# 什么是seajs？

SeaJS是一个遵循CMD规范的JavaScript模块加载框架，可以实现JavaScript的[模块](https://baike.baidu.com/item/%E6%A8%A1%E5%9D%97/437198?fromModule=lemma_inlink" \t "_blank)化开发及加载机制。

无论是中小型站点，还是大型复杂应用，使用 SeaJS，使用 SeaJS，可以更好地组织 JavaScript 代码。提升代码的可维护性。

# 快速入门

在 SeaJS 的世界里，一个文件就是一个模块。所有模块都遵循 CMD 规范, 我们可以像在 Node 环境中一样来书写模块代码：

define(function(require, exports, module) {

var $ = require('jquery');

exports.sayHello = function() {

$('#hello').toggle('slow');

};

});

将上面的代码保存为 hello.js，然后就可以通过 SeaJS 来加载使用了：

seajs.config({

alias: {

'jquery': 'http://modules.seajs.org/jquery/1.7.2/jquery.js'

}

});

seajs.use(['./hello', 'jquery'], function(hello, $) {

$('#beautiful-sea').click(hello.sayHello);

});

# 模板定义

1. 全局函数define用来定义模块;

语法：define(factory)

参数: factory可以是函数、对象或字符串；

说明：factory是对象或字符串时，表示模块就是对象或字符串本身。

factory是函数时，该函数为模块的构造方法，执行时默认会传入require、exports和module。

define({"foo":"bar"});

define('My name is {{name}}');

define(function(require, exports, module) {

*// 模块代码*

});

语法：define(id?, deps?, factory);

参数：id:模块标识,字符串类型;

deps:模块依赖，数组类型；

factory:可以为函数、字符串、对象；

1. factory函数参数

**通过 require 引入依赖**

require是一个方法接受唯一字符串参数,表示其他模块的标识,返回其他模块提供的接口。

define(function(require, exports, module) {

  var a = require('./a');*// 获取模块 a 的接口*

  a.doSomething();*// 调用模块 a 的方法*

});

**通过 exports 暴露接口**

exports是一个对象,用来向外提供本模块的接口

define(function(require, exports, module) {

*// 分别定义接口*

  exports.foo = 'bar'; *// 对外提供 foo 属性*

  exports.doSomething = function() {};*// 对外提供 doSomething 方法*

*// 或者统一定义所有接口*

  module.exports = {

    foo: 'bar',

    doSomething: function() {}

  };

*// 通过 return 也可以代替exports,直接向外提供接口*

  return {

    foo: 'bar',

    doSomething: function() {}

  };

});

注意:exports 仅仅是 module.exports 的一个引用。在 factory 内部给 exports 重新赋值时，并不会改变 module.exports 的值。因此给 exports 赋值是无效的，不能用来更改模块接口。

module对象上存储了与当前模块相关的一些属性和方法

define('foo', ['bar'], function(require, exports, module) {

    module.id;*// 模块标识:foo*

    module.dependencies;*// 模块依赖:['bar']*

    module.uri;*// 模块绝对路径*

    module.exports;*// 模块对外提供的接口object*

    console.log(module.exports === exports); *// true,exports 是 module.exports 的一个引用*

    module.exports = new SomeClass();*// 如果重新给 module.exports 赋值*

    console.log(module.exports === exports); *// false,exports 将不再等于 module.exports*

});

未定义id时,id的值默认就是uri

# 模块标识

模块标识指定模块,可以是相对标识也可以是顶级标识,不包含文件扩展名。

1. 相对表示

以.或…开始,相对所在模块的路径解析

只出现在模块环境中(define的factory方法里面)

define(function(require, exports, module) {

    require.resolve('./b');*//指向与当前模块文件同路径下的b.js文件*

    require.resolve('../c');*//指向上级文件内的c.js文件*

});

1. 顶级标识

以模块系统的基础路径（即 Sea.js 的 base 路径）来解析

*// 配置基础路径*

seajs.config({

  base: 'http://example.com/assets'

});

*// 如果缺省,则默认为`sea.js`的访问路径*

*// http://example.com/assets/sea.js的base默认值为http://example.com/assets/*

*// 即使`sea.js`的访问路径包含版本号,base也不会包含seajs/x.y.z 字串,以方便版本替换*

*// http://example.com/assets/seajs/1.0.0/sea.js的base默认值为http://example.com/assets/*

define(function(require, exports, module) {

    require.resolve('gallery/jquery/1.9.1/jquery');*//指向http://example.com/assets/gallery/jquery/1.9.1/jquery.js*

});

1. 普通标识

define(function(require, exports, module) {

    require.resolve('http://cdn.com/js/a');*// 绝对路径,指向http://cdn.com/js/a.js*

    require.resolve('/js/b');*// 根路径,指向http://cdn.com/js/b.js*

});

# 文件后缀

Sea.js 在解析模块标识时， 除非在路径中有问号（?）或最后一个字符是井号（#），否则都会自动添加 JS 扩展名（.js）。如果不想自动添加扩展名，可以在路径末尾加上井号（#）。

# 启动

1.在html中引入sea.js,然后调用seajs对象的use方法即可

2.seajs.use(id, callback?)

<script src="path/to/sea.js"></script>

<script>

*// use的相对路径始终相对当前页面来解析*

*// 加载单个模块*

    seajs.use('./main', function(main) {

        main.init();

    });

*// 加载多个模块*

    seajs.use(['./a', './b'], function(a, b) {

        a.init();

        b.init();

    });

</script>

# 配置

seajs.config()方法用来进行配置,接收一个对象参数

seajs.config({

    // 别名

    alias:{

        "$":"jquery/jquery/1.10.1/jquery"

    }

});

// html中

define(function(require, exports, module) {

   var $ = require('$');//=> 加载的是 http://path/to/base/jquery/jquery/1.10.1/jquery.js

});

# 参考博客：

<https://blog.csdn.net/CRayG/article/details/109425837>

<https://seajs.github.io/seajs/docs/#quick-start>

<https://www.w3cschool.cn/seajs/>