爱创课堂前端培训

# 模块化开发

第2天课堂笔记（本课程共2天）

班级：北京前端训练营19期

讲师：彭帅伟

日期：2018年12月27日

彭帅伟老师

QQ: 284337908

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

# 复习：

如果想要使用seajs需要在页面去引入

引入之后会向全局暴露两个变量: seajs、 define

在模块外部想要引入其它模块需要通过use方法

seajs.use方法接受两个参数

第一个参数是模块的依赖集合是一个数组，数组中的每一项都是一个模块的文件地址（如果引入的只有一项，可以省略数组）

第二个参数是回调函数， 函数中的参数与前面模块向外暴露的功能是一一对应的

根目录：

seajs引入的文件路径是以seajs所在的文件目录

seajs对js文件比较敏感，所以可以省略.js后缀

定义模块：

提供了六种定义模块的方式

可以传递一个参数:

可以是值类型（数字，字符串，布尔值）

这些数据作为接口直接暴露

可以是引用类型 （arr, obj）

这些数据也是作为接口向外暴露

可以传递一个函数，此时函数遵循commonjs规范

有三个参数

require 用来引入其它模块的

exports: 向外暴露功能的对象

module: 模块信息

可以传递两个参数：

第一个参数可以是字符串表示模块的id

第一个参数还可以是数组， 表示模块的依赖集合

第二个参数是回调函数（与第三种定义模块方式中的函数表现形式一致）

可以传递三个参数：

第一个参数是字符串，表示模块的id

第二个参数是数组，表示模块的依赖集合

第三个参数是回调函数（与第三种定义模块方式中的函数表现形式一致）

require模块：

1 require不能被简写

2 require不能被修改

1 require不能被赋值

2 require不能赋值给其它变量

3 require不能在子函数中作为参数传递

4 require在子函数中不能被修改（以上3点）

3 require不能拼接

接口定义：

exports.接口

module.exports.接口

module.exports = 值类型

module.exports = {}

module.exports = 函数

return 值类型

return {}

return 函数

define(值类型)

define({})

模块信息：

id: 表示模块的id

uri: 表示模块的文件地址

exports: 向外暴露功能的对象

# 异步加载模块

在浏览器端使用一个文件，需要到服务器端加载该文件，这个过程是异步的，seajs为了实现这样的功能，在模块的内部提供了一个异步加载模块文件的方法叫做require.async

require.async方法可以实现异步加载其它模块文件

require.async方法接受两个参数

第一个参数是加载模块的集合

如果是字符串， 表示一个模块文件

如果是一个数组，表示模块的依赖集合，集合中的每一项都是一个模块文件地址

第二个参数是回调函数

回调函数中的参数是前面模块向外暴露的功能（一一对应的）

作用域是全局作用域

require.async方法只能在模块内部被访问，在模块外部不能使用

require.async与seajs.use方法是一样的，只不过seajs建议我们

在模块外部要使用seajs.use方法

在模块内部要使用require.async

require.async与seajs.use方法一样不能加载具有id的模块

但是我们通过模块的依赖集合使require.async方法成功的加载具有id的模块

|  |
| --- |
| 1. define(["module/dom"], function(require, exports, module) { 2. // 异步加载 3. var dom = require.async("myId", function(dom) { 4. console.log(dom); 5. }); 6. }) |

在条件语句中，执行require以及require.async方法

对于require

在if执行块中

加载文件并执行文件

在if不执行块中

加载了文件， 但是没有执行文件

对于require.async

在if执行块中

加载文件并执行文件

在if不执行块中

没有加载和执行文件

|  |
| --- |
| 1. // 条件语句 2. if (true) { 3. // 这里会执行 4. // var dom = require("module/dom"); 5. // console.log(dom); 6. // async 7. require.async("module/dom", function(dom) { 8. console.log(dom); 9. }) 10. } else { 11. // 这里不会执行 12. // var color = require("module/color"); 13. // console.log(color); 14. // async 15. require.async("module/color", function(color) { 16. console.log(color); 17. }) |

在异步语句中，执行require以及require.async

对于require

立即加载模块文件， 但是会在异步语句执行之后执行require加载

对于async

没有立即加载模块文件， 但是回来异步语句执行之后加载文件并执行文件

|  |
| --- |
| 1. // 异步操作 2. setTimeout(function() { 3. // require 4. // var dom = require("module/dom"); 5. // console.log(dom); 6. // async 7. require.async("module/color", function(color) { 8. console.log(color); 9. }) 10. }, 2000) |

# 配置

seajs对象提供了config方法，用来设置配置信息的

配置：本身这个框架就具有的功能， 只是隐藏了， 需要开启，就要进行配置

插件：本身这个框架没有这个功能，只是别人实现了，如果想用使用需要引入文件，才能使用里面的各种功能

## 2.1 alias

作用：简化文件引用的

例如：文件名称过长， 我们可以简化它

值是一个对象

key: 新的文件名称

value: 原始文件路径

举例：

|  |
| --- |
| 1. // 配置alias 2. seajs.config({ 3. alias: { 4. "jquery": "module/js/jquery-1.7.2" 5. } 6. }) 7. // 引入jquery 8. seajs.use("jquery", function(jq) { 9. console.log(jq); 10. }) |

结果：

|  |
| --- |
|  |

## 2.2 paths

作用: 简化路径的

例如：文件的路径名称过长， 我们可以简化它

值是一个对象

key: 新的路径名

value: 原始路径

举例：

|  |
| --- |
| 1. // 使用seajs.config方法进行配置 2. seajs.config({ 3. // 配置paths 4. paths: { 5. "module": "module/js" 6. } 7. }) 8. // 引入jquery 9. seajs.use("module/jquery-1.7.2", function(jq) { 10. console.log(jq); 11. }) |

## 2.3 map

作用: 批量处理文件

值是一个二维数组

数组中的每一项都是一个数组

数组的第一项表示匹配到的文件

数组中的第二项表示要修改的文件

比如压缩之后的文件.min.js， 在我们引入文件的时候，不想书写.min.js， 这个时候要通过map进行配置， 配置哪些文件可以带.min.js, 哪些文件可以不需要带上.min.js

举例：

|  |
| --- |
| 1. // 配置 2. seajs.config({ 3. // 配置map 4. map: [ 5. ["jquery.js", "jquery.min.js"] 6. // [".js", ".min.js"] 7. ] 8. }) |

结果：

|  |
| --- |
|  |

## 2.4 vars

作用: 解决字符串不能拼接的问题

值是一个对象

key: 模板变量名称

value: 模板变量名称对应的值

举例：

main.js：

|  |
| --- |
| 1. // 定义模块 2. define(function(require, exports, module) { 3. // var jq = require("module/js/j" + "jquery" + "-1.7.2"); 4. var jq = require("module/js/{j}-1.7.2"); 5. console.log(jq); 6. }) |

执行代码：

|  |
| --- |
| 1. // 配置 2. seajs.config({ 3. // 配置vars 4. vars: { 5. "j": "jquery" 6. } 7. }) |

结果：

|  |
| --- |
|  |

## 2.5 base

作用: 改变根目录

举例：

|  |
| --- |
| 1. // 配置 2. seajs.config({ 3. // 配置base 4. base: "module/js" 5. }) 6. // 引入入口文件 7. seajs.use("main", function() {}); |

结果：

|  |
| --- |
|  |

## 2.6 加载css插件

seajs-css.js文件

使用方式非常简单

第一步要在seajs文件之后引入seajs-css.js文件

第二步在模块中直接引入css文件即可

.css拓展名是不能省略的，因为seajs对js文件敏感， 但是对css文件不敏感

举例：

|  |
| --- |
| 1. <script type="text/javascript" src="sea.js"></script> 2. <!-- 第一步在seajs文件之后引入 --> 3. <script type="text/javascript" src="seajs-css.js"></script> 4. <script type="text/javascript"> 5. // 第二步直接引入css文件 6. seajs.use("css"); 7. </script> |

## 2.7 预加载文件

seajs-preload.js文件

使用也是非常简单

第一步在seajs文件之后引入preload.js文件

第二步要进行配置， 定义preload配置

值是一个数组， 数组中的每一项都是要预加载的文件

我们就可以使用这些预加载的文件了

加载预加载的文件的时间点

在seajs.use方法之后，模块执行之前

所以，preload预加载的文件尽量放在最前面去配置

举例：

|  |
| --- |
| 1. <script type="text/javascript" src="sea.js"></script> 2. <!-- 第一步在seajs之后引入preload.js文件 --> 3. <script type="text/javascript" src="seajs-preload.js"></script> 4. <script type="text/javascript"> 5. // 第二步要进行perload配置 6. seajs.config({ 7. preload: ["jquery-1.7.2"] 8. }) 9. seajs.use("main", function(main) { 10. console.log(main); 11. console.log($); 12. }) 13. </script> |

## 下午复习：

异步加载模块：

在seajs中提供了一个require.async方法可以实现在模块内部异步的去加载其它模块

接收两个参数

第一个 参数是加载的模块依赖集合

如果是一个字符串， 表示一个模块文件

如果是一个数组，表示多个模块文件

第二个参数是回调函数

函数中的参数就是前面模块向外暴露的功能（一一对应的）

作用域是全局作用域

require.async与seajs.use方法一样

在模块的外部要使用seajs.use方法

在模块内部要使用require.async方法

在条件语句中执行require 以及require.async

对于require

在if执行块中

加载文件并执行文件

在if不执行块中

加载文件，但是不执行

对于async

在if执行块中

加载并执行文件

在if不执行块中

没有加载， 也没有执行文件

在异步操作中，执行require 以及require.async

require

立即加载了文件，但是会在异步语句执行之后执行require加载

async

没有立即加载，会在异步语句执行之后加载模块文件

配置：

本身这个框架就有的功能， 只是隐藏了，需要开启， 所以要进行配置

插件：

本身这个框架没有该这些功能，只是别人实现了， 如果想要使用，需要引入这些文件

alias:

作用： 用于简化文件的引用

值是一个对象

key: 新的文件名称

value: 原始文件名称

paths:

作用: 用于简化文件路径

值是一个对象

key: 新的路径名

value: 原始路径

map:

用于批量处理文件

值是一个二维数组

数组中的每一项都是一个数组

数组中的第一项是匹配到的文件

数组中的第二项要修改的文件

vars:

解决字符串不能拼接的问题

值是一个对象

key: 模板变量名称

value: 模板变量名称对应的值

base:

改变根目录的

css插件：

使用方式非常简单

第一步在seajs文件之后引入seajs.css.js文件

第二步直接在模块内，引入css文件即可

需要注意的是css文件中的.css拓展名不能省略

预加载文件：

seajs-preload.js

使用方式也是非常简单

第一步在seajs之后引入文件

第二步要进行配置，定义preload配置

值是一个数组， 数组中的每一项都是要预加载的文件

加载预加载文件的时间点

在seajs.use方法之后， 模块执行之前

# requirejs

## 3.1 体验requirejs

requirejs是第一个模块化开发框架，提出了Module transports规范，基于前端模型的一个规范（模块加载时都是异步的）， 受到nodejs的影响， 也支持commonjs规范，建议我们使用module transports规范

requirejs遵守AMD规范

requirejs的官网

requirejs.org

requirejs.cn

requirejs与seajs一样如果想要使用， 需要在页面中引入

根目录：

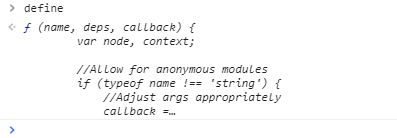
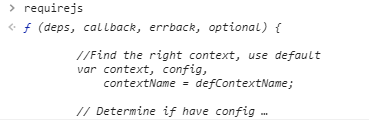
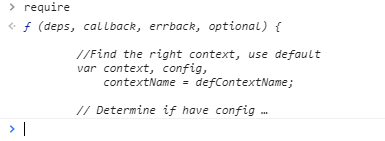
如果有data-main属性，此时是以data-main的属性值为根目录

如果没有data-main属性，此时是以引入的html文件目录为根目录

seajs的根目录是以引入seajs文件的目录为根目录

requirejs的根目录是由引入的html文件来决定

引入requirejs之后会向全局暴露3个变量：



require和requirejs是一样的

引入requirejs的script标签上有一个data-main属性， 它的属性值就是项目的入口文件

|  |
| --- |
| 1. <script type="text/javascript" data-main="js/main.js" src="js/require.js"></script> |

除了以上方式可以引入入口文件

requirejs中require方法就相当于seajs中seajs对象

require方法与seajs.use方法使用方式一模一样

所以，可以通过require方法来引入入口文件

接受两个参数

第一个参数是加载的模块依赖集合

必须是一个数组

第二个参数是回调函数

回调函数中的参数就是前面模块向外暴露的功能（一一对应）

作用域是全局作用域

requirejs对js文件敏感，因此可以省略.js后缀

|  |
| --- |
| 1. // 引入入口文件 2. require(["js/main", "js/dom"], function() { 3. console.log(this, arguments); 4. }) |

## 3.2 定义模块

requirejs与seajs一样都有一个define方法用于定义模块的

传递一个参数

可以是值类型

不能是字符串

也支持对象

当传递一个函数的时候，此时函数遵循commonjs规范，requirejs提供了参数注入的技术，你想要使用哪个模块需要在参数中去注入

此时，向外暴露功能一定是基于commonjs规范

当没有注入exports和module的时候

作用域是window  
当注入exports和module的时候

作用域指向向外暴露功能的对象

传递两个参数

第一个参数可以是字符串

第一个还可以是数组

第二个参数是加载模块的函数

如果第一个参数是字符串，表示模块的id，此时函数是遵循commonjs规范(与上面传递函数的表现形式一致)

如果第一个参数是数组，表示模块的依赖集合，此时函数遵循module transports规范，想要使用哪个模块必须在依赖集合中去注入那个模块， 包括内置模块（require, exports, module）

函数中的参数就是模块依赖集合中向外暴露的接口并且是一一对应的

传递三个参数

第一个参数是字符串，表示模块的id

第二个参数是数组， 表示模块的依赖集合

第三个参数是加载模块的函数（此时函数遵循module trasnports规范）

在module transports规范中，定义的文件名称一定不要与内置模块(require, exports, modul)的名称是相同的

在工作中，最后一种是比较常用的（id一定要与文件路径相统一）

|  |
| --- |
| 1. // 定义模块 2. // 传递数字 3. define(123); 4. // 传递字符串 5. // define("abc"); // 这种方式不可以 6. // 传递布尔值 7. define(true); 8. // 传递对象 9. define({ 10. a: 1, 11. b: 2 12. }); 13. // 传递函数 14. define(function(require, exports, module) { 15. console.log(this, arguments); 16. this.color = "red"; 17. }) 18. // 传递两个参数， 第一个是字符串 19. // define("abc", function() { 20. define("dom", function(require) { 21. console.log(this, arguments); 22. }) 23. // 传递两个参数， 第一个参数是一个数组 24. define(["color", "require", "exports", "module"], function(color, require, exports, module) { 25. console.log(this, arguments); 26. console.log(color); 27. console.log(require); 28. console.log(exports); 29. console.log(module); 30. }) 31. // 传递三个参数 32. // define("abc", ["color"], function() { 33. define("dom", ["color", "require", "exports", "module"], function() { 34. console.log(this, arguments); 35. }) |

## 3.3 引入具有id的模块

在一个模块中没有id，是可以直接引入的

如果在一个模块中有显示的id，是无法直接引入的

第一步在模块的依赖集合中加载模块文件

第二步通过requier方法来指定id （这里的require是异步的）

commonjs是无法引入具有id的模块

|  |
| --- |
| 1. // 引入具有id的模块 2. define(["color", "require"], function(color, require) { 3. console.log(this, arguments); 4. // 通过require指定id 5. require(["myId"], function(color) { 6. console.log(color); 7. }); 8. }) |

## 3.4 模块的覆盖关系

如果一个模块文件中有两个没有id的模块，前面会覆盖后面的模块

如果一个模块文件中有两个相同id的模块，前面会覆盖后面的模块

如果一个模块文件中有两个不同id的模块，是可以同时存在

|  |
| --- |
| 1. // 定义具有id的模块 2. // define("myId", { 3. // color: "red" 4. // }) 5. // 定义两个没有id的模块 6. // define({ 7. // b: 2 8. // }) 9. // define({ 10. // a: 1 11. // }) 12. // 定义两个相同id的模块 13. // define("myId", { 14. // color: "red" 15. // }) 16. // define("myId", { 17. // color: "blue" 18. // }) 19. // 定义两个id不相同的模块 20. define("myId", { 21. color: "blue" 22. }) 23. define("myId1", { 24. color: "red" 25. }) |

## 3.5 接口定义

在seajs中向外暴露接口的方式

exports.接口

module.exports.接口

module.export = 值类型

module.exports = {}

module.exports = 函数

以上几种都是commonjs规范，对于require同样适用

return 值类型

return 对象

return 函数

这几种方式，是module transports规范， 因为它们不需要依赖requier, exports, module, 开发更灵活

define（值类型） 不能字符串

也支持对象

除了以上10种方式， 当传递一个函数的时候并且在函数中注入exports和module， 作用域指向向外暴露功能的对象，所以也可以通过this向外暴露功能

|  |
| --- |
| 1. define(function(require, exports, module) { 2. // 这种方式是绝对不允许 3. // exports = { 4. // } 5. // 1 6. exports.a = 123; 7. // 2 8. module.exports.a = 123; 9. // 3 10. module.exports = 234; 11. // 4 12. module.exports = { 13. a: 1, 14. b: 2 15. } 16. // 5 17. module.exports = function() { 18. console.log("sueccess"); 19. } 20. // }) 21. // module transports 22. // define([], function() { 23. // 6 24. return 123; 25. // 7 26. return { 27. a: 1, 28. b: 2 29. } 30. // 8 31. return function() { 32. console.log("suecess") 33. } 34. // }) 35. // 9 36. define(123); 37. // define("abc"); // 这种方式不允许 38. // 10 也支持对象 39. define({ 40. a: 1, 41. b: 2 42. }) 43. define(function(require, exports, module) { 44. console.log(this); 45. this.color = "red"; 46. }) |

## 3.6 模块信息对象

id: 表示模块的id，默认是与uri不相同，

uri: 模块的文件地址，是相对于html文件的

如果没有data-main属性， 会在路径的前面多一个./

config：是规定如何配置模块的

exports: 向外暴露功能的对象

|  |
| --- |
|  |

# 四、配置

配置：本身这个框架具有的功能，只是隐藏了，需要开启， 要进行配置

require提供了一个define方法用于进行配置

## paths

作用： 用于简化文件路径

值是一个对象

key: 新的文件路径名称

value: 原始路径

举例：

|  |
| --- |
| 1. // 使用requier.config进行配置 2. require.config({ 3. // 配置paths 4. paths: { 5. "module": "module/header" 6. } 7. }) 8. define(["module/header"], function(header) { 9. console.log(header); 10. }) |

## shim

用于将文件转为模块的

key表示模块的路径

value是一个对象

配置接口对象 exports

配置依赖集合 deps

举例：

|  |
| --- |
| 1. // 使用requier.config进行配置 2. require.config({ 3. // 配置shim 4. shim: { 5. "lib/jquery-1.7.2": { 6. // 配置接口 7. exports: "$", 8. // 配置依赖集合 9. deps: [] 10. } 11. } 12. }) |

## map

作用配置模块文件

key模块的文件地址

value是一个对象

key: 引入的模块文件

value: 修改的模块文件

举例：

|  |
| --- |
| 1. // 使用requier.config进行配置 2. require.config({ 3. // 配置shim 4. shim: { 5. "lib/jquery-1.7.2": { 6. // 配置接口 7. exports: "$", 8. // 配置依赖集合 9. deps: [] 10. }, 11. "lib/jquery100": { 12. // 配置接口 13. exports: "$", 14. // 配置依赖集合 15. deps: [] 16. } 17. }, 18. // 配置map 19. map: { 20. "module/header": { 21. "lib/jquery": "lib/jquery-1.7.2" 22. }, 23. "module/footer": { 24. "lib/jquery": "lib/jquery100" 25. } 26. } 27. }) |

## baseUrl

改变根目录

优先级关系：baseUrl > data-main > require

## css插件

在requirejs不能直接引入css文件

要作为模块来引入， 所以要进行配置

map: {

“\*”: {

css: 文件目录

}

}

引入的css文件中，要加上css!前缀

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |