Javascript模块化编程（三）：require.js的用法

http://www.ruanyifeng.com/blog/2012/11/require\_js.html

这个系列的[第一部分](http://www.ruanyifeng.com/blog/2012/10/javascript_module.html)和[第二部分](http://www.ruanyifeng.com/blog/2012/10/asynchronous_module_definition.html)，介绍了Javascript模块原型和理论概念，今天介绍如何将它们用于实战。

我采用的是一个非常流行的库[require.js](http://requirejs.org/)。



**一、为什么要用require.js？**

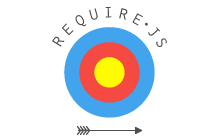
最早的时候，所有Javascript代码都写在一个文件里面，只要加载这一个文件就够了。后来，代码越来越多，一个文件不够了，必须分成多个文件，依次加载。下面的网页代码，相信很多人都见过。

　　<script src="1.js"></script>  
　　<script src="2.js"></script>  
　　<script src="3.js"></script>  
　　<script src="4.js"></script>  
　　<script src="5.js"></script>  
　　<script src="6.js"></script>

这段代码依次加载多个js文件。

这样的写法有很大的缺点。首先，加载的时候，浏览器会停止网页渲染，加载文件越多，网页失去响应的时间就会越长；其次，由于js文件之间存在依赖关系，因此必须严格保证加载顺序（比如上例的1.js要在2.js的前面），依赖性最大的模块一定要放到最后加载，当依赖关系很复杂的时候，代码的编写和维护都会变得困难。

require.js的诞生，就是为了解决这两个问题：



　　（1）实现js文件的异步加载，避免网页失去响应；

　　（2）管理模块之间的依赖性，便于代码的编写和维护。

**二、require.js的加载**

使用require.js的第一步，是先去官方网站[下载](http://requirejs.org/docs/download.html)最新版本。

下载后，假定把它放在js子目录下面，就可以加载了。

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

有人可能会想到，加载这个文件，也可能造成网页失去响应。解决办法有两个，一个是把它放在网页底部加载，另一个是写成下面这样：

　　<script src="js/require.js" **defer async="true"** ></script>

async属性表明这个文件需要异步加载，避免网页失去响应。IE不支持这个属性，只支持defer，所以把defer也写上。

加载require.js以后，下一步就要加载我们自己的代码了。假定我们自己的代码文件是main.js，也放在js目录下面。那么，只需要写成下面这样就行了：

　　<script src="js/require.js" **data-main="js/main"**></script>

data-main属性的作用是，指定网页程序的主模块。在上例中，就是js目录下面的main.js，这个文件会第一个被require.js加载。由于require.js默认的文件后缀名是js，所以可以把main.js简写成main。

**三、主模块的写法**

上一节的main.js，我把它称为"主模块"，意思是整个网页的入口代码。它有点像C语言的main()函数，所有代码都从这儿开始运行。

下面就来看，怎么写main.js。

如果我们的代码不依赖任何其他模块，那么可以直接写入javascript代码。

　　// main.js

　　alert("加载成功！");

但这样的话，就没必要使用require.js了。真正常见的情况是，主模块依赖于其他模块，这时就要使用AMD规范定义的的require()函数。

　　// main.js

　　require(['moduleA', 'moduleB', 'moduleC'], function (moduleA, moduleB, moduleC){

　　　　// some code here

　　});

require()函数接受两个参数。第一个参数是一个数组，表示所依赖的模块，上例就是['moduleA', 'moduleB', 'moduleC']，即主模块依赖这三个模块；第二个参数是一个回调函数，当前面指定的模块都加载成功后，它将被调用。加载的模块会以参数形式传入该函数，从而在回调函数内部就可以使用这些模块。

require()异步加载moduleA，moduleB和moduleC，浏览器不会失去响应；它指定的回调函数，只有前面的模块都加载成功后，才会运行，解决了依赖性的问题。

下面，我们看一个实际的例子。

假定主模块依赖jquery、underscore和backbone这三个模块，main.js就可以这样写：

　　require(['jquery', 'underscore', 'backbone'], function ($, \_, Backbone){

　　　　// some code here

　　});

require.js会先加载jQuery、underscore和backbone，然后再运行回调函数。主模块的代码就写在回调函数中。

**四、模块的加载**

上一节最后的示例中，主模块的依赖模块是['jquery', 'underscore', 'backbone']。默认情况下，require.js假定这三个模块与main.js在同一个目录，文件名分别为jquery.js，underscore.js和backbone.js，然后自动加载。

使用require.config()方法，我们可以对模块的加载行为进行自定义。require.config()就写在主模块（main.js）的头部。参数就是一个对象，这个对象的paths属性指定各个模块的加载路径。

　　require.config({

　　　　paths: {

　　　　　　"jquery": "jquery.min",  
　　　　　　"underscore": "underscore.min",  
　　　　　　"backbone": "backbone.min"

　　　　}

　　});

上面的代码给出了三个模块的文件名，路径默认与main.js在同一个目录（js子目录）。如果这些模块在其他目录，比如js/lib目录，则有两种写法。一种是逐一指定路径。

　　require.config({

　　　　paths: {

　　　　　　"jquery": "**lib/**jquery.min",  
　　　　　　"underscore": "**lib/**underscore.min",  
　　　　　　"backbone": "**lib/**backbone.min"

　　　　}

　　});

另一种则是直接改变基目录（baseUrl）。

　　require.config({

**baseUrl: "js/lib",**

　　　　paths: {

　　　　　　"jquery": "jquery.min",  
　　　　　　"underscore": "underscore.min",  
　　　　　　"backbone": "backbone.min"

　　　　}

　　});

如果某个模块在另一台主机上，也可以直接指定它的网址，比如：

　　require.config({

　　　　paths: {

　　　　　　"jquery": "https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min"

　　　　}

　　});

require.js要求，每个模块是一个单独的js文件。这样的话，如果加载多个模块，就会发出多次HTTP请求，会影响网页的加载速度。因此，require.js提供了一个[优化工具](http://requirejs.org/docs/optimization.html)，当模块部署完毕以后，可以用这个工具将多个模块合并在一个文件中，减少HTTP请求数。

**五、AMD模块的写法**

require.js加载的模块，采用AMD规范。也就是说，模块必须按照AMD的规定来写。

具体来说，就是模块必须采用特定的define()函数来定义。如果一个模块不依赖其他模块，那么可以直接定义在define()函数之中。

假定现在有一个math.js文件，它定义了一个math模块。那么，math.js就要这样写：

　　// math.js

　　define(function (){

　　　　var add = function (x,y){

　　　　　　return x+y;

　　　　};

　　　　return {

　　　　　　add: add  
　　　　};

　　});

加载方法如下：

　　// main.js

　　require(['math'], function (math){

　　　　alert(math.add(1,1));

　　});

如果这个模块还依赖其他模块，那么define()函数的第一个参数，必须是一个数组，指明该模块的依赖性。

　　define(['myLib'], function(myLib){

　　　　function foo(){

　　　　　　myLib.doSomething();

　　　　}

　　　　return {

　　　　　　foo : foo

　　　　};

　　});

当require()函数加载上面这个模块的时候，就会先加载myLib.js文件。

**六、加载非规范的模块**

理论上，require.js加载的模块，必须是按照AMD规范、用define()函数定义的模块。但是实际上，虽然已经有一部分流行的函数库（比如jQuery）符合AMD规范，更多的库并不符合。那么，require.js是否能够加载非规范的模块呢？

回答是可以的。

这样的模块在用require()加载之前，要先用require.config()方法，定义它们的一些特征。

举例来说，underscore和backbone这两个库，都没有采用AMD规范编写。如果要加载它们的话，必须先定义它们的特征。

　　require.config({

　　　　shim: {  
  
　　　　　　'underscore':{  
　　　　　　　　exports: '\_'  
　　　　　　},

　　　　　　'backbone': {  
　　　　　　　　deps: ['underscore', 'jquery'],  
　　　　　　　　exports: 'Backbone'  
　　　　　　}

　　　　}

　　});

require.config()接受一个配置对象，这个对象除了有前面说过的paths属性之外，还有一个shim属性，专门用来配置不兼容的模块。具体来说，每个模块要定义（1）exports值（输出的变量名），表明这个模块外部调用时的名称；（2）deps数组，表明该模块的依赖性。

比如，jQuery的插件可以这样定义：

　　shim: {

　　　　'jquery.scroll': {

　　　　　　deps: ['jquery'],

　　　　　　exports: 'jQuery.fn.scroll'

　　　　}

　　}

**七、require.js插件**

require.js还提供一系列[插件](https://github.com/jrburke/requirejs/wiki/Plugins)，实现一些特定的功能。

domready插件，可以让回调函数在页面DOM结构加载完成后再运行。

　　require(['domready!'], function (doc){

　　　　// called once the DOM is ready

　　});

text和image插件，则是允许require.js加载文本和图片文件。

　　define([

　　　　'text!review.txt',

　　　　'image!cat.jpg'

　　　　],  
  
　　　　function(review,cat){

　　　　　　console.log(review);

　　　　　　document.body.appendChild(cat);

　　　　}

　　);

类似的插件还有json和mdown，用于加载json文件和markdown文件。

（完）