# 服务器和Ajax

第4天课堂笔记（本课程共4天）

目录

[服务器和Ajax 1](#_Toc476949332)

[一、模板引擎 3](#_Toc476949333)

[1.1 模板引擎原理 3](#_Toc476949334)

[1.2 underscore.js 4](#_Toc476949335)

[二、模板修正 11](#_Toc476949336)

# 一、模板引擎

## 1.1 模板引擎原理

拼接字符串很不爽，还容易出错。

所以就有工程师在大量的实践中，提出了模板引擎的概念，就是在一个完整的字符串中，把未定的量用特殊的语法比如：

|  |
| --- |
| 1. @xinqing@ |

来表示。然后把数据替换这些标记，这个操作叫做数据绑定。

|  |
| --- |
| 1. //模板 2. var str = "好**@xinqing@**啊，我买了一个**@dongxi@**，花了**@qian@**元钱"; 3. //字典，数据 4. var dictionary = { 5. "xinqing" : "高兴", 6. "dongxi" : "vivo手机", 7. "qian" : 7000 8. } 9. //数据绑定 10. str = str.replace(**/\@([a-z]+)\@/g** , function(match,$1,index,string){ 11. return dictionary[$1]; 12. }); 13. console.log(str); |

模板一般都放在页面的

|  |
| --- |
| 1. <script type=”text/template”><script> |

标签对儿里面。

字典都是从Ajax读来的。

|  |
| --- |
| 1. function compile(templateString , dictionary){ 2. return templateString.replace(/\@([a-z]+)\@/g , function(match,$1,index,string){ 3. return dictionary[$1]; 4. }); 5. } |

## 1.2 underscore.js

手册： <http://www.css88.com/doc/underscore/#reduce>

先回忆一下jQuery是干嘛的？jQuery解决了DOM开发中的问题，比如选择元素难、事件绑定不能批量、animate难的问题、浏览器兼容问题太多。jQuery是DOM编程界的霸主。jQuery离开DOM开发就是渣渣，比如用jQuery你不能开发canvas游戏，不能开发node.js。

underscore不限制使用场景，我们canvas游戏真得靠它，node.js里面都能够用它。underscore封装了一堆实用函数，这些实用函数基本都是针对数组、对象、函数的。这些小功能，都不复杂，但是我们自己写挺麻烦的。所以就是屌人集合在了一起，叫做underscore。

熟练知道：

|  |
| --- |
| 1. \_.max(); 2. \_.max(list, [iteratee], [context])  返回**list**中的最大值。如果传递**iteratee**参数，**iteratee**将作为**list**中每个值的排序依据。如果**list**为空，将返回*-Infinity*，所以你可能需要事先用[isEmpty](http://www.css88.com/doc/underscore/#isEmpty)检查 **list** 。 3. \_.min(); 4. \_.min(list, [iteratee], [context])  返回**list**中的最小值。如果传递**iteratee**参数，**iteratee**将作为**list**中每个值的排序依据。如果**list**为空，将返回*-Infinity*，所以你可能需要事先用[isEmpty](http://www.css88.com/doc/underscore/#isEmpty)检查 **list** 。 5. \_.without(); 6. 返回一个删除所有**values**值后的 **array**副本。（愚人码头注：使用*===*表达式做相等测试。） |
|  |

underscore中内置了一个模板引擎：

|  |
| --- |
| 1. //模板字符串 2. var templateString = "好<%= xinqing %>啊，今天我买了<%= dongxi%>，花了<%=qian%>元"; 3. //通过模板字符串生成一个数据绑定函数 4. var compile = \_.template(templateString); 5. //字典 6. var dictionary = { 7. "xinqing" : "高兴", 8. "dongxi" : "vivo手机", 9. "qian" : 7000 10. } 11. //调用数据绑定函数完成数据绑定，只需要传一个参数，就是字典。 12. console.log( compile(dictionary) ); |

大量公司在把underscore当做事实上的标准。

这是一个js库，就是一个js小包，说白了就是一个js文件。



[优雅的underscore](http://yalishizhude.github.io/2015/08/02/beatiful-underscore/)

暴露的唯一的顶层变量就是\_。 underscore就是下划线的意思。**处理函数、数组、对象等等的，不涉及DOM、动画、Ajax，他是一个实用库。开发任何东西，比如canvas游戏、dom效果、nodejs都可以用underscore**。

函数举例：

|  |
| --- |
| // shuffle函数，返回一个随机乱序的 list 副本  var arr = [1,2,3,4,5,6,7,8,9];  var arr2 = **\_.shuffle(**arr**)**;  console.log(arr2); |

|  |
| --- |
| // sample函数，从 list中产生一个随机样本，第二个参数6表示产生6个  var arr = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];  var arr2 = \_.sample(arr,6);  console.log(arr2); |

我们自己也可以写产生随机样本的函数（当然很难）

|  |
| --- |
| var arr = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];  var result = [];  for(var i = 0 ; i < 6 ; i++){  while(true){  // 随机一个下标  var r = parseInt(Math.random() \* 20);  // 验证你产生的这个下标对应的arr里面的数字，在不在result里面  if (result.indexOf(arr[r]) == -1) {  break;  }  }  result.push(arr[r]);  }  console.log(result); |

each方法，迭代器：

|  |
| --- |
| var arr = ["东风","幺鸡","八万","一饼"];  each(arr,function(item){  console.log(item);  }); |

我们自己也可以写一个each：

|  |
| --- |
| var arr = ["东风","幺鸡","八万","一饼"];  function each(array , fn){  for(var i = 0 ; i < array.length ; i++){  fn(array[i]);  }  }  each(arr,function(a){  console.log(a);  }); |

filter方法：

|  |
| --- |
| var arr = [  {"xingming":"小明","age":12,"sex":"男"},  {"xingming":"小红","age":18,"sex":"女"},  {"xingming":"小绿","age":2,"sex":"女"},  {"xingming":"小刚","age":2,"sex":"男"},  {"xingming":"小灰","age":12,"sex":"男"}  ];  var result = \_.filter(arr,function(item){  return item.age > 10;  });  console.log(result); |

contains方法：返回true||false

|  |
| --- |
| // 检测数组中是否包含某项  var arr = ["东风","幺鸡","八万","一饼"];  var result = \_.contains(arr,"八万");  alert(result); |

最大值、最小值：

|  |
| --- |
| var arr = [45,232,45,324];  alert(\_.max(arr));  alert(\_.min(arr)); |

特别的，可以返回一组对象数组中，按年龄的最小值：

|  |
| --- |
| var arr = [  {"xingming":"小明","age":12,"sex":"男"},  {"xingming":"小红","age":18,"sex":"女"},  {"xingming":"小绿","age":2,"sex":"女"},  {"xingming":"小刚","age":2,"sex":"男"},  {"xingming":"小灰","age":12,"sex":"男"}  ];  var result = \_.min(arr,function(item){  return item.age;  });  console.log(result); |

排序：

|  |
| --- |
| var arr = [  {"xingming":"小明","age":12,"sex":"男"},  {"xingming":"小红","age":18,"sex":"女"},  {"xingming":"小绿","age":2,"sex":"女"},  {"xingming":"小刚","age":2,"sex":"男"},  {"xingming":"小灰","age":12,"sex":"男"}  ];  // 按年龄排序：  var result = \_.sortBy(arr,function(item){  return item.age;  });  console.log(result); |

统计：

|  |
| --- |
| var arr = [  {"xingming":"小明","age":22,"sex":"男"},  {"xingming":"小红","age":18,"sex":"女"},  {"xingming":"小绿","age":2,"sex":"女"},  {"xingming":"小刚","age":2,"sex":"男"},  {"xingming":"小灰","age":12,"sex":"男"}  ];  var result = \_.countBy(arr,function(item){  if(item.age >= 18){  return "成年人";  }else{  return "未成年人";  }  });  console.log(result); |



|  |
| --- |
| // 返回对象键的个数  var obj = {"a":undefined,"b":undefined,"c":undefined};  alert(\_.size(obj)); |

|  |
| --- |
| // 删除指定项  var arr = ["东风","幺鸡","八万","一饼"];  arr = \_.without(arr,"幺鸡"); |

|  |
| --- |
| // 合并数组并且去重  var arr = \_.union([1,2,3],[2,3,4],[4,5,6]);  console.log(arr); |

|  |
| --- |
| // 寻找指定项  var arr = ["东风","幺鸡","八万","一饼"];  alert(\_.indexOf(arr,"八万")); |

克隆对象：创建 一个浅复制（浅拷贝）的克隆object。任何嵌套的对象或数组都通过引用拷贝，不会复制。

|  |
| --- |
| \_.clone({name: 'moe'});  => {name: 'moe'}; |

|  |
| --- |
| // 深度比较  var arr1 = ["东风","幺鸡","八万","一饼"];  var arr2 = ["东风","幺鸡","八万","一饼"];  alert(arr1 == arr2);  alert(\_.isEqual(arr1,arr2)); |

\_.template()函数：

|  |
| --- |
| //① 模板字符串  var str = "我是<%=name%>,我<%=nianling%>岁了,我是<%=xingbie%>。";  //② 得到一个函数，模板编译函数  var compiled = \_.template(str);  //③ 运行这个函数，参数是字典，返回值是数据绑定之后的字符串  var result = compiled({  "name" : "小明",  "nianling" : 12,  "xingbie" : "男"  });  alert(result); |

# 二、模板修正



变为：

|  |
| --- |
| <p class="biaoqian">  <a href="#">标签1</a>  <a href="#">标签2</a>  <a href="#">标签3</a>  </p> |

我们可以认为制造一个模板字符串：

|  |
| --- |
| <p class="biaoqian">  <%=biaoqian%>  </p> |

此时dictionary中没有biaoqian，进行修正。

|  |
| --- |
| // 模板修正，我们现在人为的增添一个字典项：  dictionary.biaoqian = "";  // 遍历这本字典的标签：  \_.each(dictionary.m\_label\_names,function(item){  dictionary.biaoqian += "<a href='http://baijia.baidu.com/?tn=listarticle&labelid=" + item.m\_id + "'>" + item.m\_name +"</a>"  }); |

先去自定义一个模板标记占位，然后在dictionary补充这个词。

|  |
| --- |
| <span class="zuozhe">  <%=m\_writer\_name%>  <%=v%>  </span> |

修正模板：

|  |
| --- |
| dictionary.v = dictionary.m\_attr\_first\_pub == 1 ? "<img src='v\_e077275.png' class='vicon'/>" : ""; |