翡翠教育 EMERALD EDUCATION

任务1 列表页数据绑定(8 小时)

1.1 预期效果

图 1-1 列表页效果图

1.2 掌握技能

表 1-1 任务说明

任务名称	子任务	任务内容
列表页	1	url 参数获取，需绑定的数据获取
	2	单页面数据绑定
	3	瀑布流加载实现

1.3 任务分解

1. url 参数获取，需绑定的数据获取

(1) 在访问的 url 中添加参数

```
guanwang/list.html?type=xiaoniaoNews
```

在打开的列表页顶部 URL 后面手动添加一个参数，如：以前访问列表页的时候用的地址是 guanwang/list.html，此时我们在网页的 url 后面添加?type=xiaoniaoNews；

说明：网站一般会有多种类型的列表（当前网站只有“小鸟动态列表”，但以后很可能会添加其他列表），而这些列表如果都是文章，那么会共用同样的列表页，只是传递不同的参数，以实现不同的数据加载；

而这些参数的传递，一般通过 url 参数传递，可以 hash 值传递（在 url 后面跟#号，然后加参数）如：list.html#xiaoniaoNews

或者通过请求值传递（在 URL 后面跟？号，然后加参数，每个参数用&隔开）

list.html?type=xiaoniaoNews

第一种方式传递参数不利于分割，第二种参数传递方式更好，可以传多个值，我们这里采用第二种方式。每种类型的列表，使用键值对 参数名=值 的形式传递。

以后在实现页面连通跳转的时候，我们会给头部的导航添加相应的带参数的 url 链接，这样，跳转到列表页的时候，就能自动识别相应的页面。

(2) 在 list.js 末尾编写 url 参数获取方法

```
//获取页面 url 传过来的参数
function getUrlParams(name){
    var reg = new RegExp("(^|&)"+ name +"=([^&]*)(&|$)");
    var r = window.location.search.substr(1).match(reg);
    if(r!=null)
        return  r[2];
    else
        return "";
}
```

Reg 表示匹配出：&+url 参数名字=值+& ，其中&可以不存在。这样会返回一个数组

["type=xiaoniaoNews", "", "xiaoniaoNews", ""]

我们需要的值在第三个，所有返回 r[2];

如果没匹配到，则返回空字符“”;

(3) 调用 getUrlParams(name)方法调试

```
alert(getUrlParams("type"))
```

在$(function(){})中执行上面的测试方法，会弹出相应的值。(调试的时候，需要 URL 中有相应的参数，如 guanwang/list.html?type=xiaoniaoNews)。

测试完成，先注释此调试代码，以后的代码中需要用此代码，再使用。

(4) 需要绑定的数据获取（此处仅为测试查看），在$(function(){})中执行下面代码

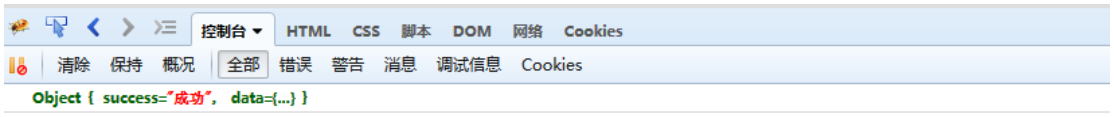
```
console.log(listData["listData01"]);
```

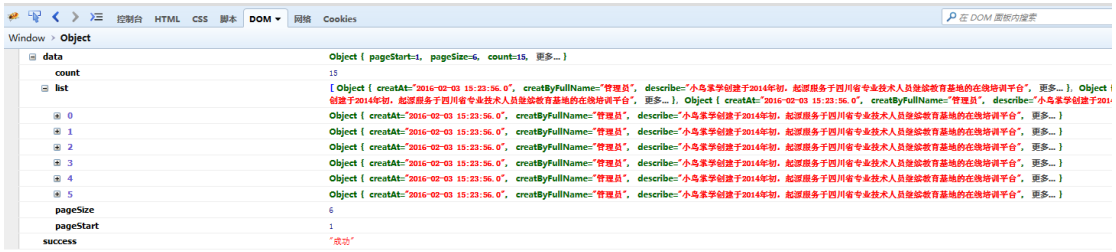
由于我们在 listData.js 中已经存储了列表的数据，这里就可以直接调用了（数据为格式和 ajax 返回的数据一致，这个项目我们先不涉及服务器交互，所以准备好数据用放到 js 中模拟数据绑定。）

如果编写静态页面的时候忘记引入 listData.js，这里请先自行引入。

打开控制台，就能看到打印出来的数据，点击就能查看数据的详细。如下图。

查看到结果之后，我们把此测试代码删除，下一步开始正式的数据绑定。





2. 单页面数据绑定

(1) 加载数据方法调用，在$(function(){})中执行数据加载方法。(具体方法下一步再写)

```
loadArticleList();
```

(2) 申明一个全局变量，在 list.js 顶部编写。

```
var GLOBAL = GLOBAL || {};
```

GLOBAL 用于储存一些需要全局储存的信息。

(3) loadArticleList()方法，在 list.js 最下边编写。

```
function loadArticleList(){
    if(!GLOBAL.pageStart){
        $("#articleList").html("");
        GLOBAL.pageStart = 0;
    }
```

```
    //此方法以后的代码在此接着编写
}
```

如果第一次加载，我们没有给 GLOBAL.pageStart 赋值，这时，我们需要赋值 GLOBAL.pageStart = 0；同时，清除$("#articleList")的内容。

(4) 获取数据

```
var itemHtml = '';
var result = listData["listData0" +GLOBAL.pageStart]; //此数据在 listData.js 里
var list = result.data.list;
```

本来应该是获取了 url 参数，然后 ajax 请求数据的，但，这里，我们将数据储存在了 listData.js 中，所以，直接获取就可以了。

listData["listData0" +GLOBAL.pageStart]是根据相应的 pageStart 值，取 listData.js 里存储的数据。大家可以去 listData.js 里查看全部的 json 数据。

这里申明了变量 itemHtml 用于储存后续拼装的的需要放到 html 中的列表代码。

把列表的数据存到变量 list 中，大家可以用 console.log（）方法打印出来看看结构。

(5) 在 list.html 中编写 html 模板。

```
<div id="itemHtml" style="display:none;">
    <div class="content_one" articleid="$articleId$">
        <div class="img_wrap"><img src="$articleCover$"></div>
        <div class="content_text">
            <div class="title_small">$articleTitle$</div>
            <div class="date">$updateTime$</div>
            <p>$describe$</p>
        </div>
        <img class="list_img_over_xiaojiantou"
src="images/list_img_over_xiaojiantou.png">
    </div>
</div>
```

在 list.html 中，复制一份. content_one。放到页面 body 中任意位置，外面包裹一

层 div，设置 id 为 itemHtml，设置 style 隐藏。对模板内部需要替换值的地方，取相应的名字。

(6) 在上上步的 js 代码中接着编写数据处理方法

```
if(!list || !list.length){
    $("#articleList").html("暂时没有内容，敬请期待！");
} else {
    var updateTime;
    for(var i=0; i < list.length; i++){
        updateTime = list[i].updateAt || list[i].creatAt;
        itemHtml = $("#itemHtml").html().replace("$articleCover$",
list[i].coverImg)
                .replace("$articleId$", list[i].sysId)
                .replace("$articleTitle$", list[i].title)
                .replace("$updateTime$", updateTime ? updateTime.substr(0,
10) : "")
                .replace("$describe$", list[i].describe);
        $("#articleList").append(itemHtml);
    }
};
```

If 是判断返回的数据中有没有数据，如果没有数据，就显示"暂时没有内容，敬请期待！";

如果有数据，就执行 else 中的内容，

先申明一个变量 var updateTime;（文章会有发表日期和编辑更新日期，此处声明用于储存需要显示的日期）

接着，对数据就行循环加载：for 循环。

循环内部 updateTime = list[i].updateAt || list[i].creatAt;为日期赋值。

获取$("#itemHtml")的 html，就是我们上一步在 list.html 中编写的模板。

对它执行中间相应值的替换。替换后的 html 就是我们需要显示的 DOM 了，直接 append 到$("#articleList")中即可显示。

3. 瀑布流加载实现

(1) loadArticleList()方法内部末尾编写下一页的 pageStart 定位

```
GLOBAL.pageStart = result.data.pageStart + 1;
```

```
GLOBAL.pageCount = Math.ceil(result.data.count/result.data.pageSize);
if(GLOBAL.pageStart >= GLOBAL.pageCount){
    $("#listMore").css("opacity","0").prev("img").attr("src","images/lis
t_gomore_bg_nomore.jpg");
};
```

1、在返回数据的 pageStart 基础之上+1，赋值给 GLOBAL.pageStart

2、根据返回的 count 和 pageSize 计算工有多少页，赋值给 GLOBAL.pageCount

3、判断，如果已经是最后一页了，就隐藏加载按钮，并且让提示下一页的图片换成"没有下更多了~~~"

(2) 点击下一页加载的代码，在$(function(){})内部最后编写。

```
$("#listMore").click(function(){

        if(GLOBAL.pageStart < GLOBAL.pageCount){
            loadArticleList();
        }

})
```

当点击下一页的时候，判断，如果不是最后一页，就执行数据加载方法。

(3) 点击每个文章链接，跳转到文章页。

```
$("#articleList").delegate(".content_one", "click", function(){
    window.open("article.html?"+"type=" + getUrlParams("type")+
"&articleId=" + $(this).attr("articleId"), "_blank");
});
```

由于.content_one 是后来添加到页面的 dom，直接绑定 click 事件是不起作用的，这里我们采用事件委托的方式，把事件委托到$("#articleList")上。

jquery 的 delegate 介绍：

$(selector).delegate(childSelector,event,data,function);

childSelector 必需。规定要附加事件处理程序的一个或多个子元素。

event 必需。规定附加到元素的一个或多个事件。由空格分隔多个事件值。必须是有效的事件。

data 可选。规定传递到函数的额外数据。

function 必需。规定当事件发生时运行的函数。

当点击的时候，我么执行 window.open 方法，打开 article.html 页面，传递两个参数“type”和“articleId”，以便后面的详情页面接收参数显示对于的数据。

（文章页面的数据绑定将在下一个任务书里编写）

列表数据加载的 js， 编写完毕！

1.4 参考资料

无

1.5 扩展练习

教师自行补充