 插件编写及JQ入门

1.插件

    插件是一个独立的功能，与函数相比，他是更加完整的功能。

假设将一个汽车比作项目：

    函数好比汽车的螺丝，螺帽。

    而插件就好比发动机，变速箱。

原生js放大镜插件：

class Magnifier {

        constructor(newSmallBox, newBigBox, newMask) {

            this.smallBox = newSmallBox;

            this.bigBox = newBigBox;

            this.mask = newMask;

        }

        onmouseover() {

            let that = this;

            this.smallBox.onmouseover = function() {

                that.bigBox.style.display = "block";

                that.mask.style.display = "block";

            }

        }

        onmouseout() {

            let that = this;

            this.smallBox.onmouseout = function() {

                that.bigBox.style.display = "none";

                that.mask.style.display = "none";

            }

        }

        onmousemove() {

            let that = this;

            this.smallBox.onmousemove = function(evt) {

                let e = evt || event;

                let left = e.pageX - this.offsetLeft - that.mask.offsetWidth / 2;

                let top = e.pageY - this.offsetTop - that.mask.offsetHeight / 2;

                if (left < 0) {

                    left = 0;

                }

                let maxLeft = this.offsetWidth - that.mask.offsetWidth;

                if (left > maxLeft) {

                    left = maxLeft;

                }

                if (top < 0) {

                    top = 0;

                }

                let maxTop = this.offsetHeight - that.mask.offsetHeight;

                if (top > maxTop) {

                    top = maxTop;

                }

                that.mask.style.left = left + "px";

                that.mask.style.top = top + "px";

                let x = that.bigBox.offsetWidth \* left / that.mask.offsetWidth;

                let y = that.bigBox.offsetHeight \* top / that.mask.offsetHeight;

                that.bigBox.style.backgroundPositionX = -x + "px";

                that.bigBox.style.backgroundPositionY = -y + "px";

            }

        }

        eventBind() {

            this.onmouseover();

            this.onmouseout();

            this.onmousemove();

        }

}

测试:

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8" />

        <title></title>

        <style>

            \*{

                margin: 0;

                padding: 0;

            }

            #small-box{

                width: 400px;

                height: 400px;

                background: url(img/13.jpg) no-repeat;

                background-size: 400px 400px;

                position: absolute;

                left: 100px;

                top: 150px;

                cursor: move;

            }

            #big-box{

                width: 400px;

                height: 400px;

                background: url(img/13.jpg) no-repeat;

                background-size: 1600px 1600px;

                position: absolute;

                left: 600px;

                top: 150px;

                display: none;

            }

            #mask{

                width: 100px;

                height: 100px;

                position: absolute;

                background: yellow;

                opacity: 0.3;

                display: none;

            }

        </style>

    </head>

    <body>

        <div id="small-box">

            <div id="mask">

            </div>

        </div>

        <div id="big-box">

        </div>

    </body>

</html>

<script src="js/mfPulgin.js"></script>

<script>

    let oSmallBox = document.getElementById("small-box");

    let oBigBox = document.getElementById("big-box");

    let oMask = document.getElementById("mask");

    new Magnifier(oSmallBox,oBigBox,oMask).eventBind();

</script>

原生js弹出层插件：

     class PopPlugin {

         constructor(newBox) {

             this.box = newBox;

             this.btnDiv = null;

         }

         setBoxPosition() {

             this.box.style.display = "block";

            //要想获取offsetWidth display必须设置为block

             this.box.style.left = window.innerWidth / 2 - this.box.offsetWidth / 2 + "px";

             this.box.style.top = window.innerHeight / 2 - this.box.offsetHeight / 2 + "px";

             this.creatBtn();

         }

         creatBtn() {

             if (this.btnDiv == null) {

                 this.btnDiv = document.createElement("button");

             }

            //记得加定位

             this.btnDiv.style.position = "absolute";

             this.btnDiv.style.width = 50 + "px";

             this.btnDiv.style.height = 25 + "px";

             this.btnDiv.innerHTML = "关闭";

             this.box.appendChild(this.btnDiv);

            //必须先有元素才有offsetWidth 这些东西

             this.btnDiv.style.left = this.box.offsetWidth - this.btnDiv.offsetWidth + "px";

             this.closeBox();

         }

         closeBox(){

             let that = this;

             this.btnDiv.onclick = function(){

                 that.box.style.display = "none";

             }

         }

     }

测试:

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            #box{

                width: 350px;

                height: 400px;

                background: skyblue;

                position:absolute;

                display: none;

            }

        </style>

    </head>

    <body>

        <button>打开</button>

        <div id="box">

        </div>

    </body>

</html>

<script src="js/弹出层插件.js"></script>

<script type="text/javascript">

    let btn = document.querySelector("button");

    let box = document.querySelector("#box");

    btn.onclick = function() {

        let p = new PopPlugin(box);

        p.setBoxPosition();

    }

</script>

</script>

原生jsajax插件:

    class myAjax{

        constructor(newObj) {

            this.obj = newObj;

        }

        ajax() {

            //1.创建对象 通过兼容模式

            let xhr;

            if (window.ActiveXObject) {

                //ie

                xhr = new ActiveXObject("Microsoft.XMLHttp");

            } else {

                //非ie

                xhr = new XMLHttpRequest();

            }

            this.obj.type = this.obj.type.toLowerCase();

            if (this.obj.type == "get") {

                let urlParam = this.obj.url;

                urlParam += "?" + this.obj.data;

                xhr.open(this.obj.type,urlParam, this.obj.isAsyn);

                xhr.send();

            } else if (this.obj.type == "post") {

                xhr.open(this.obj.type, this.obj.url, this.obj.isAsyn);

                //这句话必须抄 post的请求头

                xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

                xhr.send(this.obj.data);

            }

            let that = this;

            xhr.onreadystatechange = function() {

                if (xhr.readyState == 4 && xhr.status == 200) {

                    that.obj.callBack(xhr.responseText);

                }

            }

        }

    }

测试:

<script src="js/ajax插件.js"></script>

<script>

    function fun(str){

        console.log(str);

    }

    new myAjax({

        "isAsyn":true,

        "type":"get",

        "url":"test.php",

        "data":"name=laowang",

        "callBack":fun

    }).ajax();

</script>

原生js选择器插件:

    class Select {

        //静态

        static $(str) {

            if (str.charAt(0) == "#") {

                return document.getElementById(str.substring(1, str.length));

            } else if (str.charAt(0) == ".") {

                return document.getElementsByClassName(str.substring(1, str.length))[0];

            } else {

                return document.getElementsByTagName(str)[0];

            }

        }

    }

测试：

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

        <div id="box">

            123

        </div>

        <div class="box">

            456

        </div>

        <p>

            789

        </p>

    </body>

</html>

<script src="js/选择器插件.js"></script>

<script>

    // console.log(Select.$("#box").innerHTML);

    Select.$("#box").style.background = "red";

    console.log(Select.$(".box").innerHTML);

    console.log(Select.$("p").innerHTML);

</script>

2.jQuery入门



jQuery的特点：

jQuery 是一个快速的简洁的javascript框架，可以简化查询DOM对象、处理事件、制作动画、处理Ajax交互过程。

1>提供了强大的功能函数

2>解决浏览器兼容性问题

3>纠正错误的脚本知识

4>体积小，使用灵巧(只需引入一个js文件)

5>易扩展、插件丰富

jQuery的作用：

\* 程序员的角度：简化JavaScript和Ajax编程，能够使程序员从设计和书写繁杂的JS应用中解脱出来，将关注点转向功能需求而非实现细节上，从而提高项目的开发速度。

\* 用户体验的角度：改善了页面视觉效果，增强了与页面的交互性，体验更绚丽的网页物资。

方便地选择页面元素(模仿CSS选择器更精确、灵活)

动态更改页面样式/页面内容(操作DOM，动态添加、移除样式)

控制响应事件(动态添加响应事件)

提供基本网页特效(提供已封装的网页特效方法)

快速实现通信(ajax)。



如何引入JQuery包：

\*   jQuery库文件不需要安装，只需使用<script>标签引入到HTML文件中，拿到jQuery的库文件后，就跟自己写的JS文件同样的引入方式，即可，以下是几种引入方式

    \*   a.引入本地的Jquery

    \*   b.引入cdn在线提供的库文件（稳定可靠高速）

<script src="http://libs.baidu.com/jquery/2.0.0/jquery.js"></script>

      写第一个JQuery案例

  <script type=“text/javascript” src=“”></script>

  <script type=“text/javascript”>

  $(function(){

  alert(“jQuery 你好!”);

  });

  </script>

第一个Jquery代码（讲解$(function(){})：

  <script type=“text/javascript” src=“jquery.js”></script>

  写第一个JQuery案例

  <script type=“text/javascript”>

  $(function(){

      alert(“jQuery 你好!”);

  });

  </script>

\* 在JQuery库中，$是JQuery的别名，$()等效于就jQuery()。即以上代码等价于：jQuery(function(){alert(“jQuery 你好!”);});

$()相当于页面初始化函数，当页面加载完毕，会执行$()，即jQuery()。

\* $(function(){ })

当页面加载完毕，会执行$()，即jQuery()。

$(function(){内容}) 是$(document).ready(function(){

    内容

});的简写。

\* $(document).ready(function(){})和window.onload 的区别

\* ready表示文档已加载完成（不包含图片等非文字媒体文件）

\* onload是指页面包含图片在内的所有元素都加载完成

\* $(document).ready(function(){})要比window.onload先执行

\* jQuery包装集

在jQuery的世界中将所有的对象，无论是一个还是一组，都封装成一个jQuery包装集，即集合。也就是说，$()的返回结果都是集合，不是单个对象。

      比如获取包含一个元素的jQuery包装集。

                //testDiv是页面某元素的id

      var jQueryObject = $(“#testDiv");

            虽然，通过id获得的是一个元素对象，但是依然以集合的方式返回，只不过，集合中只有一个元素而已。

\* jQuery包装集与原生DOM 对象的互转

    \* jQuery包装集转原生DOM对象

    \* 原生DOM对象转jQuery包装集

jQuery包装集是一个集合，所以我们可以通过索引器访问其中的某一个元素

//testDiv是页面某元素的id

var domObject = $(“#testDiv”)[0];//$(“#testDiv”).get(0);

//testDiv是页面某元素的id

var div = document.getElementById("testDiv");

var domToJQueryObject = $(div);

jQuery选择器：

\* jQuery提供了异常强大的选择器用来帮助我们获取页面上的对象，并且将对象以jQuery包装集的形式返回（就像DOM中获得页面的节点一样）

“$”符号在jQuery中代表对jQuery对象的引用，返回jQuery对象

如：

//根据id获取jQuery包装集

var jQueryObject = $(“#testDiv”);

1.基础选择器

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            \*{

                height: 25px;

            }

        </style>

    </head>

    <body>

        <div id="box1" class = "box">

            1

        </div>

        <div id="box2" class = "box">

            1

        </div>

        <div id="box3" class = "box">

            1

        </div>

        <p>2</p>

        <p>2</p>

        <span>3</span>

        <strong>4</strong>

    </body>

</html>

<script src="jQuery.js"></script>

<script>

    $(function(){

        //1.ID选择器(只代表唯一一个元素)

        //两种写法

        //a.js写法

        //$("#box1").css("backgroundColor","red");

        //b.css写法

        //$("#box1").css({backgroundColor:"yellow"});

        //2.类选择器

        //$(".box").css({backgroundColor:"red"});

        //3.标签选择器

        //$("p").css({backgroundColor:"red"});

        //4.群组选择器

        //$("strong,p,span").css({backgroundColor:"red"});

        //5.通配符选择器(这是选择全部元素)

        $("\*").css({backgroundColor:"red"});

    });

</script>

2.层次选择器

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

    <body>

        <div id="box1" class="box">

            1

        </div>

        <div id="box2" class="box">

            2

        </div>

        <div id="box3" class="box">

            3

        </div>

        <div id="box4" class="box">

            4<p>

                <div id="box5" class="box">5</div>

            </p>

        </div>

    </body>

    </body>

</html>

<script src="jQuery.js"></script>

<script>

    $(document).ready(function(){

        //1.后代选择器

        //$("body div").css({backgroundColor:"red"});

        //2.子选择器

        //$("body>div").css({    backgroundColor: "red"});

        //3.相邻选择器

        //$("#box2+div").css({backgroundColor:"skyBlue"});

        //4.兄弟选择器

        //$("#box2~div").css({background:"yellow"});

    });

</script>

3.属性选择器

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

        <div id="box1" class="">

            1

        </div>

        <div id="" class="">

            1

        </div>

    </body>

</html>

<script src="jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    $(function(){

        //3.属性选择器

        //div带有class属性的选择器

        //$("div[class]").css("backgroundColor","red");

        //div带有id属性的选择器

        //$("div[id]").css("backgroundColor","green");

        //指定id为box1

        //$("div[id='box1']").css("backgroundColor","yellow");

        //还可以同时存在多个属性

        //$("div[class][id]").css("backgroundColor","skyblue");

    });

</script>

4.伪类选择器

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

        <div id="">

            0

        </div>

        <div id="">

            1

        </div>

        <div id="box2">

            2

        </div>

        <div id="box3">

            3

        </div>

        <div id="">

            4

        </div>

    </body>

</html>

<script src="jQuery.js"></script>

<script>

    //选中偶数行 下标从0开始

    //$("div:even").css("backgroundColor","skyblue");

    //选中奇数行

    //$("div:odd").css("backgroundColor","skyblue");

    //选中第一个元素

    //$("div:first").css("backgroundColor","skyblue");

    //选中最后一个元素

    //$("div:last").css("backgroundColor","skyblue");

    //eq(n)选中第N个div元素

    //$("div:eq(1)").css("backgroundColor","skyblue");

    // 也可以这样写

    //$("div").eq(1).css("backgroundColor","skyblue");

    //div里除了#box2元素,其他都被影响

    //$("div:not('#box2')").css("backgroundColor","skyblue");

    // 也可以这样写

    //$("div").not('#box3').css("backgroundColor","skyblue");

    //gt(n) 匹配出大于n

    // $("div:gt(1)").css("backgroundColor","skyblue");

    //lt(n) 匹配出小于n

    //$("div:lt(3)").css("backgroundColor","skyblue");

</script>

5.内容选择器

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

        <div id="">

            hello world

        </div>

        <div id="">

            1

        </div>

        <div id="">

            2

        </div>

        <div id="">

            3

            <span id="s"></span>

        </div>

        <div id=""></div>

    </body>

</html>

<script src="jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    //contains

    //字符串中存在hello world的元素

    //$("div:contains('hello world')").css({background:"red"});

    //empty

    //找内容为空的选择器,空格什么东西都不能有

    //$("div:empty").html("666");

    //has(选择器)

    //根据子元素进行筛选

    $("div:has('#s')").css("background","red");

</script>

6.可见性选择器

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

        <table>

            <tr style="display:none">

                <td>Value 1</td>

            </tr>

            <tr>

                <td>Value 2</td>

            </tr>

        </table>

    </body>

</html>

<script src="jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    //改变不可见元素的样式

     $("tr:hidden").css("background","green");

    //改变可见元素的样式

    $("tr:visible").css("background","red");

</script>

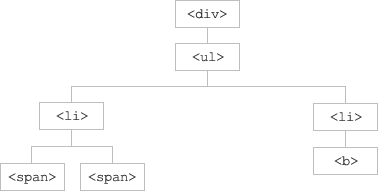
3.jQuery遍历

什么是遍历？

jQuery 遍历，意为“移动”，用于根据其相对于其他元素的关系来“查找”（或选取）HTML 元素。以某项选择开始，并沿着这个选择移动，直到抵达您期望的元素为止。

下图展示了一个家族树。通过 jQuery 遍历，您能够从被选（当前的）元素开始，轻松地在家族树中向上移动（祖先），向下移动（子孙），水平移动（同胞）。这种移动被称为对 DOM 进行遍历。

图示解释：



\* <div> 元素是 <ul> 的父元素，同时是其中所有内容的祖先。

\* <ul> 元素是 <li> 元素的父元素，同时是 <div> 的子元素

\* 左边的 <li> 元素是 <span> 的父元素，<ul> 的子元素，同时是 <div> 的后代。

\* <span> 元素是 <li> 的子元素，同时是 <ul> 和 <div> 的后代。

\* 两个 <li> 元素是同胞（拥有相同的父元素）。

\* 右边的 <li> 元素是 <b> 的父元素，<ul> 的子元素，同时是 <div> 的后代。

\* <b> 元素是右边的 <li> 的子元素，同时是 <ul> 和 <div> 的后代。

提示：祖先是父、祖父、曾祖父等等。后代是子、孙、曾孙等等。同胞拥有相同的父。

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

        <div id="box0">

            0

            <p>00</p>

            <div id="">

            </div>

        </div>

        <div id="box1">

            1

            <p>11</p>

        </div>

        <div id="box2">

            2

            <p>22</p>

        </div>

        <div id="box3">

            3

            <p>33</p>

        </div>

        <div id="box4">

            4

            <p>44</p>

        </div>

    </body>

</html>

<script src="jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    //next下一个元素

    //$("#box2").next().css({background:"red"});

    //nextAll下面所有元素

    //$("#box2").nextAll().css({background:"red"});

    //prev上一个元素

    //$("#box2").prev().css({background:"red"});

    //prevAll上面所有元素

    //$("#box2").prevAll().css({background:"red"});

    //该方法必须传参

    //Dom元素的遍历

    //find("参数") 寻找body下的所有div  后代遍历的方法

    //$("body").find("div").css({background:"yellow"});

    //children body的div会被影响

    //$("body").children("div").css({background:"yellow"});

    //children 不传参时默认传递通配符

    //$("body").children().css({background:"yellow"});

    //parent找父元素

    //$("p").eq(0).parent().css({background:"yellow"});

    //not 返回不匹配标准的所有元素。

    $("div").not("#box3").css({background:"yellow"});

</script>

4.jQuery的文本与事件

1.文本的值与元素内容的读写

  val():文本的值

  html():元素的内容

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8" />

        <title></title>

    </head>

    <body>

        文本内容:<input type="text" value="777"/>

        <p>我是隔壁的老王</p>

    </body>

</html>

<script src="js/jQuery.js"></script>

<script>

    $(function(){

        //使用val()方法来读写文本内容来

        //val() == value

        //读

        console.log($("input").val());

        //写

        $("input").val(888);

        //使用html()方法来读写元素的内容来

        //html() == innerHTML

        console.log($("p").html());

    });

2.jQuery的事件全都去除on

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            div{

                background: skyblue;

                height: 300px;

                width: 300px;

            }

        </style>

    </head>

    <body>

        <div id="">

        </div>

    </body>

</html>

<script src="js/jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    jQuery(document).ready(function(){

        $("div").click(function(){

            //注意:$(this)的写法

            $(this).css({background:"red"});

        });

        $("div").mouseover(function(){

            //注意:$(this)的写法

            $(this).css({background:"yellow"});

        });

        $("div").mouseout(function(){

            //注意:$(this)的写法

            $(this).css({background:"blue"});

        });

    });

</script>

jQuery的事件可以传自定义参数:

    $(function(){

        $("div").click({name:"老王",pwd:123456},function(evt){

            let e = evt || event;

            console.log(e.data);

        });

    });

5.jQuery动画

1.简单动画

a.方法：hide,show,toggle

<script>

    $(function(){

        //简单动画hide,show,toggle

        $("button").eq(0).click(function(){

            //第二个参数为在动画完成时执行的函数，每个元素执行一次。

            $("div").hide(1000,function(){

                console.log("heihei");

            });

        });

        $("button").eq(1).click(function(){

            $("div").show(1000);

        });

//         $("button").eq(2).click(function(){

//             $("div").toggle(1000);

//         });

        //做一些奇奇怪怪的事情

        $("button").eq(2).click(function(){

            $("div").toggle(1000,function(){

                $("div").toggle(1000,arguments.callee);

            });

        });

    });

</script>

b.方法：slideUp，slideDown，slideToggle

    //滑动

    $(function() {

        $("button").eq(0).click(function() {

            //第二个参数为在动画完成时执行的函数，每个元素执行一次。

            $("div").slideUp(1000, function() {

                console.log("heihei");

            });

        });

        $("button").eq(1).click(function() {

            $("div").slideDown(1000);

        });

        //         $("button").eq(2).click(function(){

        //             $("div").toggle(1000);

        //         });

        //做一些奇奇怪怪的事情

        $("button").eq(2).click(function() {

            $("div").slideToggle(1000, function() {

                $("div").slideToggle(1000, arguments.callee);

            });

        });

    });

c.淡入淡出：常用 fadeIn fadeOut fadeToggle

       $(function() {

       $("button").eq(0).click(function() {

           //第二个参数为在动画完成时执行的函数，每个元素执行一次。

           $("div").fadeIn(1000, function() {

               console.log("heihei");

           });

       });

       $("button").eq(1).click(function() {

           $("div").fadeOut(1000);

       });

       //         $("button").eq(2).click(function(){

       //             $("div").toggle(1000);

       //         });

       //做一些奇奇怪怪的事情

       $("button").eq(2).click(function() {

           $("div").fadeToggle(1000, function() {

               $("div").fadeToggle(1000, arguments.callee);

           });

       });

   });

2.自定义动画

animate:

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            div {

                background: red;

                width: 100px;

                height: 100px;

                position: absolute;

            }

        </style>

    </head>

    <body>

        <button>按钮1</button>

        <button>按钮2</button>

        <div id="box">

        </div>

    </body>

</html>

<script src="js/jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    $(function(){

        $("button").eq(0).click(function(){

//             $("#box").animate({

//                 width:"500px",

//                 height:50,

//                 opacity:0.3,

//                 left:"200px"

//             },1000).animate({//列队动画

//                 width:"100px",

//                 height:500,

//                 opacity:1,

//                 left:"20px"

//             });

            $("#box").animate({

                left:0

            },1000).animate({//列队动画

                left:500

            },1000).animate({

                top:500

            },1000).animate({

                left:0

            },1000).animate({

                top:30

            });

        });

        $("button").eq(1).click(function(){

            //无参数stop():立刻结束当前动画

            //参数1:清空动画队列

            //参数2:立刻执行完当前动画

            $("#box").stop(true,true);

        });

    });

</script>

6.手风琴

1.简单版本

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style type="text/css">

            \*{

                margin: 0;

                padding: 0;

            }

            ul{

                width: 500px;

                height: 100px;

                background: red;

                position: relative;

                margin: 0 auto;

            }

            ul li{

                list-style: none;

                width: 40px;

                height: 100px;

                background: yellow;

                float:left;

                border: 1px solid black;

            }

        </style>

    </head>

    <body>

        <ul>

            <li></li>

            <li></li>

            <li></li>

            <li></li>

        </ul>

    </body>

</html>

<script src="js/jQuery.js" type="text/javascript" charset="utf-8"></script>

<script>

    $(function(){

        $("li").mouseover(function(){

            $(this).animate({

                width:240

            }).siblings("li").stop().animate({

                width:40

            });

        });

    });

</script>

2.完美版

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style type="text/css">

            \*{

                margin: 0;

                padding: 0;

            }

            ul{

                height: 150px;

                width: 800px;

                position: relative;

                margin: 0 auto;

                background: yellow;

            }

            ul li{

                width:40px;

                height:150px;

                list-style: none;

                overflow: hidden;

                border: 1px solid black;

                float: left;

            }

            img{

                background-size: 240px;

                height:150px;

                width:240px;

            }

        </style>

    </head>

    <body>

        <ul>

            <li>

                <img src="images/1.jpg" >

            </li>

            <li>

                <img src="images/2.jpg" >

            </li>

            <li>

                <img src="images/3.jpg" >

            </li>

            <li>

                <img src="images/4.jpg" >

            </li>

            <li>

                <img src="images/5.jpg" >

            </li>

        </ul>

    </body>

</html>

<script src="js/jQuery.js" type="text/javascript" charset="utf-8"></script>

<script type="text/javascript">

    $(function(){

        $("li").mouseover(function(){

            $(this).animate({

                width:240

            }).siblings("li").stop().animate({

                width:40

            });

        });

    });

</script>

作业：

0.课堂案例，整理笔记

1.表格的各行变色效果

2.jQuery修改五星好评

3.根据文档学习jQueryAjax

4.根据文档学习jQueryDom