

SeaJS 从入门到精通

seajs.org

2012.07

Topics

- I. 解决什么问题
- II. SeaJS 入门
- III. 核心设计与实现
- IV. 开源与未来

1. 解决什么问题

页面中引入 script

```
<html>
<head>
<script src="library.js"></script> <!-- include a library of JavaScript code -->
</head>
<body>
<p>This is a paragraph of HTML</p>
<script>
// And this is some client-side JavaScript code
// literally embedded within the HTML file
</script>
<p>Here is more HTML.</p>
</body>
</html>
```

这是犀牛书里推荐的写法，
用来写写博客网站没什么问题。

当单个 script 变大时

A screenshot of a web browser window showing a single JavaScript file. The address bar displays 'view-source:a.tbcdn.cn/app/tc/detail.source.js'. The code is a single large block of JavaScript, with line numbers visible on the left side of the editor. The code includes DOM queries, CSS style changes, and event listeners for mouseover and mouseout events.

```
7710     tab = Dom.query( ".detail-report" );
7711
7712     var over = function(){
7713         Dom.css( tip_info, "visibility", "visible" );
7714         Dom.addClass( tip, "hover" );
7715         Dom.addClass( tab, "report-hover" );
7716     }
7717     var out = function(){
7718         Dom.css( tip_info, "visibility", "hidden" );
7719         Dom.removeClass( tip, "hover" );
7720         Dom.removeClass( tab, "report-hover" );
7721     }
7722     return{
7723         init:function(){
7724             Event.add( tab, "mouseover", over );
7725             Event.add( tab, "mouseout", out );
7726         }
7727     }
7728 }();
7729 }());
```

近8000行的单文件，多人协作不方便。
解决办法：拆。

拆开后的烦恼

```
<link rel="stylesheet" href="http://images.apple.com/v/home/k/styles/billboard.css" type=
<link rel="stylesheet" href="http://images.apple.com/home/styles/billboard.css" type="te
<script src="http://images.apple.com/global/scripts/lib/prototype.js" type="text/javascrip
<script src="http://images.apple.com/global/scripts/lib/scriptaculous.js" type="text/jav
<script src="http://images.apple.com/global/scripts/browserdetect.js" type="text/javascr
<script src="http://images.apple.com/global/scripts/apple_core.js" type="text/javascript
<script src="http://images.apple.com/global/scripts/search_decorator.js" type="text/java
<script src="http://images.apple.com/global/ac_retina/ac_retina.js" type="text/javascrip


<script src="http://images.apple.com/global/scripts/promomanager.js" type="text/javascri
<script type="text/javascript">
    Event.onDOMReady(function() {
        new AC.PromoManager('home-promo-rotate-20120618', 'promo-rotate');
    });
</script>
>
id="home" class="home">
<script type="text/javascript">
var searchSection = 'global';
var searchCountry = 'us';
pt>
```

拆开能缓解可维护性，但性能不佳：

- 1) 请求数太多
- 2) 同步阻塞

请求数太多的解决方案： combo

```
<script type="text/javascript" src="http://r.suc.itc.cn/
combo.action?v.11112401&r=
/itoolbar/plugins/jquery-1.6.2.js|
/itoolbar/core/passport.js|
/itoolbar/core/base64.js|
/itoolbar/core/jquery.cookie.js|
/itoolbar/jquery.itoolbar.index.js
&t=js&c=utf-8" charset="utf-8"></script>
```

Size Content	Time Latency	Timeline	1.45s	2.18s	2.91s
56.00KB 151.36KB	1.97s 286ms				

同步阻塞严重影响无缓存时的页面打开速度！

同步阻塞的解决方案：loader



```
70 FP.add({  
71   'direct-promo': {  
72     fullpath: 'http://a.tbcdn.cn/p/fp/2010c/js/fp-direct-promo-min.js?  
t=20111115.js'  
73   },  
74   'fp-mods': {  
75     fullpath: 'http://a.tbcdn.cn/??s/kissy/1.1.6/switchable/switchable-pkg-  
min.js,s/kissy/1.1.6/suggest/suggest-pkg-min.js,s/kissy/1.1.6/datalazyload/datalazyload-  
pkg-min.js,s/kissy/1.1.3/flash/flash-pkg-min.js,p/search/searchsuggest-  
min.js,p/fp/2011a/expressway/profile-min.js,p/fp/2011a/header/header-  
min.js,p/fp/2011a/attraction/attraction-min.js,p/fp/2011a/attraction/ald-  
min.js,p/fp/2011a/expressway/expressway-min.js,p/fp/2011a/category/category-  
min.js,p/fp/2011a/category/sub-promotion-min.js,p/fp/2011a/guide/alimama-ecpm-  
min.js,p/fp/2010c/js/fp-hubble-monitor-min.js,p/fp/2011a/hotsale/p4p-  
min.js,p/fp/2011a/local-life/local-life-min.js,p/fp/2011a/footer/footer-min.js?  
t=20111128.js'  
76   }  
77 });
```

解决了同步阻塞，但需要配置，不够简洁。
同时带来了一些新的问题。

YUI 的命名空间困局

```
YUI().use("io-base", "mod-a", "mod-b", function(Y) {  
  
    // Y.on 是 mod-a 添加的、还是 mod-b 添加的?  
    // 如果都往 Y 上添加 on 方法，会出现什么问题?  
    Y.on(...);  
  
    // io-base 在 Y 上添加了 io 方法。  
    // 是怎么知道的呢？查文档？为什么不是 "Y.IO" ?  
    var request = Y.io(...);  
  
});
```

Y 是另一个“全局”变量！！！！

版本“锁定”的苦恼

YUI 2.7.0

```
<script  
src="http://a.tbcdn.cn/tbra/1.0/tbra-widgets.js"></script>  
<script  
src="http://a.tbcdn.cn/yui/2.7.0/build/container/container-min.js"></script>  
<script  
src="http://a.tbcdn.cn/kissy/1.0.8/ks-core-min.js"></script>
```

KISSY 1.0.8

老代码多
升级代价大

+

版本冲突
资源不够
时间紧张
...

=

决定在旧版本
上继续开发

依赖旧版本的代码更多



“在线”开发与调试

```
#/etc/hosts
```

```
127.0.0.1    a.tbcdn.cn
```



Willow

Proxy



所有方案都不够敏捷。

跨环境共享

```
QueryString.parse( 'a=b&b=c' );  
//=> { a: 'b', b: 'c' }
```

```
QueryString.stringify( { 'foo': 'bar' } );  
//=> 'foo=bar'
```

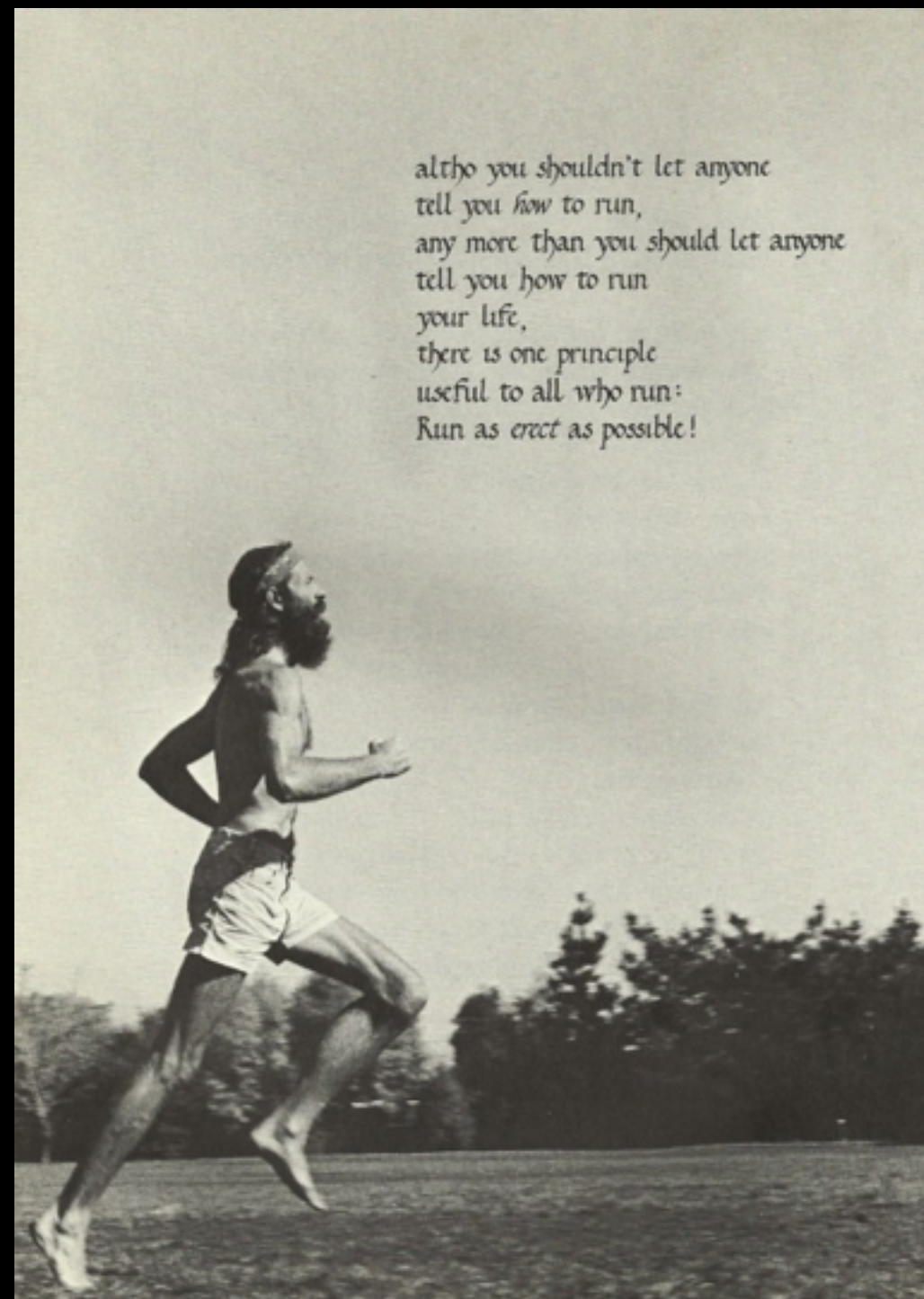


这是一个梦，但并不遥远。

核心问题

- 可维护性
- 性能

JavaScript development
needs to be done differently.



等待还是前行？

II. SeaJS 入门

SeaJS 是什么

A **Module Loader** for the **Web**

SeaJS 的应用场景

- SeaJS 是更自然的代码组织方式
- 只要项目的 JS 文件超过 3 个，就适合用
- 文件越多，则越适合

基本用法

init.js

```
define(function() {  
    alert('Hello, world!');  
});
```

test.html

```
<script src="libs/seajs/1.2.0/sea.js"></script>  
<script>  
    seajs.use('./init');  
</script>
```

- 一个模块一个文件
- 使用 define 定义模块
- 使用 use 使用模块

基本用法

init.js

```
define(function(require, exports) {  
  exports.message = 'Hello, world!';  
});
```

test.html

```
<script src="libs/seajs/1.2.0/sea.js"></script>  
<script>  
  seajs.use('./init', function(init) {  
    alert(init.message);  
  });  
</script>
```

- 使用 exports 对外提供接口
- 使用 use 使用加载的模块对象

基本用法

init.js

```
define(function(require, exports) {  
  var weather = require('./weather');  
  var temperature = weather.getTemperature('Beijing');  
  
  exports.message = 'The temperature of Beijing is ' +  
    temperature;  
});
```

weather.js

```
define(function(require, exports) {  
  var io = require('./io');  
  ...  
  exports.getTemperature = function(city) { ... };  
});
```

- 使用 require 获取其他模块对象
- 自动处理依赖
- 关注点分离：直接依赖的模块 + 向外提供的接口

基本用法

模块的定义：

```
define(function(require, exports, module) {  
    var a = require('./a');  
    ...  
    exports.x = ... ;  
});
```

模块的使用：

```
seajs.use('./init', function(init) {  
    ...  
});
```

快速参考（最常用的 7 个 API）：

<https://github.com/seajs/seajs/issues/266>

SeaJS 入门文档

使用文档:

<http://seajs.org/docs/#api>

更多例子:

<http://seajs.org/docs/examples/>

For Users

- 模块系统
- CMD 模块定义规范
- 模块标识
- require 书写约定
- 模块加载器
- 配置
- 常用插件
- 打包部署
- 快速参考

请仔细看例子和文档，阅读完这些文档，
SeaJS 就绝对入门了。

切忌浮躁
心静自然牛

III. 核心设计与实现

1. 模块系统

什么是系统

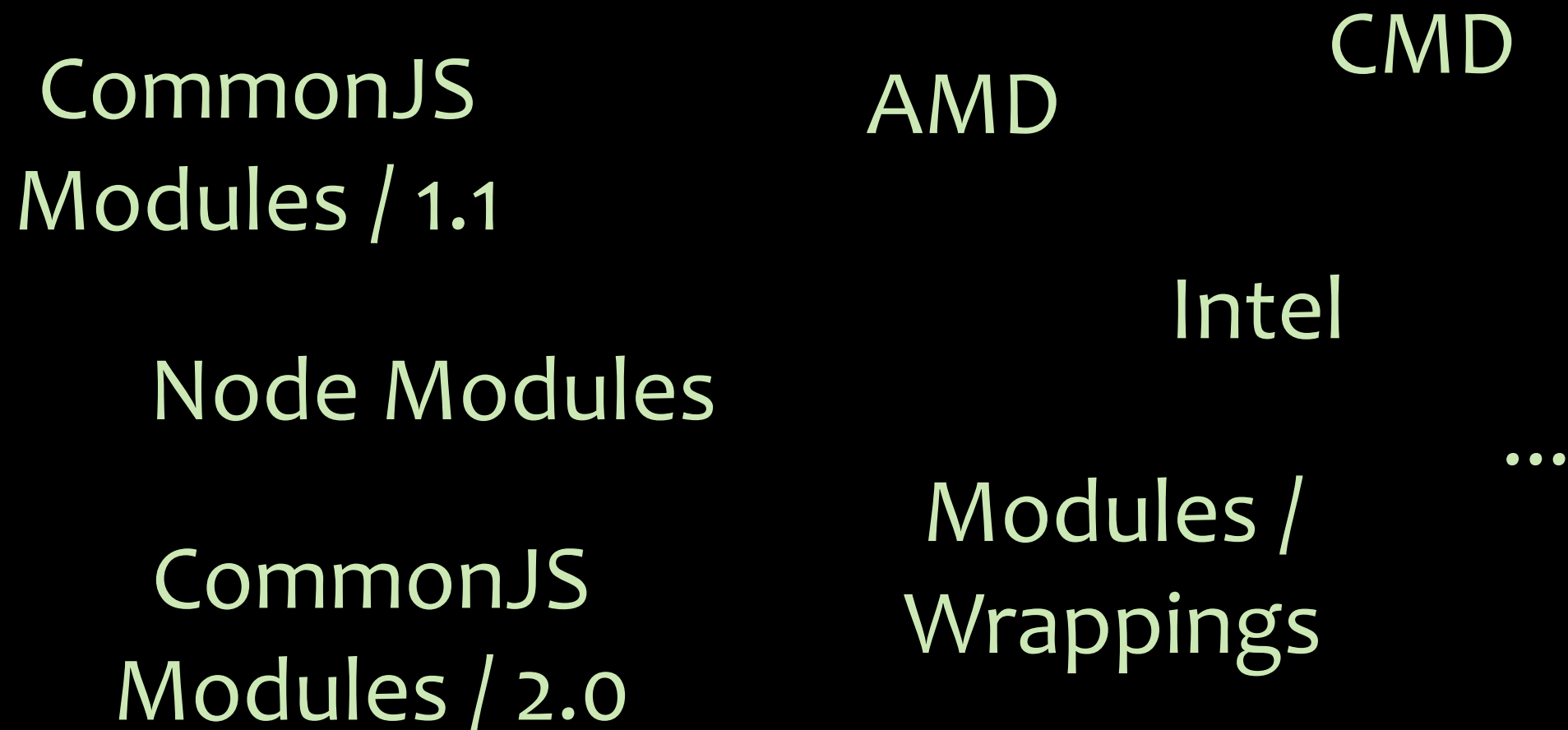
- 系统由个体组成
- 个体之间有关连，按照规则协同完成任务

<https://github.com/seajs/seajs/issues/240>

模块系统的基本问题

- 系统成员：模块是什么？
- 系统通讯：模块之间如何交互？

模块定义规范



所有这些规范，都是为了解决
模块系统的两个基本问题。

CMD

- CMD - Common Module Definition
- 尽量与 CommonJS Modules/1.1 以及 Node Modules 的规范保持一致
- 同时考虑 Web 特性

<https://github.com/seajs/seajs/issues/242>

CMD 模块

```
define(function(require, exports, module) {  
  
    var $ = require('jquery')  
    var math = require('./math')  
  
    exports.doSomething = ...  
  
})
```

2. 模块加载器

加载器的基本功能

- 模块定义规范的实现，这是模块系统的基础。
- 模块系统的启动与运行。

<https://github.com/seajs/seajs/issues/260>

Node 的实现

```
var math = require('./math')
```

Step 1: resolveFilename

Step 2: load

Step 3: compile

<https://github.com/joyent/node/blob/master/lib/module.js>

从 Server 到 Web

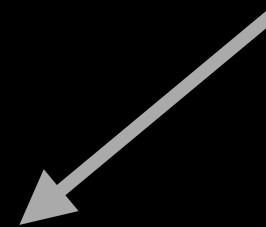
- node_modules 查找不适合 Web 端
- 文件的同步读取不适合 Web 端
- 跨域
- 性能
- 浏览器兼容性



SeaJS 的实现

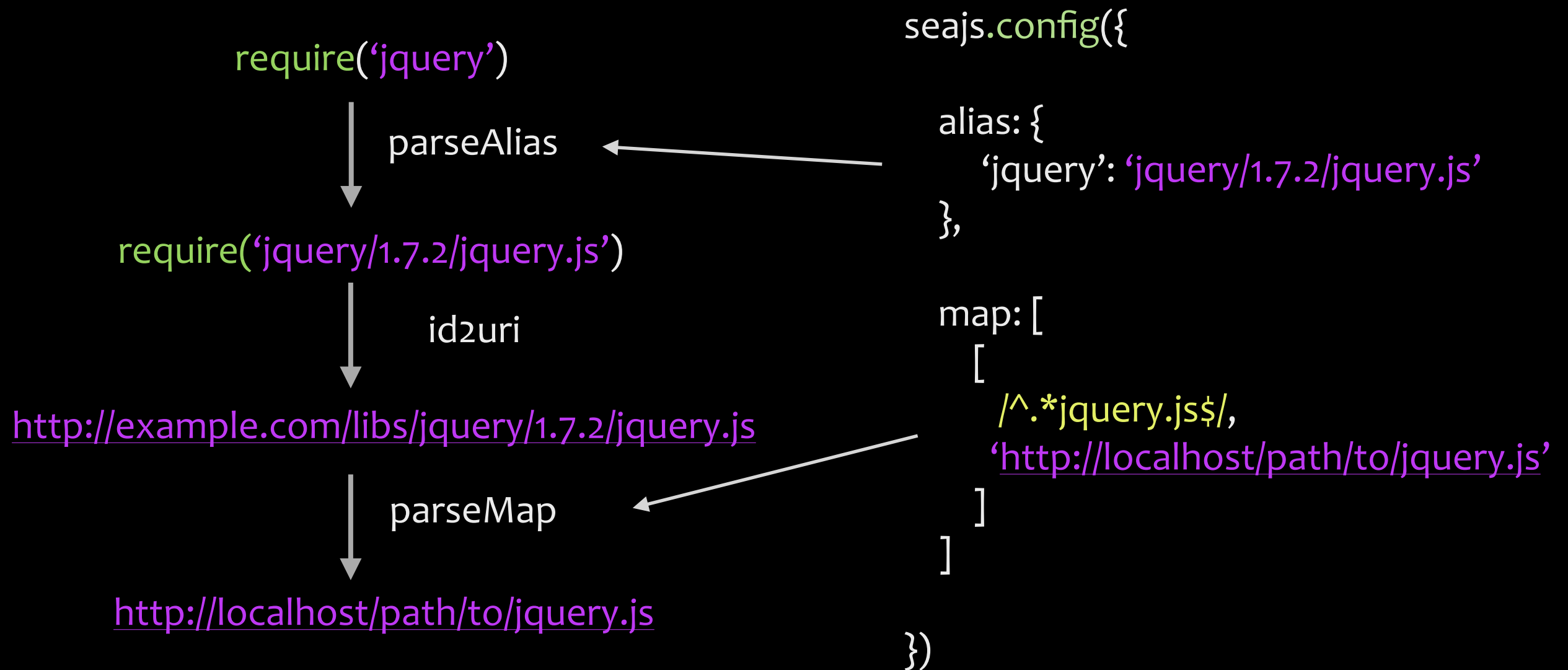
```
/* a.js */  
define(function(require, exports, module) {  
  var b = require('./b')  
  var c = require('./c')  
  // ...  
})
```

```
/* main.js */  
seajs.use('./a')
```



- Step 1: 解析 './a'
- Step 2.1: 下载 a
- Step 2.2: 执行 define, 保存 a 的 factory
- Step 2.3: 得到依赖 b 和 c
- Step 2.4: 加载 b 和 c
- Step 3: 执行 a 的 factory, 得到 a 的 module.exports

Step 1: 路径解析



Step 2: 模块加载

- Inserted Script、XHR、Web Worker...
- SeaJS 选择 Inserted Script 方案

如何得到依赖

factory.toString() + 正则匹配

<https://github.com/seajs/seajs/blob/master/src/util-deps.js>



require('./xxx')

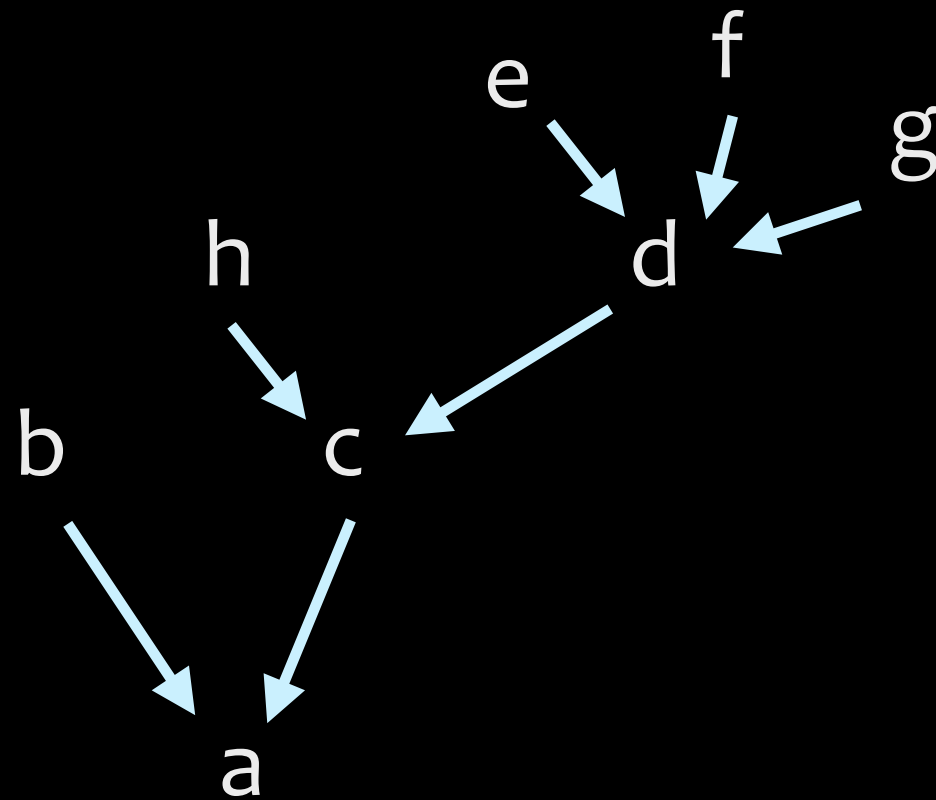
Rule 1: factory 第一个参数的命名必须是 require

Rule 2: require 函数只能接收字符串值

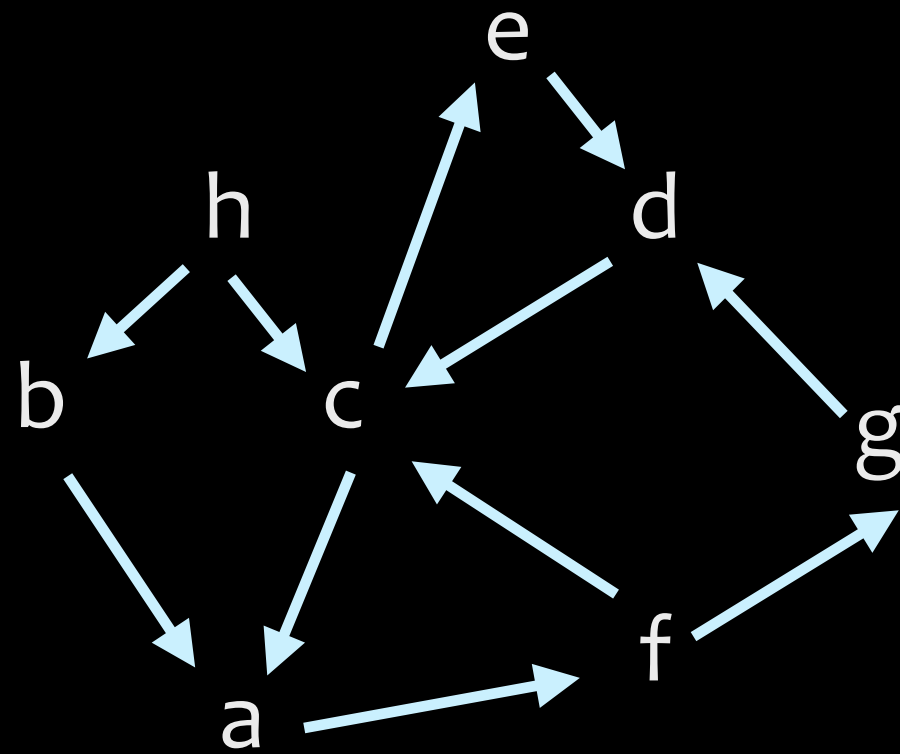
Rule 3: 不要覆盖 require

<https://github.com/seajs/seajs/issues/259>

依赖的回调树



循环依赖



加载时的循环等待

`isCircularWaiting(module, uri)`

<https://github.com/seajs/seajs/blob/master/src/module.js>

编译时的循环等待

```
if (module.status === STATUS.COMPILING) {  
    return module.exports  
}
```

<https://github.com/seajs/seajs/blob/master/src/module.js>

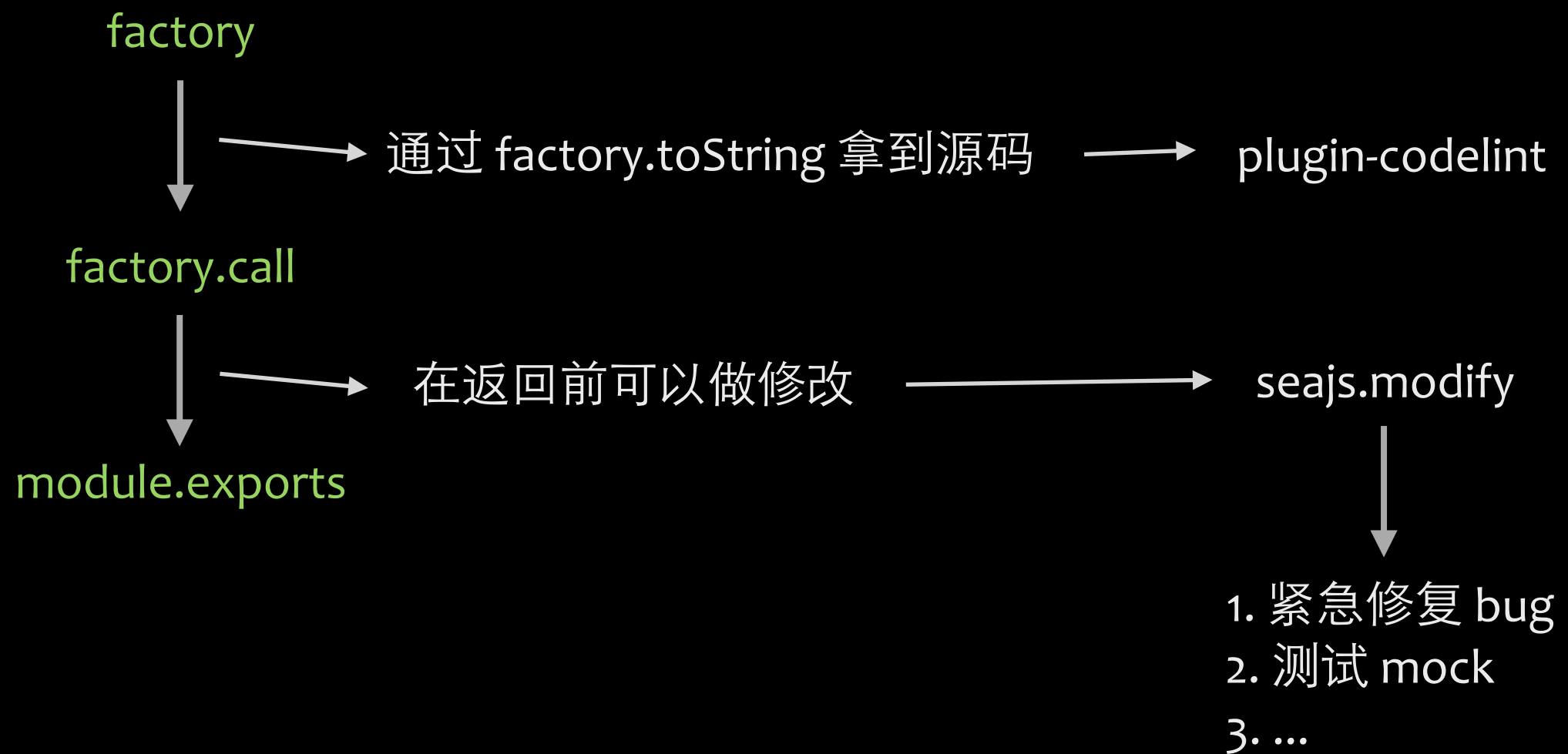
Step 3: 代码编译

```
/* a.js */  
define(function(require, exports, module) {  
  var b = require('./b')  
  var c = require('./c')  
  // ...  
})
```

```
module.require = require  
module.exports = {}
```

```
module.factory.call(  
  window,  
  module.require,  
  module.exports,  
  module  
)
```

编译前后



原理就这么简单！
实现的细节请参考源码。

3. 调试与插件开发

调试友好

For Advanced Users

- 常见错误
- SeaJS 的调试接口
- debug 调试插件
- combo 插件
- modify 功能
- 与 Node 兼容

<http://seajs.org/docs/#api>

插件开发

For Plugin Developers

- 插件开发指南
- 源码阅读指引

<http://seajs.org/docs/#api>

期待你的参与

Contributors

Contributions over time



Commit Activity

Commit activity over the previous year



Code Frequency

Additions and deletions over time



Impact

Individual impact over time



<https://github.com/seajs/seajs>

4. SeaJS 的可靠性

SeaJS 的基本假设

A --- 表示 a.js 执行时的时间

a --- 表示 a.js 的 onload / onerror 时的时间

开发时，SeaJS 要求：A 与 a 紧相邻

上线后，SeaJS 要求： $A < a$

<http://seajs.org/test/research/onload-order/test.html>

<https://github.com/seajs/seajs/issues/130>

疯狂的测试用例

<http://seajs.org/test/runner.html>

PC、Mobile

理论上是个浏览器就应该可以跑

已有哪些公司在用



IV. 开源与未来

开源的目的

- 把好的东西分享出来
- 让好的东西变得更好
- 其他一切皆是浮云

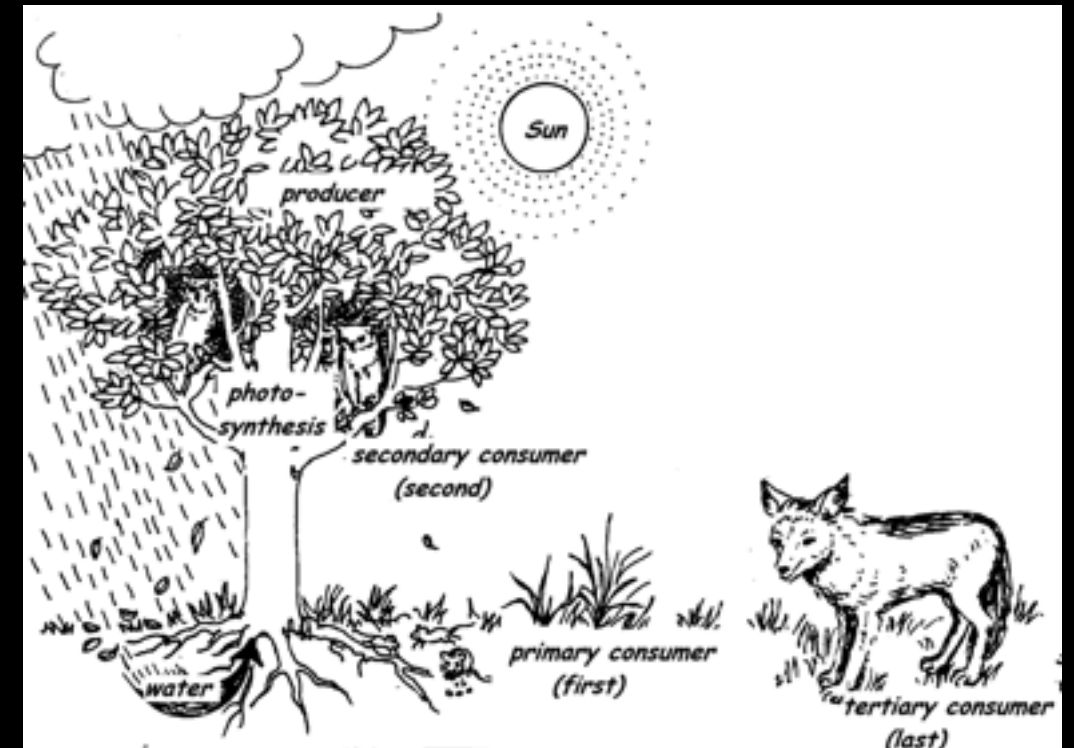
开源中最重要的

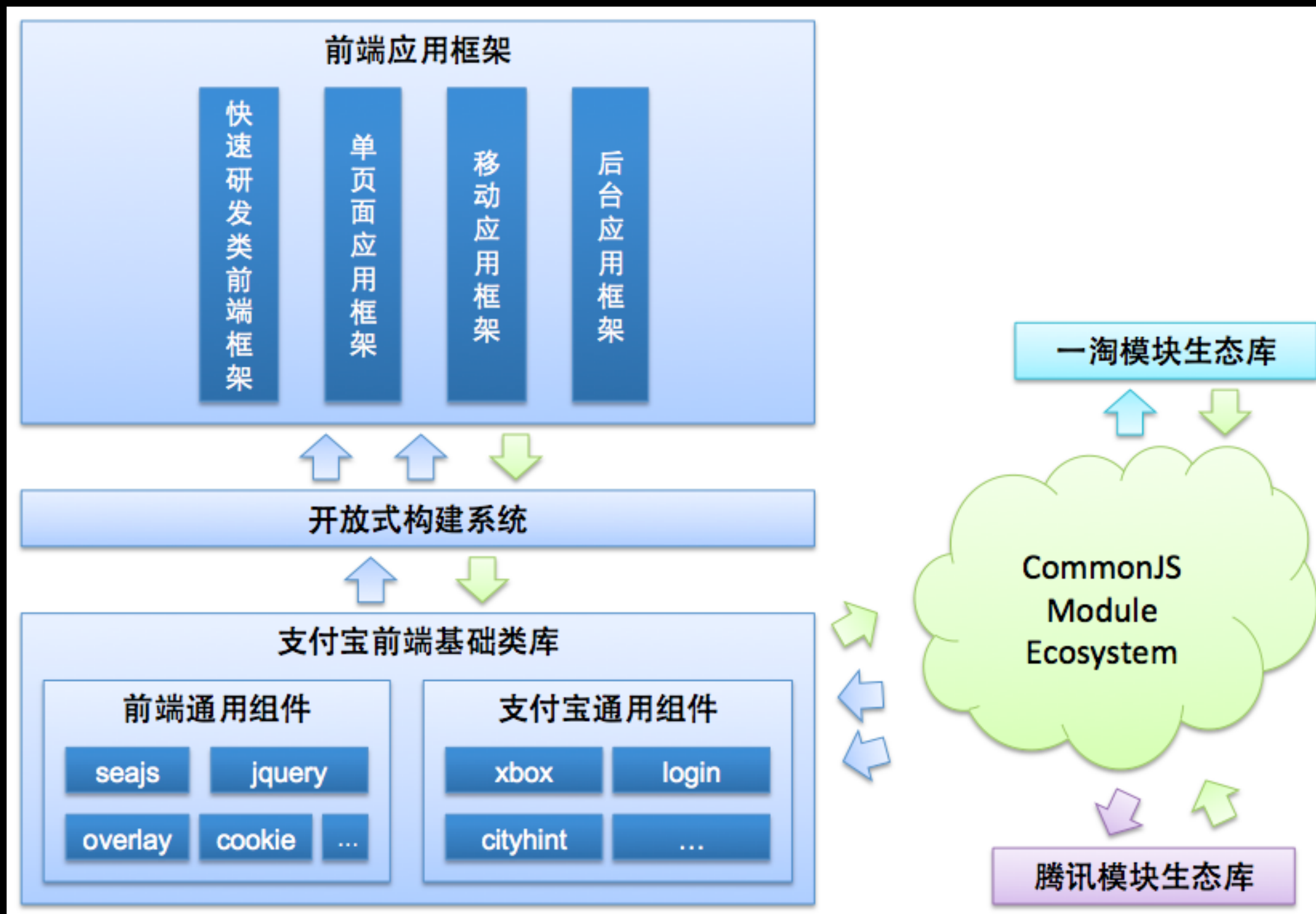
- 一个优秀、靠谱的想法
- 疯狂而持久的坚持

开源项目起步时，梦想都很丰满，但现实都很骨感。很多人等不到后天的太阳，经常离开于明天的晚上。

JavaScript 生态圈

- 马云：建立新商业文明
- 我们：构建 JavaScript 新生态圈

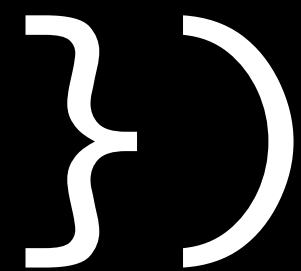




Arale 是一个开放、简单、易用的前端基础类库。



Questions?



seajs.org