**事件（蓝色字体都是兼容问题）**

**1.概念：**事件是可以被JavaScript侦测到的行为。网页中的每个元素都可以产生某些可以触发JavaScript函数的事件

**2.事件分类：**

**鼠标事件**

**键盘事件**

**UI事件**

**表单事件**

**鼠标事件有：**

onclick ，ondblclick，onmouseover，onmouseout，onmousemove

Onmousedown，onmouseup，（）onmouseenter，onmouseleave这个事件不冒泡）

oncontextmenu鼠标右键菜单展开时。

鼠标滚轮事件：

onmousewheel ：IE和谷歌浏览器支持，火狐不支持。该事件可以通过传统的方式绑定，也可通过事件监听的方式绑定事件处理程序。

DOMMouseScroll：火狐所能支持的，它必须通过事件监听的方式绑定事件处理程序。

**处理鼠标滚轮事件兼容问题：两个都写；因为互不报错**

**键盘事件有**：onkeydown 某个键盘按键被按下。 //IE8及以下版本不会重复触发！

Onkeyup(兼容oninput)某个键盘按键被松开。

onkeypress 键盘<字符键>被按下,而且如果按住不放的话，会重复触发此事件。 //IE8及以下版本不支持浏览器版本 操作系统

**UI事件有：**

window.onload 页面元素（包括图片多媒体等）加载完成后

window.onbeforeunload 用户退出页面前所产生的事件。（有兼容性问题）

window.onresize 窗口或框架被重新调整大小。（有兼容性问题）

onselect  文本框文本被选中。（有兼容性问题）

**表单事件：**

作用于<form >，input系列的标签，包括文本域 textarea，下拉框 select等。

onblur 元素失去焦点时触发。

onfocus 元素获得焦点时触发。

onchange 元素内容被改变时触发。

onreset 重置按钮被点击。(注册在form节点对象上)

onsubmit 确认按钮被点击。（注册在form节点对象上）

oninput 输入字符时触发（IE8及以下版本有兼容性题）可以用我们的 onkeyup 事件来做一个兼容处理

**事件传播：事件冒泡**

一般是不要事件冒泡的,所以才有阻止冒泡的方法

但是有兼容问题。

If(e.stopPropagation){//e事件对象

e.stopPropagation();

}else{

e.cancelBubble = true;

}

**事件对象**

**1.概念：**

事件执行过程中的状态，用来保存当前事件的信息对象，

就是一个工具库，可以为我们提供获取和事件相关的信息。

**获取事件对象：**

有兼容问题，处理方法：var e = window.event||e;

**2. EVENT对象的属性和方法**

**公共属性和方法：**

Target，返回触发此事件的元素；

有兼容问题，处理方法：Var t = e.target||e.srcElement;

Type,返回触发事件的类型，一般用于判断。

preventDefault(),通知浏览器不要执行与事件关联的默认动作。

有兼容问题，处理方法：

If(e.preventDefault){

e.preventDefault();

}else{

e.returnValue = false;

}

**3. 鼠标/键盘事件Event属性**

e.altKey 返回当事件被触发时，"ALT" 是否被按下。 返回true 或 false

e.ctrlKey 返回当事件被触发时，"CTRL" 键是否被按下。 返回true 或 false

e.button 返回当事件被触发时,哪个鼠标按钮被点击.0 左键1滚轮2右键(ie全是0）

e.which 该属性声明了被敲击的键生成的 Unicode 字符码(ascii码)（ie不支持）

e.keyCod可以解决which的兼容问题。

e.clientX 返回当事件被触发时，鼠标指针相对于浏览器可视区域的水平坐标。

e.clientY 返回当事件被触发时，鼠标指针相对于浏览器可视区域的垂直坐标。

e.screenX 返回当某个事件被触发时，鼠标指针相对于电脑屏幕的水平坐标。

e.screenY 返回当某个事件被触发时，鼠标指针相对于电脑屏幕的垂直坐标。

4 .

e.offsetX,e.offsetY 给谁加就相当与 以他为0,0点的距离

e.offsetLeft,e.offsetTop 没有加position，就以body为标准，父元素加了就以父元素为标准

e.offsetWidth,e.offsetHeight 元素的宽和高

e.clientX,e.clientY 可视化区域的距离

e.offsetX在火狐不兼容， = e.clientX-e.offsetLeft;

Var height = document.documentElement.scrollTop||document.body.scrollTop;

e.pageX 也存在兼容问题 = e.clientX + height;

获取body节点对象：

标准：Document.body， ie：document.documentElement

**事件监听**

滚轮事件一定要用事件监听写。

事件监听绑定工具：

Function addEvent(dom,type,fn){

If(e.addEventListener)//标准{

dom.addEventListener(type,fn); //捕捉

}else{//ie

dom.attachEvent(“on” + type,fn); //冒泡

}

}

**除了前面的事件兼容问题还有：**

1.which： which||e.keyCode；

2.获取指定样式属性的值

Function getStylePropertyValue(dom,propertyName){

If(window.getComputedStyle){

Return getComputedStyle(dom,null)[propertyName];

}else{

Return dom.currentStyle[propertyName];

}

}

1. 判断滚轮事件中往什么方向滚动

e.wheelDelta = 120向上滚动和火狐的e.detail = -3;

e.wheelDelta = -120向下滚动和火狐的e.detail = 3;

思路的产生必然基于需求

**案例：愤怒的小鸟**

思路：

第一步：获取所要操作的节点对象

第二步：为文档添加一个键盘事件（onkeydown）

第三步：通过onkeydown事件所提供的事件对象获取keyCode属性来甄别所按下的是否是上下左右键。

第四步：根据上下左右键更改图片的坐标和旋转的方向。

**案例：滚轮切换**

思路：

第一步：获取所要操作的节点对象

第二步：为document添加一个鼠标滚轮滚动事件

第三步：

获取浏览器可视区域的高度：document.body.clientHeight

甄别滚轮滚动方向,然后决定是否改变bigBox的top值（负值）

第四步：循环遍历li节点对象，并为其添加一个点击事件-->改变bigBox的top值

**案例：拖拽**

思路：

第一步：获取所要操作的节点对象

第二步：为要点击的标签节点对象添加一个鼠标按下事件（onmousedown）

第三步：在鼠标按下事件程序中为document添加一个鼠标移动事件（onmousemove）

第四步：在鼠标移动的过程中，根据鼠标的坐标更改box盒子的坐标

第五步：当鼠标弹起，移除document的onmousemove事件

document.onmousemove = null;//移除移动事件

这里涉及到了“垃圾回收器”，它会自动的清理无用数据

函数属于引用类型的数据

第六步：回放

用对象obj接收X，Y的坐标，然后用数组.push();接收obj，当点击回放时，是一段播放的过程，肯定用了定时器，回放第一个就截取第一个坐标arr.shift();当arr.shift();返回为undefined这时就关闭定时器。

**Cookie存储（会话跟踪技术，本地存储技术）**

http协议：一个数据传输格式，即电脑与电脑之间的规定与规则；

http协议是无状态的，也就是说没有记忆力，（下次在访问的时候，他会认为你是第一次来访问的）。

若是后面的处理需要之前的信息，没有记忆力，就要每次都要重传，这样导致数据量增大。就是因为如此，才有了cookie。

标准格式的cookie 的属性有name = value，expires，domain，path，secure。

Name = value是必须有的后面的属性可以不写。

完整形式：[]中是可选项

document.cookie =“name=value[;expires=date][;path=path-to-resource][;domain=域名][;secure]”

只有当前页面在cookie指定路径或指定路径的子路径，才能象服务器发送cookie。

若删除的cookie的路径的层级比当前文件要高，则必须指定cookie的路径来删除，否则无法删除。

解码：encodeURIComponent(str);

编码：decodeURIComponent(str);

**添加和修改：**

Function setCookie(obj){

Var cookieStr = encodeURIComponent(obj.name) + “=” + encodeURIComponent(obj.value);

If(obj.expires){//过期时间

cookieStr+=”;expires=” + obj.expires;

}

If(obj.path){//路径

cookieStr+=”;path=” + obj.path;

}

If(obj.domain){//域

cookieStr+=”;domain=” + obj.domain;

}

If(obj.secure){//安全

cookieStr+=”;secure=”;

}

Document.cookie = cookieStr;

}

**删除**

Function removeCookie(obj){

Var date = new Date();

date.setSeconds(-1);

setCookie({

Name:obj.name,

Value:obj.value,

Expires:date,

Path:obj.path;

Domain:obj.domain,

Secure:obj.secure

});

}

**查找**

Function getCookie(cname){

Var cookieStr = document.cookie;

Var cookieArr = cookieStr.split(“; ”);

For(var i = 0;i<cookieArr.length;i++){

Var arr = cookieArr[i].split(“=”);

If(cname==decodeURIComponent(arr[0]) ){

Return decodeURIComponent(arr[1]);

}

}

Return “”;

}

**JSON的使用**

JSON.parse(字符串数组或对象);

JSON.string(数组和对象);

Eval(“1+1” );//2

它会自动的把字符串代码转换成真正意义上的js代码

节点对象.scrollIntoView();//把需要显示的节点对象显示在可视区域中