JQuery进阶

**jQuery中的Ajax**

在jQuery中，$.Ajax()方法属于最底层的方法，第2层是load()，$.get()，和$.post()，第3层是$.getScript()和$.getJSON()方法。

**一、 load( ) 方法**

**结构**

load( url , [data] , [callback] )

**参数解释:**

**1.1 应用**

案例：简单的load使用

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8" />

        <title></title>

    </head>

    <body>

        <div id="box">

        </div>

    </body>

</html>

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

<script type="text/javascript">

    $(function(){

        //load方法加载html

        $("#box").load("testFile/test1.txt",function(){

            //异步操作

            console.log($("#box").html());

        });

    });

</script>

test1.txt:

hello world

案例:页面头部重复引用

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style type="text/css">

            \*{

                margin: 0;

                padding: 0;

            }

            header{

                height: 100px;

                background: yellow;

            }

            header ul{

                height: 100px;

                width: 800px;

                position: relative;

                margin: 0 auto;

                opacity: 0.5;

            }

            header ul li{

                list-style: none;

                width:150px;

                background: red;

                text-align: center;

                line-height: 100px;

                border: 1px solid black;

                float: left;

            }

            section{

                height: 300px;

                background: green;

                opacity: 0.3;

            }

            footer{

                height: 300px;

                background: blue;

                opacity: 0.3;

            }

        </style>

    </head>

    <body>

        <header>

        </header>

        <section>

        </section>

        <footer>

        </footer>

    </body>

</html>

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

<script>

    $(function(){

        $("header").load("testFile/2\_header.html",function(){

            $("li").click(function(){

                alert($(this).html());

            });

        });

    });

</script>

2\_header.html内容

<ul>

    <li>主题1</li>

    <li>主题2</li>

    <li>主题3</li>

    <li>主题4</li>

    <li>主题5</li>

</ul>

**1.2筛选载入的HTML文档**

上面的例子是将2\_header.html页面中的内容都加载到header的元素里，如果只要加载页面内的某些内容，可以使用load（URL selector）方法的URL参数来达到目的。

注意：URL和选择器之间有一个空格。

例如只需要加载2\_header.html的ul元素中内容，可以使用如下代码：

    $(function(){

        $("header").load("testFile/3\_header.html ul",function(){

            $("li").click(function(){

                alert($(this).html());

            });

        });

    });

3\_header.html:文件内容

<ul>

    <li>主题1</li>

    <li>主题2</li>

    <li>主题3</li>

    <li>主题4</li>

    <li>主题5</li>

</ul>

<p>123123</p>

**1.3传递方式**

load（）方法传递方式根据参数data来自动指定。如果没有参数传递，则采用get方式，反之，则自动转换为post方式。

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

    $("#cont").load("xi.html li"，{name:"rain"，age"18"}，function（）{

   //code....

  });

})

**1.4 回调参数**

对于必须在加载完成后才能继续的操作，load（）方法提供了回调函数（callback），该函数有3个参数，分别代表“请求返回的内容”，“请求状态”，“XMLHttpRequest对象”，代码如下：

$("header").load("testFile/3\_header.html ul"，function（responseText，textStates，XMLHttpRequest）{

   //responseText：请求返回的内容

   //textStates：请求状态：success   error

   //XMLHttpRequest：XMLHttpRequest对象

  });

注意：在load（）方法中，无论Ajax请求是否成功，只要请求完成。回调函数就被触发。

**二、 $.get( ) 方法和$.post（）方法**

load（）通常是从web服务器上获取静态的数据文件，如果需要专递一些参数给服务器中的页面，可以使用 $.get( ) 方法和$.post（）方法（或$.ajax()方法）

**2.1 $.get( ) 方法**

结构

$.get( url，[ data ]，[ callback ]，[ type ])

参数解释:

**2.1.1 应用**

     $(function(){                               //返回值   状态

            $.get("testFile/4test.php",{name:"老王"},function(res,status){

              console.log(res);

    });

**如果服务器结构到传递的data数据并成功返回，那么就可以通过回调函数将返回的数据显示到页面上**

$.get()方法的回调函数只有两个参数，

function(){

  //data：返回的内容，可以是XML文档，json文件，XHML片段等等

  //textStatus：请求状态：success  error

}

**data参数代表请求返回的内容，textStatus参数代表请求状态，而且回调含税只有当数据成功（success）后才能被调用**

 $.get的返回值:

$(function(){                               //返回值   状态

   $.get("testFile/4test.php",{name:"老王"},function(res,status){

       console.log(res);

       }).then(function(res){

          console.log("成功"+res);

        },function(res){.

          console.log("失败"+res);

        });

  });

**2.2 $.post( ) 方法**

它与$.get( )方法的结构和使用方式都相同，不过之间仍然有一下区别

**2.2.1get与post区别**

1.post的安全性高于get；如果以get方式请求，请求参数会拼接到url后面，安全性性低，以post方式请求，请求参数会包裹在请求体中，安全性更高

2.数量区别：get方式传输的数据量小，规定不能超过2kb，post方式请求数据量大，没有限制。

3.传输速度：get的传输速度高于post

因为使用方法相同，因此只要改变jQuery函数，就可以将程序在GET请求和POST请求之间切换

$(function(){                               //返回值   状态

  $.post("testFile/4test.php",{name:"哈哈"},function(res,status){

     console.log(res);

   });

});

**三、 $.getScript( ) 方法和$.getJson（）方法**

3.1 $.getScript( )方法

有时候，在页面出现时就获取所需的全部JavaScript文件是完全没有必要的，可以在需要的时候直接加载。jQuery提供了 $.getScript( )方法来直接加载js文件，与加载一个HTML片段一样简单方便，并且不需要对JavaScript文件进行处理，JavaScript文件会自动执行。

3.1.1结构

$.getScript( url , fn )

url：请求地址

fn：回调函数

3.2.1应用

新建一个testFile/6randColor.js文件，随机生成颜色的函数，代码如下

    function changeColor(obj){

        let color = "#";

        let str = "0123456789abcdef";

        for(let i=0; i<6; i++){

            color += str[Math.round(Math.random()\*15)];

        }

        //obj.style.background = color;

        obj.css({background:color});

    }

当我们点击按钮时加载js文件，给body添加颜色

jQuery代码如下

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

    </body>

</html>

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

<script>

//     let oBody = document.querySelector("body");

//     changeColor(oBody);

    $(function(){

//         $.getScript("testFile/6randColor.js",function(){

//             changeColor($("body"));

//         });

        $.getScript("testFile/6randColor.js",function(){

            changeColor($("body"));

        });

    });

</script>

**3.2 $.getJson( )方法**

$.getJson( )方法用于加载JSON文件，与$.getScript( )方法的用法相同

$.getJSON( url，回调函数})

当点击加载时，页面上看不到任何效果，虽然函数加载了JSON文件，但是并没有告知JavaScript对返回的数据如何处理，为此，所以在回调函数里我们可以处理返回的数据

通常我们需要遍历我们得到的数据，虽然这里我们可以使用传统的for循环，但是我们也可以通过$.each()，可以用来遍历对象和数组，$.each()函数是以一个数组或者对象为第1个参数，以一个回调函数作为第2个参数，回调函数拥有两个参数，第1个为对象的成员或者数组的下标，第2个位对应变量或内容

$(function(){

  $("button").click(function(){

    $.getJSON( "text.json" , function( data ){

       $.each( data ， function（index，comment）{

         处理数据...

      }

    })

  })

})

**四、 $.Ajax（）方法**

前面用到的$.load()，$.get()，$.post()，$.getScript()，$.getJSON()这些方法，都是基于$.ajax()方法构建的，$.ajax()是jQuery最底层的Ajax实现，因为可以用来代替前面的所有方法。

所以如果除了上面，还需要编写一下复杂的Ajax程序，那么就要用$.ajax()，$.ajax()不仅能实现与$.load()，$.get()，$.post()同样的功能，而且还可以设定**beforeSend（提交前回调函数）、error（请求失败后处理）、success（请求成功后处理）、complete（请求完成后处理）回调函数**，通过设定这些回调函数，可以给用户更多的Ajax提示信息，另外，还有一些参数可以设置Ajax请求的超时时间或者页面的“最后更改”状态。

之前的文章里面有说过$.Ajax（）方法。

这里不再赘述，直接引之前的例子

$.ajax({

**url：**请求地址

**type:**"get | post | put | delete " 默认是get,

**data:**请求参数 {"id":"123","pwd":"123456"},

**//dataType:**请求数据类型"html | text | json | xml | script | jsonp ",

**success:**function(data,dataTextStatus,jqxhr){ },//请求成功时**error:**function(jqxhr,textStatus,error)//请求失败时

})

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

    </head>

    <body>

    </body>

</html>

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

<script type="text/javascript">

    $(function(){

        $.ajax('',{

            url:"testFile/5test.php",

            data:{

                name:"laowang",

                pwd:"123"

            },

            type:'get',//HTTP请求类型

            success:function(data){

                console.log(data);

            },

            error:function(xhr,type,errorThrown){

                console.log("失败data");

            }

        });

    });

</script>

2.jQueryDom(各种API)

祖先：

parent();

parents();

后代:

children();

find();

兄弟:

siblings();

next();

nextAll();

prev();

prevAll();

过滤(筛选):

eq();

操作元素属性：（获取，设置）

.attr();

.attr()的参数设置有几种方式:

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style>

            #box{

                width: 100px;

                height: 100px;

                background: #0000FF;

                border: 10px solid red;

            }

        </style>

    </head>

    <body>

        <div id="box">

        </div>

    </body>

</html>

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

<script>

    $(function(){

        //一个参数为读

        console.log($("#box").attr("id"));

        //两个参数为改或者添加

        $("#box").attr("id","myBox");

        //$("#myBox").attr("name","oBox");

        //添加多个属性

        $("#myBox").attr({

            a:1,

            b:2,

            c:3

        });

        //css

        console.log($("#myBox").css("width"));

        $("#myBox").css({

            width:"450px",

            height:300,

            background:"red"

        });

        //一次读取多个值

        console.log($("#myBox").css(["width","height","background"]));

    });

</script>

宽度: witdh outerWidth innerWidth

    $(function(){

        console.log($("#box").eq(0).width()); //content

        console.log($("#box").eq(0).outerWidth());//content+border+padding

        console.log($("#box").eq(0).innerWidth());//content+padding

    });

偏移:

    $(function(){

        //offset

        console.log($("#box").offset());

        //注意,没有加position也可以偏移

        $("#box").offset({

            left:100,

            top:100

        });

    });

scrollTpo:

        获取scrollTop的值:

        window.onscroll = function(){

            console.log($(window).scrollTop());

        }

返回顶端：

        设置scrollTop的值:

        $("button").click(function(){

                $(window).scrollTop(0);

        });

index();返回当前元素在父元素的下标

    $(function(){

        $("li").click(function(){

            console.log($(this).index());

        });

    });

节点操作：

JQuery中的DOM操作主要对包括：建【新建】、增【添加】、删【删除】、改【修改】、查【查找】【像数据库操作】。下面的DOM操作将围绕上面的DOM树进行学习JQueryDOM操作。

　　一、查--查找DOM节点

　　　　查找节点非常容易，使用选择器就能轻松完成各种查找工作。例：查找元素节点p返回p内的文本内容$("p").text();例：查找元素节点p的属性返回属性名称对应的属性值$("p").attr("title"),返回p的属性title的值。

　　二、建--新建DOM节点

　　　　1、创建元素节点

　　　　　　创建元素节点并且把节点作为<ul>元素的子节点添加到DOM节点树上。先创建元素点，创建元素节点使用Jquery的工厂函数$()来完成，格式如下：$(    ),该方法会根据传入的html字符串返回一个DOM对象，并将DOM对象包装成一个JQuery对象后返回。创建一个元素节点JQuery代码如下：

        //创建元素

        let oLi = $("<li>");

        oLi.html(666);

        //添加元素

        $("ul").append(oLi);

　　　三、增--添加DOM节点

　　　　　　动态新建元素不添加到文档中没有实际意义，将新建的节点插入到文档中有多个方法，如下：append()、appendTo()、prepend()、prependTo()、after()、insertAfter()、before()、insertBefore()。

　　　　1、append()方法

　　　　　　append()方法向匹配的元素内部追加内容，　方法如下：$("target").append(element);例：

           $("ul").append(oLi);

　　　　2、appendTo()方法

　　　　　　appendTo()方法将所有匹配的元素追加到指定的元素中，该方法是append()方法的颠倒[操作主题的颠倒并非操作结果]操作。方法如下：$(element).appendTo(target);例：

          oLi.appendTo($("ul"));

　　　　3、prepend()方法

　　　　　　prepend()方法将要匹配的元素头插，方法如下：$(target).prepend(element);例：

　　　　　　$("ul").prepend(oLi);

　　　　　　该方法将查找元素ul然后将新建的li元素作为ul子节点，且作为ul的第一个子节点插入到ul中。

　　　　4、prependTo()方法

　　　　　　prependTo()方法将元素添加到每一个匹配的元素内部前置，方法如下：$(element).prependTo();例：

　　　　  　oLi.prependTo($("ul"));

　　　　　　该方法将新建的元素li插入到查找到的ul元素中作为ul的第一个子节元素。

　　　　5、after()方法

　　　　　　after()方法向匹配的元素后面添加元素，新添加的元素做为目标元素后的紧邻的兄弟元素。方法如下：$(target).after(element);例:

　　　　　　$("li").eq(2).after(oLi);

　　　　6、insertAfter()方法

　　　　　　insertAfter()方法将新建的元素插入到查找到的目标元素后，做为目标元素的兄弟节点。方法如下：$(element).insertAfter(target);例:

　　　　　　oLi.insertAfter($("li").eq(1));

　　　　7、before()方法

　　　　　　before()方法在每一个匹配的元素之前插入，做为匹配元素的前一个兄弟节点。方法如下:$(target).before(element);例：

　　　　　　$("li").eq(1).before(oLi);

　　　　8、insertBefore()方法

　　　　　　insertBefore()方法将新建元素添加到目标元素前，做为目标元素的前一个兄弟节点，方法如下:$(element).insertBefore(target);例:

　　　　　　oLi.insertBefore($("li").eq(1));

　　　　　　增加元素的方法前四个是添加到元素内部，后四个是添加到元素外部的操作，有这些方法可以完成任何形式的元素添加。

　  　四、删--删除DOM节点操作

　　　　　　如果想要删除文档中的某个元素JQuery提供了两种删除节点的方法：remove()和empty();

　　　　1、remove()方法

　　　　　　remove()方法删除所有匹配的元素，传入的参数用于筛选元素，该方法能删除元素中的所有子节点，当匹配的节点及后代被删除后，该方法返回值是指向被删除节点的引用，因此可以使用该引用，再使用这些被删除的元素。方法如下：$(element).remove();例:

　　　　　　$("li").eq(1).remove();

　　　　　　该示例中先删除所有的span元素，把删除后的元素使用$span接收，把删除后的元素添加到ul后面做为ul的兄弟节点。该操作相当于将所有的span元素以及后代元素移到ul后面。

　　　　2、empty()方法。

　　　　　　empty()方法严格来讲并不是删除元素，该方法只是清空节点，它能清空元素中的所有子节点。方法如下:$(element).empty();例：

　　　　　　$("li").eq(2).empty();

　　　　　　该示例使用empty方法清空ul中第一个li的文本值。只能留下li标签默认符号”·“。

3.懒加载

什么是懒加载:

懒加载其实就是延迟加载，是一种对网页性能优化的方式，比如当访问一个页面的时候，优先显示可视区域的图片而不一次性加载所有图片，当需要显示的时候再发送图片请求，避免打开网页时加载过多资源。

什么时候用懒加载:

当页面中需要一次性载入很多图片的时候，往往都是需要用懒加载的。

懒加载原理:

我们都知道HTML中的<img>标签是代表文档中的一个图像。。说了个废话。。

<img>标签有一个属性是src，用来表示图像的URL，当这个属性的值不为空时，浏览器就会根据这个值发送请求。如果没有src属性，就不会发送请求。

嗯？貌似这点可以利用一下？

我先不设置src，需要的时候再设置？

nice，就是这样。

我们先不给<img>设置src，把图片真正的URL放在另一个属性data-src中，在需要的时候也就是图片进入可视区域的之前，将URL取出放到src中。

需要用到知识点:

(1) 图片距离顶部的高度:

img.offsetTop

(2) 当前窗口的高度:

window.innerHeight

(3) 滚动条滚动的高度:

document.body.scrollTop || document.documentElement.scrollTop

加载图片的判断条件:

图片距离顶部的距离 < 浏览器滚动条滚动的高度 + 浏览器窗口的高度

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <title></title>

        <style type="text/css">

            div{

                position: absolute;

                width: 200px;

            }

        </style>

    </head>

    <body>

    </body>

</html>

<script>

    class WaterFall{

        constructor() {

        }

        createDiv(){

            for(let i=0; i<14; i++){

                let div = document.createElement("div");

                div.innerHTML = i;

                let rnd = Math.round(Math.random()\*300+50);//生成50~350的随机数

                div.style.height = rnd + "px";

                div.date\_src = "url(img/" + i + ".jpg)";

                div.style.backgroundSize = "200px" + " " + rnd + "px";

                document.body.appendChild(div);

            }

            this.change();

        }

        change(){

            let oArrDiv = document.getElementsByTagName("div");

            let n = Math.floor(window.innerWidth/210);

            let arrH = [];

            for(let i=0; i<oArrDiv.length; i++){

                //图片距离顶部的距离 < 浏览器滚动条滚动的高度 + 浏览器窗口的高度

                if(oArrDiv[i].offsetTop < document.documentElement.scrollTop + document.documentElement.clientHeight){

                    this.loadImg(oArrDiv[i]);

                }

                if(arrH.length == n){

                    let index = this.findMin(arrH);

                    oArrDiv[i].style.left = 200\*index + 10\*index + "px";

                    oArrDiv[i].style.top = arrH[index] + 10 + "px";

                    arrH[index] += oArrDiv[i].offsetHeight + 10;

                }else{

                    oArrDiv[i].style.top = "0" + "px";

                    oArrDiv[i].style.left = 200\*i + 10\*i + "px";

                    arrH[i] = oArrDiv[i].offsetHeight;

                }

            }

        }

        findMin(arrH){

            let index = 0;

            for(let i=0; i<arrH.length; i++){

                if(arrH[index]>arrH[i]){

                    index = i;

                }

            }

            return index;

        }

        moveScroll(){

            let top = 0;

            let that = this;

            window.onscroll = function(){

                top = document.body.scrollTop || document.documentElement.scrollTop;

                if(top > 400){

                    that.createDiv();

                }

            }

        }

        loadImg(oDiv){

            if(oDiv.src == null){

                oDiv.style.background = oDiv.date\_src;

            }

        }

    }

    let wf = new WaterFall();

    wf.createDiv();

    wf.createDiv();

    wf.moveScroll();

</script>