边的left值为0(初始值),此时pic立刻加1,移动ul, 用户看到的就是第二个图片

设置小按钮的背景颜色

左边按钮,需要判断pic值是不是0,如果是0此时pic=5,ul的left为5图片的宽,立刻pic-1, 然后ul移动一张图片,此时用户看到了第五个图片

5.自动播放在页面加载的时候,启动一个定时器,里面调用的就是右边按钮的点击事件的 处理函数

BOM中顶级对象:Window

DOM中顶级对象:Document

jQuery中顶级对象:$--jQuery

三大系列:

offset系列:

(父级元素margin+父级元素padding+父级元素border+自己的margin)

offsetLeft:元素距离左边位置的值

offsetTop:元素距离上面位置的值

offsetWidth:获取元素的宽度(有边框)

offsetHeight:获取元素的高度(有边框)

scroll系列:卷曲

scrollLeft:元素向左卷曲出去的距离

scrollTop:元素向上卷曲出去的距离

scrollWidth:元素中内容的实际的宽度,如果没有内容,或者内容很少,元素的宽度

scrollHeight:元素中内容的实际的高度,如果没有内容,或者内容很少,元素的高度

client系列:

clientWidth:可视区域的宽度,没有边框

clientHeight:可视区域的高度,没有边框

clientLeft:左边框的宽度

clientTop:上边框的宽度

clientX:可视区域的横坐标

clientY:可视区域的纵坐标

动画函数:-----------------在于会调用

匀速动画:重点内容----

变速动画(缓动动画)---第一个版本写出来---jQuery中就有动画函数,直接使用

如何缓动的?

(目标-当前)/10

增加的回调函数,增加了透明度和层级