

高富帅seajs使用示例及spm合并压缩工具露脸

这篇文章发布于 2012年07月11日, 星期三, 16:45, 归类于 [JS实例](#)。阅读 149802 次, 今日 7 次 [52 条评论](#)

by zhangxinxu from <http://www.zhangxinxu.com>

本文地址: <http://www.zhangxinxu.com/wordpress/?p=2476>

一、扯淡高富帅

很久很久以前.....的很久很久以后, 也就是昨天的昨天(2012-07-07), D2前端技术论坛, 下午3点, 分会场, @老赵分享其开源项目(什么来着? 名字似乎很难记, 让我找找~~) Jscex. 印象较深的是末了, 其戏称seajs为高富帅。

为何有此感慨? 同样是开源项目, seajs因为①原作者个人影响力②推广渠道或者说团队影响力③背后干爹的大力支持等原因, 其知名度以及接受程度要比Jscex高不少。

我自己似乎也有类似的感触, 10年年初的时候, 我自己折腾了一个关于CSS3的小项目, 有别于一般的参考文档, 这里允许用户(主要是同行)添加相关的文章, 以及一些需要注意的技术点。

相关技术文章博览

文章信息

1. [拾人牙慧 - CSS3实现Opera浏览器的logo](#) (2010-4)
2. [CSS “渐进增强”在web制作中常见应用举例](#) (2010-5-23)
3. [让IE6/IE7/IE8浏览器支持CSS3属性](#) (2010-05-23) b

拓展知识点 (由各前端同仁提供)

知识点(1) 提供者: 张鑫旭

目前的CSS3支持border-radius属性值的合并缩写

```
-moz-border-radius: 152px/25px;  
-webkit-border-radius: 152px 25px;  
border-radius: 152px/25px;
```

虽知响应应甚少, 然而, 几年下来, 令我喷舌的是, 居然没有1条更新信息时出自他人之手。头一年, 我还乐此不疲地更新相关文章以及自己发现的一些知识注意点, 但后来, 自己也缴械投降, 懒得折腾了, 纯粹当作静态的参考文档看了。

所谓独木难支, 个人的精力总是有限的, 影响力也是有限的。很多时候, 自己折腾点东西就像鹅毛飘到水里一样, 涟漪都没一个, 难免心有悸动: 要不去傍个高富帅、白富美~~

哈哈, 不过人贵自知, 不是人人都能成为郭美美的, 沉淀心态, 低调蜕变, 说不定, 以前一些渴求的东西会以附加产物的形式悄悄来到你的身边。

二、继续扯, 关于新东西的学习

我记得大学, 应该是根叔吧, 关于毕业, 其说过(大致意思): “如果毕业的时候, 你觉得你做什么都没有问题, 那你就是合格的本科生。”

我将其理解为: 毕业真正考量的不是学到了多少东西, 而是是否有足够强大的学习能力。

不少人询问过我, CSS该怎么学, JS该怎么学? 我能回答的多是, 要重视基础, 要舍得吃苦~~至于其他的, 说得再多, 其实也不一定有用。一来每个人的学习能力与领悟力有差异; 二来, 自己的学习方法不一定适合他人。指个大方向, 应该更有用的, 正所谓“师父领进门修行在个人”。

基础是个很玄妙的东西, 扎实的基础犹如武侠里男主角深厚的内功, 内功深厚了学什么其他功夫都很快, 而且可以很好驾驭; 同样, 基础扎实, 一些新技术什么的上手也快, 而且可以自如灵活运用!

前端发展日新月异，热门的东西容易产生很强烈的热气，使人产生浮躁之气，尤其对于新手而言，这种燥气是不利的。说不定就会舍本逐末，我突然想起了@渔人码头之前发过的一个微博——“说实话看了这标题，看了这年龄，看了这评价，我鸭梨山大！鸭梨山大！鸭梨山大！”，图如下：



所谓“资深”，并不是顺应所谓的潮流，追逐一些新东西。这些光鲜亮丽，华而不实的东西，犹如时尚的衣服一样，第二年就老气过时了。潮流一直在前行，而你，却停不了；因为一旦停下来，就落后了，out了，这样的过活是很累的；而实际上，你所掌握的又可能都是些浮于表面的东西，不过倒是糊弄糊弄些外行，混口饭吃吃！

貌似扯远了，赶快，用绳子拴住，拉直，用力拉，拉啊，拉……尼玛，终于拉回来了，擦下汗😓，我们回到正题，关于seajs. seajs可以算是前端届国产新贵，其年龄（我不确定）应该小于2岁，不过百度之，可以看到相关的文章还是不少的，确实有“高富帅”的迹象。seajs主要处理模块加载，基于commonjs规范，使用spm压缩合并颗粒JS模块。

具体如何如何使用，如用法啊，参数啊什么的，可以点击[这里的手册](#)进行查看。更深入的理解可以参考原作者@玉伯也叫射雕在D2上分享的[PDF文件](#)（来自微盘）。

下面这篇文章也是很不错的：[JavaScript模块化开发库之SeaJS](#)，对模块等对比的分析解释是很受用的。

下面我想讲讲我是如何学习类似seajs这样的新东西。

新东西的学习

1.感性认识，情感化认知

并不是所有的程序员都适合做前端，据说，只有20%~30%，原因？我想，可能一部分原因就是情感化思维，毕竟，我们所做的东西离用户是很接近的。就我而言，我很在意一个新事物从整体上给我的feeling, 这种感性的认识可以让我准确的把握这个事物的特质，于是会形成更合适的策略处理之。换个容易理解的说法：关于追求新认识的姑娘👧，我很在意这个姑娘整体上给我的feeling, 这种感性的认识可以让我准确把握这个姑娘的特质，于是，脑中会自然形成更合适的追求策略。微博上不是有这么个段子嘛：追女孩子，如果这个女孩懵懂无知，就带她去看世间繁华；如果这个女孩看尽世间百态，就带她去骑旋转木马！整体感觉（感性认识）→类型特质→追求策略。

现在的问题是，如果获得准确的整体feeling呢！道听途说，一面之缘等容易产生偏差。我们需要通过多个途径去了解。首当其冲是官方网站对其描述；然后，就是其他人对其评价，认识等。还是追妹子例子，去官网相当于是询问其父母对这个姑娘的评价（当然，父母都会惯着自己的孩子，不好的话一般都不会说的）；询问其他人就是询问姑娘朋友们、同事们的评价，非诚勿扰很多男嘉宾就死在“好友评价”环节，可见后者还是相对比较客观的！

因此，反应到现实就是，去官方网站查看，如果是开源项目，也可以去github查看；查看同行们写的相

关文章或微博，看看其他人是如何介绍与评价的。

2. 实现方式、API参数

就像我们使用一个全新的jQuery插件一样，其中必经一步是查看文档，熟悉其API，知道其一些特殊用法等。好比要熟悉姑娘身高体重，三围，鞋子尺码，知道姑娘的一些癖好等。

3. 实践出真知

古语有云：纸上得来终觉浅，绝知此事要躬行。实践是必须的。为何？

首先，很多时候，我们面对一个全新的东西，其文档或手册上API用法、解释等会让我们不知所云，disorder – 混乱。仿佛回到了大学学“拉普拉斯不等式”的时候，看得头疼。但是，如果你通过自己制作实例，一个一个尝试其所说的内容，很多一眼看上去头疼的东西，立马变得犹如春风拂面。

其次，因为是全新的东西，人总难免被过往的经验（或者说惯性思维）束缚，有些内容真正意思可能并不是脑中所想的那样。举个例子，按照经验式的理解，`console.log([2, 10].sort())`的结果应该是`[2, 10]`，然而，实际上，在JavaScript中，结果是`[10, 2]`，字符串排序。这种跟以往经验不同的表现差异需要通过实践才能有切身的体会。

这也是为何会出现“婚前试爱”这种现象的原因。

seajs的实践

您可能跟我一样，也是最近才接触seajs，您可能也跟我一样，被手册上频繁出现的dependencies, factory等词闪花了眼睛。只怪尔等JavaScript不够精深，但是，seajs号称API简单，因此，虽然一些深入的东西我们一下子消化不了，但是，这并不影响我们使用这个工具。

如何使用？很简单，既然seajs是用来模块化JS的，因此，我们想一个稍微复杂，又能承受的例子作为实践，从头到尾走一遍，顺便spm也摸索下，如果都通过，自然……你说姑娘都让你实践了，从头到尾都*（和谐）过了，顺便*（和谐）也摸索过了。好了，大家都懂的……

OK，我想了想，就实现一个简单的弹出框效果，有一定数量的模块JS，功能实用，且不复杂。下面，如果你是新人，可以跟我一起，看看，如何使用seajs模块化相关脚本，同时实现的弹出框效果。

三、seajs使用示例-弹出框

以下实现为自己的思路，可能比较低级与啰嗦，熟悉seajs的人可以直接跳过。

1. 半透明背景遮罩层

说到弹框，首先想到的是后面黑色半透明的遮罩层。我们就套用：

```
define(function(require, exports, module) {  
    // 模块代码  
});
```

这种写法应该就可以了，需要暴露的接口直接 `exports.*` 就可以了。

恩，我们新建一个名叫 `overlay.js` 的JavaScript文件；然后，我们就着手创建这个遮罩层元素。Wait！这个创建元素的方法应该模块化，方便其他地方公用，例如弹框中元素创建等。

因此，我们需要把 `overlay.js` 搁置一边~~

2. 元素创建模块JS

同目录下，新建一个名为 `elementCreate.js` 的JavaScript文件，同样的套用，如下JavaScript代码，注释也要规范哦 😊~~

```

/**
 * 创建元素
 */

define(function(require, exports) {
    exports.create = function(tagName, attr) {
        var element = null;
        if (typeof tagName === "string") {
            element = document.createElement(tagName);

            if (typeof attr === "object") {

```

`exports.create` 表示模块暴露在外的方法名称是 `create`，`return element`；表示该方法返回的是创建的元素对象。

3. 回到半透明遮罩层

现在，我们就可以使用 `elementCreate.js` 模块了，要使用这个模块，就得使用 `require` 这个函数（注意：`require` 名字必须是 `require`，不能修改）。

```
var elementCreate = require("./elementCreate");
```

于是，`var element = elementCreate.create("div", {})`；就可以创建一个div元素啦！相关完整代码如下（具体遮罩层实现细节可以忽略）：

```

/**
 * 黑色半透明遮罩层
 */

define(function(require, exports, module) {
    var elementCreate = require("./elementCreate");
    var overlay = (function() {
        var element = elementCreate.create("div", {
            styles: {
                display: "none",
                width: "100%",

```

您可以狠狠地点击[这里](#)：[seajs下黑色半透明遮罩效果测试demo](#)



demo页面实际上已经演示了seajs的应用，代码如下：

```

<script src="sea.js"></script>
<script>
seajs.use("./overlay", function(overlay) {
    // overlay.overlay就是遮罩层元素相关对象
});
</script>

```

4. 弹框了

遮罩层搞定了，下面就是弹框了。我们新建一个名为“`flbox.js`”的js文件。实现弹框效果，首先要黑

色半透明层（可以 `require` `overlay.js` 模块），然后标题栏，关闭按钮等创建（可以 `require` `elementCreate.js`）；最后，弹框里面内容需要Ajax加载，但是，并没有ajax的模块，因此，我们要让ajax模块插下队，先搞定之。

5. ajax模块

因为是示例页面，因此，我们没有必要写得很完善与详细，这里，我们就可以单纯实现使用 `get` 方法加载外部一段HTML。于是，捣鼓捣鼓，如下代码：

```
/**
 * ajax方法
 */

define(function(require, exports) {
    exports.get = function(url, succCall) {
        if (url + "" === "undefined") {
            console.log("请求地址缺失！");
            return;
        }
    }
});
```

轻轻松松，您可以狠狠地点击这里：[ajax模块测试demo](#)



测试OK，现在回到弹框模块。

6. 弹框模块

下面，需要的模块都有了，我们只要 `require` 进来，就可以用啦！

```
require("./elementCreate")
require("./ajax")
require("./overlay")
```

虽然，单纯地展示代码我自己也觉得挺傻的没劲，但是，对于天地不知的初学者而已，这种喂奶喂到嘴边的方式是他们学习所需要的。

```
/**
 * 简易弹框
 */

define(function(require, exports) {
    var funCreate = require("./elementCreate")
    , funAjax = require("./ajax")
    , overlay = require("./overlay");

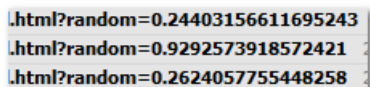
    var elewin = funCreate.create("div", {
        styles: {
```

7. ajax地址增加随机数

为了避免弹框打开时IE浏览器下的缓存问题，我们需要一个url地址增加随机查询字符串的方法。新建名为 `queryRandom.js` 的模块JS文件，键入如下代码：

```
/**
 * URL地址后面增加随机参数
 */
define(function(require, exports, module) {
    exports.queryRandom = function(url) {
        var strQueryRandom = "random=" + Math.random();
        var arrQuery = url.split("?");
        if (arrQuery[1] != undefined) {
            // 含查询字符串
            if (url.slice(-1) === "&") {
                url = url + strQueryRandom;
            } else {
                url = url + "&" + strQueryRandom;
            }
        } else {
            // 不含查询字符串
            url = url + "?" + strQueryRandom;
        }
        return url;
    };
});
```

您可以狠狠地点击这里：[seajs增加随机查询demo](#)



L.html?random=0.24403156611695243
L.html?random=0.9292573918572421
L.html?random=0.2624057755448258

8. 万事具备、终极测试

现在，我们新建一个主线js，命名为 `main.js`，配合demo页面，我们写个很简单的方法——即点击某链接，弹框显示该链接href属性所对应的页面内容。

完整代码如下：

```
/**
 * 主线js
 * 点击小图查看大图测试
 */
define(function(require, exports) {
    var query = require("./queryRandom")
        , flbox = require("./flbox");

    exports.bind = function(element) {
        element.onclick = function() {
            var href = this.href;
            flbox.open(query.queryRandom(href));
            return false;
        };
    };
});
```

您可以狠狠地点击这里：[seajs实现简易弹框效果demo](#)（不支持IE6浏览器）



而调用的脚本很简单：

```
seajs.use("./main.js", function(test) {  
    test.bind(document.getElementById("test"));  
});
```

这很好地体现的seajs API简单的优点。

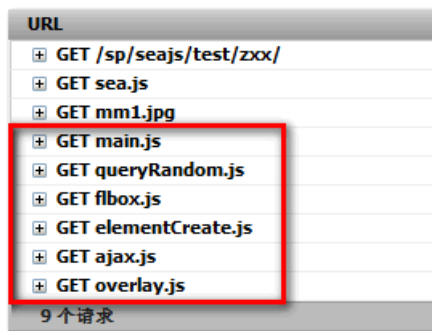
于是乎，遵循commonjs写法规范的各个子模块们，在seajs简单的API下，组合出了我们想要的弹出框效果。而，这些子模块们，又可以被用在其他地方。没有脚本冲突，没有上线时差冲突。可维护性大大提升。这就是seajs的意义所在。

上面简易弹框效果相关代码我已经打包了，点击这里（右键-[目标|链接]另存为）：[seajs-flbox.zip](#)

四、spm作用及使用

虽然说，我们需要的弹框效果已经实现了，但是，却是不能直接上线使用的，为何？

我们看下其http请求，如下截图：

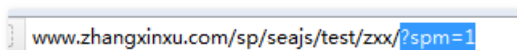


哎呀呀，这么多JS请求，吓着乌索普脆弱的小心脏了。虽然，seajs模块化的书写提高了维护性，但是，也带来了前端性能的问题！如何解决？

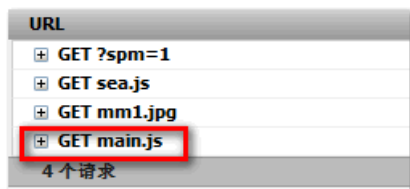
现在，就是救世主spm出场的时候了，spm → seajs package manage?

spm可以合并并压缩seajs中的各个模块JS文件。还是上面的弹框demo页面，我们在地址后面增加” ?spm=

1“，如下图所示，[访问之](#)：



再次刷新页面，查看HTTP请求数，额哈哈，原本6个JS请求，现在合并成1个啦：



命令行中如下代码即实现效果：

```
spm build main.js --combine --app_url zxx
```

然后，main.js中所用到的相对路径require的所有模块就会合并并压缩，新生成的文件（如果没有指定输出路径）会是 `__build/main.js`

```
D:\www.zhangxinxu.com\sp\seajs\test\zxx>spm build main.js --co
x
path.existsSync is now called 'fs.existsSync'.
Building D:\www.zhangxinxu.com\sp\seajs\test\zxx\main.js
... process main.js
... process queryRandom.js
... process flbox.js
... process elementCreate.js
... process ajax.js
... process overlay.js
Combined to __build\main.js
D:\www.zhangxinxu.com\sp\seajs\test\zxx>
```

spm的安装与使用

spm安装与使用github上有介绍。大致如下：

安装

先安装node和npm, <http://nodejs.org/#download>；然后安装spm, 如下命令行代码：

```
$ npm install spm -g
```

```
D:\>cd www.zhangxinxu.com\nodejs
D:\www.zhangxinxu.com\nodejs>npm install spm -g
npm http GET https://registry.npmjs.org/spm
npm http 304 https://registry.npmjs.org/spm
npm http GET https://registry.npmjs.org/uglify-js
npm http GET https://registry.npmjs.org/seajs
npm http 304 https://registry.npmjs.org/seajs
npm http 304 https://registry.npmjs.org/uglify-js
C:\Users\Administrator\AppData\Roaming\npm\spm -> C:\Users\Admin
Roaming\npm\node_modules\spm\bin\spm
spm@0.4.1 C:\Users\Administrator\AppData\Roaming\npm\node_module
├── seajs@1.2.0
└── uglify-js@1.3.2
```

build[option]模块

压缩一个JS模块：

```
$ spm build a.js
```

压缩并合并：

```
$ spm build a.js --combine
```

压缩并合并所有独立模块（包括绝对路径的）：


```
$ spm build a.js --combine_all
```

清除创建的文件：

```
$ spm build --clear
```

通过定义 `build-config.js` 可以做更多的事情：

`build-config.js`:

```
module.exports = {
  "base_path": "/path/to/libs/",
  "app_url": "http://test.com/js/app/",
  "app_path": "/path/to/app/",
  "loader_config": "path/to/init.js"
};
```

更多信息，可以调用：

```
$ spm help build
```

以上为我觉得常用的。这里推荐一篇关于spm使用的文章：[seajs的spm使用摸索](#)。这篇文章对spm各个参数都有解释，另外，还提供了自定义输出路径等参数的使用示例。可以说是对github上使用的很好补充。

spm我也是新手，也没有什么其他心得什么的，就这样吧~~

五、结了个语

中小的企业有其本身的局限性，例如优秀的技术人才，有事就会产生一些无奈。切身说法，我们的web端相关的JS脚本中各个模块从诞生之初就在一个JS文件中，住在一间房子里。一般情况下是没有问题的，但是，多多少少会出现这样的状况：

项目经理：咱们这个图片上传模块要改下，后天上线！

前端仔：没问题。

因为前端仔成功约到了和女神一起看电影，一来兴奋；二来怕加班。其进入仙人模式，欸欸欸三下五除二把JS端的改动都搞定了，就等后台那边对页面修改了！

但是，刚改完不久，悲剧来了——

项目经理：赶快，发现一个重大bug，支付计算貌似有漏洞，赶快fix它，上线！

前端仔：没问题。

又进入无敌模式🤖，快刀斩乱麻，漏洞补上了。结果，准备发布JS上线的时候，前端仔抓狂了😫

尼玛，不能上啊。图片上传页面还没改，上线了虽然支付没有问题，但是，图片上传被搞残了啊！

于是原本兴奋的前端仔像被艾尼路劈了一样，焉了🌵。于是，回到苦逼状态，把之前改好的图片上传模块又改回先前的样子……

这就是没有模块化的痛苦。然而，问题早就知道，但是，一直没有去解决，为何？没人搞这件事。其实

，静态资源压缩并合并对于有经验编程人员讲并不是难事。看大众点评网，虽然大家都是使用MooTools，但是，其静态资源就可以压缩合并，但是，我们这里，没人弄，虽然提过，虽然有人说过愿意做这个事情。我曾研究过淘宝开源的一个JS合并加载工具，但是是PHP的，而我们用的 `.net`，有点折腾不来！于是，就这样拖着~~

不过，现在，貌似可以不用依赖 `.net` 程序员们了，在node环境下，使用seajs模块化书写，spm合并并压缩。先前的一些维护问题显然不复存在。于是，我们的精力可以更多地放在代码实现以及如何更好地划分模块上，这有助于工作效率的提升。

因此，seajs我是会去使用的，虽然它的作者不是foreigner. 至于文章一开始提到的Jscex, 恩.....它可以帮我把到妹子吗？不能！可以帮我钓到大鱼吗？不能！可以帮我车子加油吗？不能！因此，我感觉我不会去用它。

扯淡居多，希望不会浪费您宝贵的时间。感谢阅读，如果表述不准确之处，欢迎指正。

对了，我有问题求教：在弹出框实例页面中，`(require, exports, module)` 这三个参数唯独 `module` 一次也没有使用，因此，我想问下，`module` 在何种情境下使用，有何作用？

《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

(本篇完) // 想要打赏？点击[这里](#)。有话说？点击[这里](#)。



« Stylus-NodeJS下构建更富表现力/动态/健壮的CSS

见多识广：CodePen项目网站简介 »

猜你喜欢

- CSS的样式合并与模块化
- 页面重构“鑫三无准则”之“无宽度”准则
- 基于HTML5 drag/drop模块拖动插入排序删除完整实例
- 万岁，浏览器原生支持ES6 export和import模块啦！
- Stylus-NodeJS下构建更富表现力/动态/健壮的CSS
- JS一般般的网页重构可以使用Node.js做些什么
- 使用electron构建跨平台Node.js桌面应用经验分享
- Service Worker实现浏览器端页面渲染或CSS,JS编译
- windows系统下批量删除OS X系统.DS_Store文件
- 小tips: 使用　等空格实现最小成本中文对齐
- JS HEX十六进制与RGB, HSL颜色的相互转换

分享到： 1

标签： javascript, npm, nodejs, seajs, spm, 压缩, 合并, 模块化

发表评论（目前52条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. 北京soho说道:
2016年03月28日 11:24
废话好多
[回复](#)



2. 围观的臭屌丝说道:
2016年03月21日 11:59
楼主是个张含韵控啊!
[回复](#)



3. 乡下来的前端说道:
2015年09月10日 15:38
module的意义在于 当你的exports只有一个方法的时候,比如
exports=function(a,b){
return a+b
}
这样是错误的 因为这样引用就丢了, 这种情况就可以用
module.exports=function(a,b){
return a+b
}
我是这么认为的~~。。装了B快跑
[回复](#)



4. 阿磊说道:
2015年06月25日 17:19
鑫哥哥, 那么问题来了。
1打包: 后每个页面会针对一个合并压缩了的js文件, 这样, 浏览器就每次请求一个页面都会把对应的js加载一次, 而公用的脚本也会重复的加载一次 (因为每个包中都有)。
2未打包: 虽然第一次请求页面会加载次数很多, 大量公用的js文件加载过来, 但是后面的页面就不会再次请求公用的js模块。
总结: 如果你是一个网站, 每个页面都有可能被独立访问, 打包好。如果是类似app 有流程的站, 我觉得不打包, 只压缩会好些。不知道你意下如何? ?
[回复](#)



5. 哦哦说道:
2015年06月24日 23:24
赞...
[回复](#)



6. **recoo**说道:

2015年06月16日 10:36

这篇文章不错！

[回复](#)



7. **阿司匹林的阿si**说道:

2015年04月16日 17:17

之前看过很多篇楼主的文章，对前端刚刚入行的新手受益匪浅。

另外看过文档，exports是对module.exports的一个引用， exports一般是用来对外暴露单一属性或者方法，比如 exports.myName = 'Arthas'， 或者exports.doSth = function(){};

当需要暴露整个对象时，文档说是应该使用module.exports = {myName: 'arthas', doSth: function(){}}, 这样的， 而不是exports = {myName: 'arthas', doSth: function(){}}

[回复](#)



nuintun说道:

2015年05月28日 10:05

正解， exports = {myName: 'arthas', doSth: function(){} } 是返回不了赋值对象的，模块的返回值是 module.exports 对象~

[回复](#)



8. **业界大牛**说道:

2015年03月14日 08:54

module是有使用的~
exports就是省略了module呢

[回复](#)



9. **刘宗源**说道:

2015年02月10日 10:29

鑫哥文风很棒！浅显易懂！点个赞

[回复](#)



10. **longboy**说道:

2015年01月21日 15:31

博主:

“smp我也是新手，也没有什么其他心得什么的，就这样吧~~”。

这里的smp应该是spm吧？

[回复](#)



11. **kayjack**说道:

2014年08月7日 17:53

spm 3不行。

[回复](#)



12. **appletxm**说道:

2014年06月25日 14:01

帅哥，你很幽默，哈哈

[回复](#)



13. **spm**打包部署老出问题说道：

2014年05月14日 14:21

您好，大牛哥，我这边 1. node.js 2. npm install spm -g 3. \$ spm -V 或者 \$ spm build 或者到项目根目录下去build一个 init.js 所有依赖模块都没有被打包，总是出现环境变量问题或者 spm is required in package.json 请问是什么问题？ ？ ？ ？

[回复](#)



14. **damon**说道：

2013年12月5日 01:25

请问一下，有没有您说的 玉伯 在D2上分享的pdf文件，貌似被删掉了，发给我一份看看，最近在折腾spm呢！万分感谢

[回复](#)



15. **噬魂**说道：

2013年10月2日 21:15

关于module这个参数，官方文档上面写得比较清楚：<https://github.com/seajs/seajs/issues/242>。

[回复](#)



16. **时生**说道：

2013年07月11日 10:48

```
if (keyAttr === "class") {  
  keyAttr = "className";  
}  
element[keyAttr] = attr[keyAttr];
```

这里本来是obj = {"class": "select"};

这样写了之后不就变成element["className"] = obj["className"]了吗

那获取出来的obj没有className这个属性呐

[回复](#)



张鑫旭说道：

2013年07月11日 18:12

@时生 没错，这就改正之！多谢纠正！

[回复](#)



17. **大神是淘宝的么？** 说道：

2013年05月13日 19:32

玉帛不也是淘宝的么

[回复](#)



张鑫旭说道：

2013年05月14日 10:32

@大神是淘宝的么？ 1. 首先大神不敢当，还只是小辈；2. 我不是淘宝的。

[回复](#)



18. **lei.shi**说道：

2013年04月24日 16:41

一向不回帖，但是从你这里看了很多资料，谢谢。

[回复](#)



19.

老左说道:

2013年04月23日 09:59

这个帖子必须订，非常喜欢你这种教程贴，真的是很适合大部分新人。并且这类新人遍地都是，真的非常赞一下楼主，你真的太给力了，很多教程，没有坚实的基础，要很快上手是困难重重的。赞！！

[回复](#)


20.

spm build 报错说道:

2013年04月21日 17:40

我项目目录在d:\test\src 这个目录下有a.js,在此目录spm build a.js 报错了，有个warning：
missing “spm” in package.json 然后a也没有被压缩，是为啥啊？求教

[回复](#)


21.

shenwa12说道:

2013年04月2日 10:14

(require, exports, module)这三个参数唯独module一次也没有使用，因此，我想问下，module在何种情境下使用，有何作用？
那是因为module是代表模块自身
除了调试用的信息 用到最多的就是
module.exports = xxxx
相比你exports.xxx = function
module.exports 能输出多个 并且看上去比较清爽
seajs2.0出了 也还可以直接return出接口。。
如果觉得没用 那就不用好了 其实意义不是很大

[回复](#)


22.

落落说道:

2013年03月12日 21:30

exports和module.exports是相等的

[回复](#)


23.

sweetehs说道:

2013年01月21日 14:05

压缩合并需要package.json啊，我试验了一下不好使！55555555

[回复](#)


24.

ajiao5198说道:

2012年12月13日 16:05

楼主还是这么搞笑

[回复](#)


25.

alert说道:

2012年09月5日 11:36

我可不可以本地打包完js 然后上传到服务器 spm要在服务器上运行打包吗

[回复](#)


26.

wutiansheng说道:

2012年08月31日 15:39

嗯，明白了，不支持file协议。要用http访问

[回复](#)



27.

wutiansheng说道：

2012年08月30日 17:30

您可以狠狠地点击这里：seajs实现简易弹框效果demo（不支持IE6浏览器）
弹出页面，可以正常运行，看到预期效果
但下载下来的示例，在IE9里，就是直接激活了链接，没有预期效果
Firefox15里，点击图片没任何反应
我是直接打开的index.html文件，不是http方式访问。

回复

菜鸟学js说道：

2014年08月7日 10:21

我也是... 本地打开..为啥没效果呢 思索再三 毫无头绪 求楼主指点..

回复

菜鸟学js说道：

2014年08月7日 17:28

对不起,自己傻逼了,ajax肯定要通过http服务器..的..

回复
28.

xiaoxiehang说道：

2012年08月21日 20:48

菜鸟不会js看着很蛋疼，求jquery的例子，不要太深奥的，跪谢

回复
29.

gtrate说道：

2012年07月31日 16:23

好吧，我通过lz的博文才看懂了seajs的精髓

回复
30.

alex说道：

2012年07月17日 11:39

楼主写的文章确实是有内容，但是这个语气怎么有点愤青。。。严重影响阅读时的心情，哈哈

回复
31.

zoowar说道：

2012年07月15日 13:19

尼玛，博主这不是写小说.....

回复
32.

阿邦说道：

2012年07月14日 11:47

悲催，我们也是.Net好多东西都扯淡被搁置了。－。
但Seajs可以用。哈哈

回复

33. **ax1.1**说道:
2012年07月13日 09:58
- 曾经在项目里实践过一把seajs,模块化并行开发什么的都挺好,就是最后spm打包不行,压缩合并后的js竟然出现了兼容问题...,spm的压缩如果能配置,或者能换成GC YC或者UglifyJS就好了。
- seajs还有个问题就是如果想在现有工程中使用,还要对现有代码模块化,即使现有代码已经是按模块写的,还要挨个加define。
- 如果楼主要压缩合并js,可以看看gruntjs,jquery就用的这个,node环境,可以配置JSHint 和 UglifyJS
相当于node的ant



[回复](#)

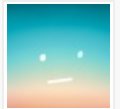
张 鑫旭说道:
2012年07月14日 22:22

@ax1.1 gruntjs, 学习了!

[回复](#)



34. **锐风**说道:
2012年07月13日 03:35
- 有空研究研究 IndexedDB 和 MutationObserver 吧..
还有 Mutation Summary.



[回复](#)

35. **xiaofanV587**说道:
2012年07月12日 20:21
- 我没用spm install jquery 就直接 引用的
原来如此 我install看看



[回复](#)

36. **Emma**说道:
2012年07月12日 13:12
- hello楼主, 经常上你的网站学习, 非常感谢!
请教下楼主公司的网址是什么, 我们公司也是用的mootools, 想上去看看学习学习!



[回复](#)

37. **cookieu**说道:
2012年07月12日 13:11
- @xiaofanV587 你是怎么配置的, 一般的库转换为seajs的库还需要额外处理下。不过像jquery spm可以直接安装的。
- spm install jquery



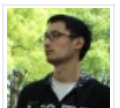
[回复](#)

38. **cookieu**说道:
2012年07月12日 13:08
- module 用于获取模块本身的信息



[回复](#)

39. **Leo Deng**说道:
2012年07月12日 11:02
- 这么多年过去了, 为什么带图的demo还是张含韵啊.....



[回复](#)

40. **honingwon**说道:
2012年07月12日 09:23

@xiaofanV587
SeaJS有检测机制，不会重复加载模块的。你可以仔细观察一下http请求。
[回复](#)
41. **honingwon**说道:
2012年07月12日 09:19

SeaJS遵循的不是commonJS规范而是CMD规范。CMD规范与commonjs modules规范很相像。
官方说，spm新版正在开发中，一起期待api和文档上有大的改进。
另，module应该在高级实践中才会有用吧，比如说插件开发。也可能是为了实现功能api的完整性才加的。
[回复](#)
42. **libo**说道:
2012年07月12日 08:31

把你的文章放在别处，也是知道你写的，哈哈
[回复](#)
43. **愚人码头**说道:
2012年07月12日 08:31

其实一个开源项目，编码和社区一样重要！
[回复](#)
44. **逸川**说道:
2012年07月11日 23:50

具体可以看下这一页，写的非常清楚：<https://github.com/seajs/seajs/issues/242>
[回复](#)
45. **逸川**说道:
2012年07月11日 23:47

module 是一个对象，上面存储了与当前模块相关联的一些属性和方法。
我觉得最有用的一点是 module.exports 用来替代 exports，你可以这么做

file a.js

```
define(function(require,exports,module){  
  module.exports = function() {};  
});
```


file b.js

```
define(function(require,exports,module){  
  new require('a');  
});
```


[回复](#)
46. **xiaofanV587**说道:
2012年07月11日 19:50

seajs 本地配置加载jquery 一直加不进来 不知道为啥。。。
还有a.js b.js c.js 都require x.js 当执行a b c时候 他加载3次 这种 不多次加载了吗 楼主遇到没
[回复](#)

最新文章

- » 常见的CSS图形绘制合集
- » 粉丝群第1期CSS小测点评与答疑
- » 分享三个纯CSS实现26个英文字母的案例
- » 小tips: 纯CSS实现打字动画效果
- » CSS/CSS3 box-decoration-break属性简介
- » CSS :placeholder-shown伪类实现Material Design占位符交互效果
- » 从天猫某活动视频不必要的3次请求说起
- » CSS vector-effect与SVG stroke描边缩放
- » CSS ::backdrop伪元素是干嘛用的?
- » 周知: CSS -webkit-伪元素选择器不再导致整行无效

今日热门

- » 常见的CSS图形绘制合集 ⁽¹⁹³⁾
- » 未来必热: SVG Sprite技术介绍 ⁽¹²⁰⁾
- » 粉丝群第1期CSS小测点评与答疑 ⁽¹¹⁵⁾
- » HTML5终极备忘大全 (图片版+文字版) ⁽⁹³⁾
- » 让所有浏览器支持HTML5 video视频标签 ⁽⁸⁶⁾
- » Selectivizr-让IE6~8支持CSS3伪类和属性选择器 ⁽⁸²⁾
- » CSS3下的147个颜色名称及对应颜色值 ⁽⁸⁰⁾
- » 视区相关单位vw, vh..简介以及可实际应用场景 ⁽⁷⁶⁾
- » 写给自己看的display: flex布局教程 ⁽⁷⁶⁾
- » 小tips: 纯CSS实现打字动画效果 ⁽⁷⁶⁾

今年热议

- » 《CSS世界》女主角诚寻靠谱一起奋斗之人 ⁽⁷⁶⁾
- » 不借助Echarts等图形框架原生JS快速实现折线图效果 ⁽⁶⁴⁾
- » 看, for..in和for..of在那里吵架! ⁽⁶⁰⁾
- » 是时候好好安利下LuLu UI框架了! ⁽⁴⁷⁾
- » 原来浏览器原生支持JS Base64编码解码 ⁽³⁵⁾
- » 妙法攻略: 渐变虚框及边框滚动动画的纯CSS实现 ⁽³³⁾
- » 炫酷H5中序列图片视频化播放的高性能实现 ⁽³¹⁾
- » CSS scroll-behavior和JS scrollToView让页面滚动平滑 ⁽³⁰⁾
- » windows系统下批量删除OS X系统.DS_Store文件 ⁽²⁶⁾
- » 写给自己看的display: flex布局教程 ⁽²⁶⁾

猜你喜欢

- CSS的样式合并与模块化
- 页面重构“鑫三无准则”之“无宽度”准则
- 基于HTML5 drag/drop模块拖动插入排序删除完整实例
- 万岁, 浏览器原生支持ES6 export和import模块啦!
- Stylus-NodeJS下构建更富表现力/动态/健壮的CSS
- JS一般般的网页重构可以使用Node.js做些什么
- 使用electron构建跨平台Node.js桌面应用经验分享
- Service Worker实现浏览器端页面渲染或CSS,JS编译
- windows系统下批量删除OS X系统.DS_Store文件
- 小tips: 使用　等空格实现最小成本中文对齐

