事件

1.事件对象

事件: 对某个元素的某种操作

事件对象：当某个事件触发时产生的对象，就是事件对象。event使用前提，必须有事件

                 不同的事件产生的事件对象不同。

事件对象的兼容： var e = evt || event;

事件对象拥有该事件相关的属性和方法

Demo:

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8" />

        <title></title>

    </head>

    <body>

        <script type="text/javascript">

            //事件对象 event

            document.onkeyup = function(evt){

                var e =  evt || event;

                alert(e);

            }

        </script>

    </body>

</html>

Demo:小老虎跳一跳



<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8" />

        <title></title>

        <style>

            #box{

                height: 100px;

                width: 100px;

                background: url(img/3.jpg);

                background-size: 100px 100px;

                position: absolute;

                left: 200px;

                top: 600px;

            }

        </style>

    </head>

    <body>

        <div id="box" onclick="jump()">

        </div>

    </body>

</html>

<script>

    var d = document.getElementById("box");

    function jump(){

        d.style.top = d.offsetTop - 200 + "px";

        setTimeout(function(){

            d.style.top = "600px";

        },2000);

    }

</script>

2.鼠标事件对象的属性

坐标属性：

pageX / pageY  相对于整个文档顶部和左侧的坐标

clientX / clientY  相对于局部窗口的左侧和顶部的坐标

offsetX /offsetY  相对于内部元素的距离左侧和顶部的坐标 常用于拖拽

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            \*{

                margin: 0;

                padding: 0;

            }

            body{

                height: 1000px;

            }

            #div1{

                height: 200px;

                width: 200px;

                border: 1px solid red;

                position: absolute;

                left: 200px;

            }

        </style>

        <script>

            document.onclick = function(evt){

                var e = evt || event;

                console.log("client" + e.clientX + "," + e.clientY);//只记录当前的局部页面算

                console.log("page" + e.pageX + "," + e.pageY);//从document页面算起...习惯上用page

                console.log("offset" + e.offsetX + "," + e.offsetY);

            }

        </script>

    </head>

    <body>

        <div id="div1">

        </div>

    </body>

</html>

3.案例图片跟随（练习）

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            #myimg{

                height: 100px;

                width: 100px;

                position: absolute;

                background: url(img/clothes.jpg);

                background-size: 100px 100px;

            }

        </style>

    </head>

    <body>

        <div id="myimg">

        </div>

    </body>

</html>

<script>

    document.onmousemove = function(evt){

        //获取实

        var e = evt || event;

        var oImg = document.getElementById("myimg");

        //操作图片X和Y的位置

        oImg.style.left = e.pageX - oImg.offsetWidth/2 + "px";

        oImg.style.top = e.pageY -oImg.offsetHeight/2 + "px";

    }

</script>

4.键盘事件

onkeyUp: 用户释放某一个按键时发生

onkeyDown: 用户按下按键时发生,任何一个按键即可发生事件

onkeyPress: 用户按下按键，并且产生一个字符时发生（意思就是说按ctrl这样的按键没有效果的）

事件对象的属性:

keyCode: 获取键盘按键值   字母 对应其ASCII值识别

A:65  a:97   0:48      回车:13

【注意】:keyPress 的ctrl+回车返回值为10

    document.onkeypress = function(evt){

        var e = evt || event;

        if(e.keyCode == 13 || e.keyCode == 10 && e.ctrlKey){

            alert("send");

        }

    }

ctrlKey:返回当事件被触发时，"Ctrl" 是否被按下,返回值为true or false

【注意】:键盘检测兼容写方法

var key = e.keyCode || e.which || e.charCode;

5.事件流

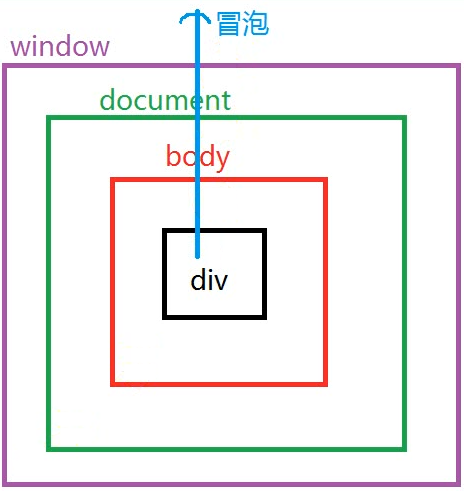
当某个事件执行时，从子元素向父元素触发 或 从父元素向子元素触发 称为事件流

事件流的两种模式：

事件冒泡：从子元素向父元素触发  --->当某个事件触发时，同样的事件会向父元素触发。

                                                            但并不是所有事件都会产生冒泡问题 onfocus onblur onload不会产生冒泡问题

事件捕获：从父元素向子元素触发



<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            #div1{

                height: 100px;

                width: 100px;

                border: 1px solid  red;

            }

            #div2{

                height: 300px;

                width: 300px;

                border: 1px solid  red;

            }

        </style>

    </head>

    <body>

        <div id="div2">

            <div id="div1">

            </div>

        </div>

    </body>

</html>

<script type="text/javascript">

    //文档结构: window document body body元素

    var father = document.getElementById("div2");

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

    window.onclick = function(){

        alert("window被点击");

    }

    document.onclick = function(){

        alert("document被点击");

    }

    document.body.onclick = function(){

        alert("body被点击");

    }

    father.onclick = function(){

        alert("father被点击");

    }

    son.onclick = function(){

        alert("son被点击");

    }

</script>

6.阻止事件冒泡

e.stopPropagation?e.stopPropagation():e.cancelBubble = true;  通过事件对象调用

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            ul{

                display: none;

            }

        </style>

    </head>

    <body>

        <button id="btn">按钮</button>

        <ul id="list">

            <li>大黄</li>

            <li>钢铁侠</li>

            <li>蜘蛛侠</li>

        </ul>

    </body>

</html>

<script>

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

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

    btn.onclick = function(evt){

        var e = evt || event;

        //阻止冒泡 时间冒泡的兼容

        // e.stopPropagation ? e.stopPropagation():e.cancelBubble = true;

                //函数为真

        //document.write(e.stopPropagation);

        //阻止冒泡

        e.stopPropagation();

        ul.style.display = "block";

    }

    //阻止冒泡

    document.onclick = function(){

        ul.style.display = "none";

    }

</script>

7.阻止浏览器默认事件

e.preventDefault?e.preventDefault():e.returnValue = false;

return false;

阻止鼠标右键

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style type="text/css">

            #box{

                height: 100px;

                width: 100px;

                background: red;

                position: absolute;

            }

        </style>

    </head>

    <body>

        <div id="box">

        </div>

    </body>

</html>

<script>

    var d = document.getElementById("box");

    //单机右键 控制div的位置

    document.oncontextmenu = function(evt){

        //console.log("嘿嘿");

        var e = evt || event;

        d.style.left = e.pageX + "px";

        d.style.top = e.pageY + "px";

        //e.preventDefault?e.preventDefault():e.returnValue = false;

        return false;

    }

</script>

阻止超链接跳转

<body>

    <a href = "http://www.baidu.com">跳转</a>

</body>

</html>

<script type="text/javascript">

    var x = document.querySelector("a");

    x.onclick = function(){

        console.log(1);

        return false;

    }

</script>

作业

0.案例一遍，整理笔记

1.跟随鼠标的提示框

2.模拟selelct下拉菜单

8.事件绑定的方式

1.obj.onclick = function(){}

2.<div onclick = "fun()">

3.addEventListener()     事件监听

9.事件监听 addEventListener() 主流高版本浏览器

事件监听的好处

1---可以为同样的元素绑定多次同一个事件

2---程序员可以使用事件监听的方式 确定触发的过程是冒泡还是捕获

事件源.addEventListener("去掉on的事件",function(){},true/false)  默认是false 冒泡

//     document.addEventListener("click",function(){

//         alert("1");

//     });

//     document.addEventListener("click",function(){

//         alert("2");

//     });

//     document.onclick = function(){

//         alert("1");

//     }

//     document.onclick = function(){

//         alert("2");

//     }

    document.addEventListener("click",function(){

            alert("document");

    },true);

    window.addEventListener("click",function(){

            alert("window");

    },true);

一个元素同时拥有捕获和冒泡的情况下，执行顺序是什么？

    document.addEventListener("click",function(){

            alert("document捕获");

    },true);

    document.addEventListener("click",function(){

            alert("document冒泡");

    },false);

    window.addEventListener("click",function(){

            alert("window捕获");

    },true);

    window.addEventListener("click",function(){

            alert("window冒泡");

    },false);

先捕获，后冒泡

10.事件监听兼容

//    ie的事件监听 没有第三个参数 默认冒泡

//                          该参数不省略on

//     document.attachEvent("onclick",function(){

//         alert("document");

//     });

----------------------------------------------------

//兼容函数         谁  事件  做什么事

    function addEvent(obj,type,callBack){

        if(obj.addEventListener){//非IE版本

            obj.addEventListener(type,callBack);

        }else{//IE版本

            obj.attachEvent("on"+type,callBack);

        }

    }

    addEvent(document,"click",function(){alert("document")});

11.事件委托

委托：让别人去做

事件委托：某个事件让其他元素来完成

    例如：页面上有1000个li，为每一个li添加单机事件    使用委托只需要在li父级上加一次事件就可以

委托的好处：

   1. 把某个事件加到父元素上，提高程序的执行效率

   2. 动态创建的元素 可以在创建元素的函数体外部为其添加事件

委托的机制：

        利用事件冒泡（常见） 或者 事件捕获

        不是所有事件都可以实现事件委托  常见到也就那么几个

委托的实现方法：

父级元素.事件 = function(){

}

    <body>

        <ul id = "list">

            <li>1</li>

            <li>2</li>

            <li>3</li>

            <li>4</li>

            <li>5</li>

        </ul>

    </body>

<script>

    //利用事件委托 为每一个li添加单机事件

    //处理事件程序为:改变当前操作的li的背景颜色

    //委托里不能用this

    var oUl = document.getElementById("list");

    oUl.onclick = function(evt){

        var e = evt || event;

        //注意 实际操作的是li

        //获取事件源

        //targat就代表了当前操作的li

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

        //alert(targat);

        if(targat.tagName == "LI"){

            targat.style.backgroundColor = "pink";

        }

    }

</script>

或

父级元素.        (事件,function(){

})

var oUl = document.getElementById("list");

    oUl.addEventListener("click",function(evt){

        var e = evt || event;

        //注意 实际操作的是li

        //获取事件源

        //targat就代表了当前操作的li

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

                //alert(targat);

        if(targat.tagName == "LI"){

            targat.style.backgroundColor = "pink";

        }

    });

12.事件委托的好处

//为所有的li添加高亮事件

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

        <ul id = "ulist">

            <li>1</li>

            <li>2</li>

            <li>3</li>

            <li>4</li>

            <li>5</li>

        </ul>

    <body>

    </body>

</html>

<script>

    //为所有的li添加高亮事件

    var list = document.getElementsByTagName("li");

    for(var i=0; i<list.length; i++){

        list[i].onmouseover = function(){

            this.style.backgroundColor = "red";

        }

        list[i].onmouseout = function(){

            this.style.backgroundColor = "";

        }

    }

</script>

//动态添加li元素

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

        <button id="btn">添加</button>

        <ul id = "ulist">

            <li>1</li>

            <li>2</li>

            <li>3</li>

            <li>4</li>

            <li>5</li>

        </ul>

    <body>

    </body>

</html>

<script>

    //为所有的li添加高亮事件

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

    var list = document.getElementsByTagName("li");

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

    //创建li

    btn.onclick = function(){

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

        e.innerHTML = "新添加的元素";

        ulist.appendChild(e);

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

           list[i].onmouseover = function(){

                this.style.backgroundColor = "red";

            }

           list[i].onmouseout = function(){

                this.style.backgroundColor = "";

           }

    }

}

</script>

完全版

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

        <button id="btn">添加</button>

        <ul id = "ulist">

            <li>1</li>

            <li>2</li>

            <li>3</li>

            <li>4</li>

            <li>5</li>

        </ul>

    <body>

    </body>

</html>

<script>

    //为所有的li添加高亮事件

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

    var list = document.getElementsByTagName("li");

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

    //创建li

    btn.onclick = function(){

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

        e.innerHTML = "新添加的元素";

        ulist.appendChild(e);

    }

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

//         list[i].onmouseover = function(){

//             this.style.backgroundColor = "red";

//         }

//         list[i].onmouseout = function(){

//             this.style.backgroundColor = "";

//         }

//     }

    ulist.onmouseover = function(evt){

        var e = evt || event;

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

        if(target.tagName == "LI"){

            target.style.backgroundColor = "red";

        }

    }

    ulist.onmouseout = function(evt){

        var e = evt || event;

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

        if(target.tagName == "LI"){

            target.style.backgroundColor = "";

        }

    }

</script>

13.拖拽效果

拖拽思路： onmousedown onmousemove  onmouseup

1.首先为需要拖拽的对象添加一个onmousedown事件

   记录：鼠标点击某个对象时的内部偏移量

   e.offsetX  e.offsetY

2.鼠标在文档上移动

   要想让操作的元素动起来，该元素必须有定位

   移动的过程，实际上改变，元素的left和top

3.停止移动，需要触发onmouseup鼠标抬起时，取消移动

  document.onmousemove = null;

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            \*{

                margin: 0;

            }

            #box{

                height: 100px;

                width: 100px;

                position: absolute;

                cursor: move;

                background: url(img/3.jpg);

                background-size: 100px 100px;

            }

        </style>

    </head>

    <body>

        <div id="box">

        </div>

    </body>

</html>

<script>

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

    box.onmousedown = function(evt){

    //为了获得内部元素距离其左顶点坐标

        var e = evt || event;

        var ofX = e.offsetX;

        var ofY = e.offsetY;

        document.onmousemove = function(evt){

            var e = evt || event;

            box.style.left = e.pageX - ofX + "px";

            box.style.top = e.pageY - ofY + "px";

        }

        document.onmouseup = function(){

            document.onmousemove = null;

        }

    }

</script>

14.拖拽的边界问题

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            \*{

                margin: 0;

            }

            #box{

                height: 100px;

                width: 100px;

                position: absolute;

                cursor: move;

                background: url(img/3.jpg);

                background-size: 100px 100px;

            }

        </style>

    </head>

    <body>

        <div id="box">

        </div>

    </body>

</html>

<script>

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

    box.onmousedown = function(evt){

        var e = evt || event;

        var ofX = e.offsetX;

        var ofY = e.offsetY;

        document.onmousemove = function(evt){

            var e = evt || event;

            var x = e.pageX - ofX;

            var y = e.pageY - ofY;

            if(x<0){

                x = 0;

            }

            var maxLeft = window.innerWidth - box.offsetWidth;

            if(x>maxLeft){

                x = maxLeft;

            }

            if(y<0){

                y = 0;

            }

            var maxTop = window. - box.offsetHeight;

            if(y>maxTop){

                y = maxTop;

            }

            box.style.left = x + "px";

            box.style.top = y + "px";

        }

        document.onmouseup = function(){

            document.onmousemove = null;

        }

    }

</script>

15.JSON对象

json: 轻量级存储工具，是一种跨平台的数据交互格式，

        作用：存储数据

json对象定义:

        var json = {键:值,......};

        说明:严格的json对象 键必须用双引号引起来

               json的值可以是任意类型的

json 操作 赋值和取值

        取值： json.键

                    json["键"]  用于for in

        遍历取值   for in

<script>

    var myJson = {

        "name":"laowang",

        "age":18

    };

    //document.write(myJson.name);

    //document.write(myJson["name"]);

    for(var t in myJson){

        console.log(t);

        console.log(myJson[t]);

    }

</script>