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>

作业

0.课堂案例，笔记整理

1.鼠标移动轨迹

2.窗口拖拽

  实现拖拽功能

  实现最大化，最小化

3.实现复选框的操作

