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，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，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);

        }

);
```