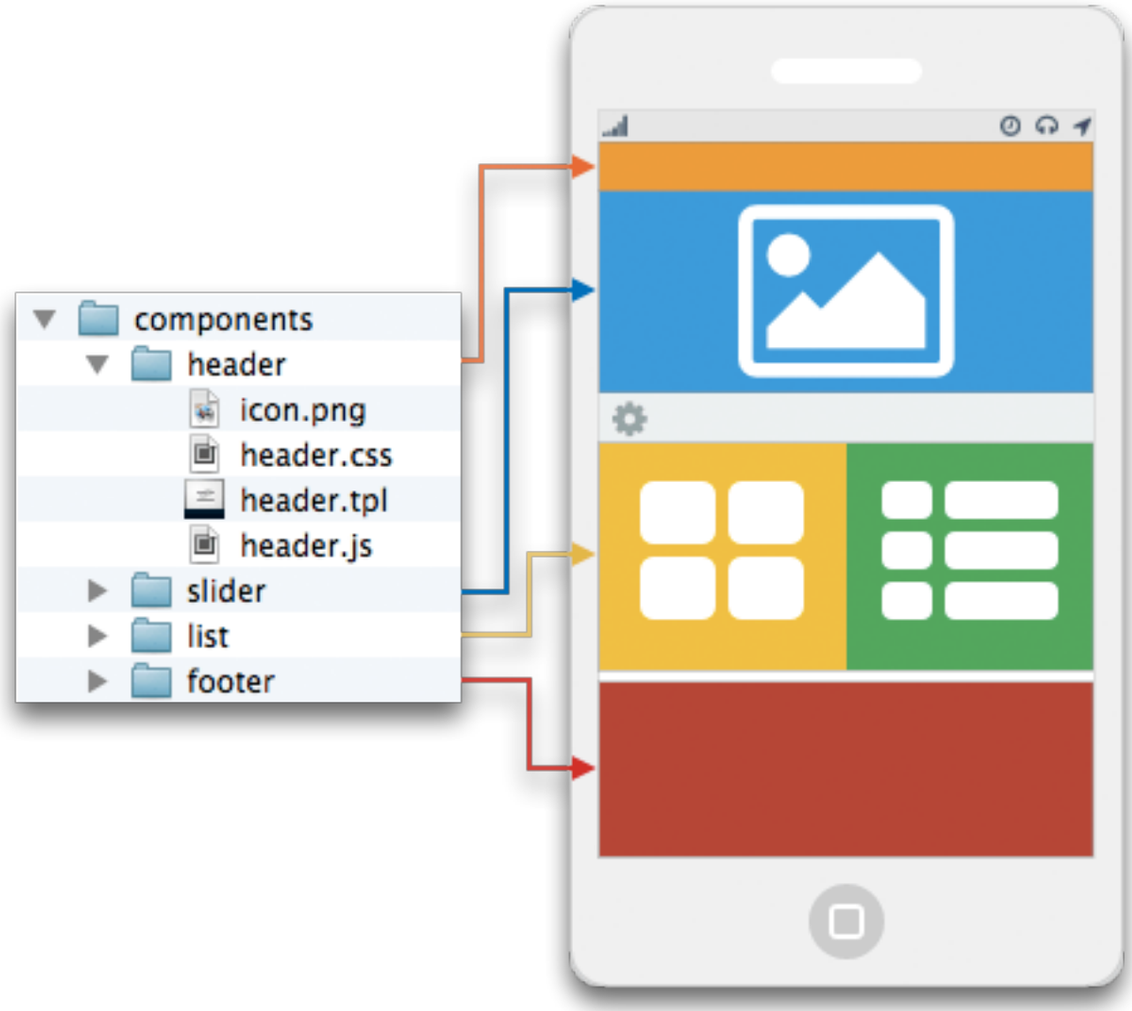
基于FIS打造WEBAPP模块化开发框架

“我们希望能像 搭积木 一样开发和维护系统，最终通过 组装模块 得到一个完整的应用。”

–工程师心声

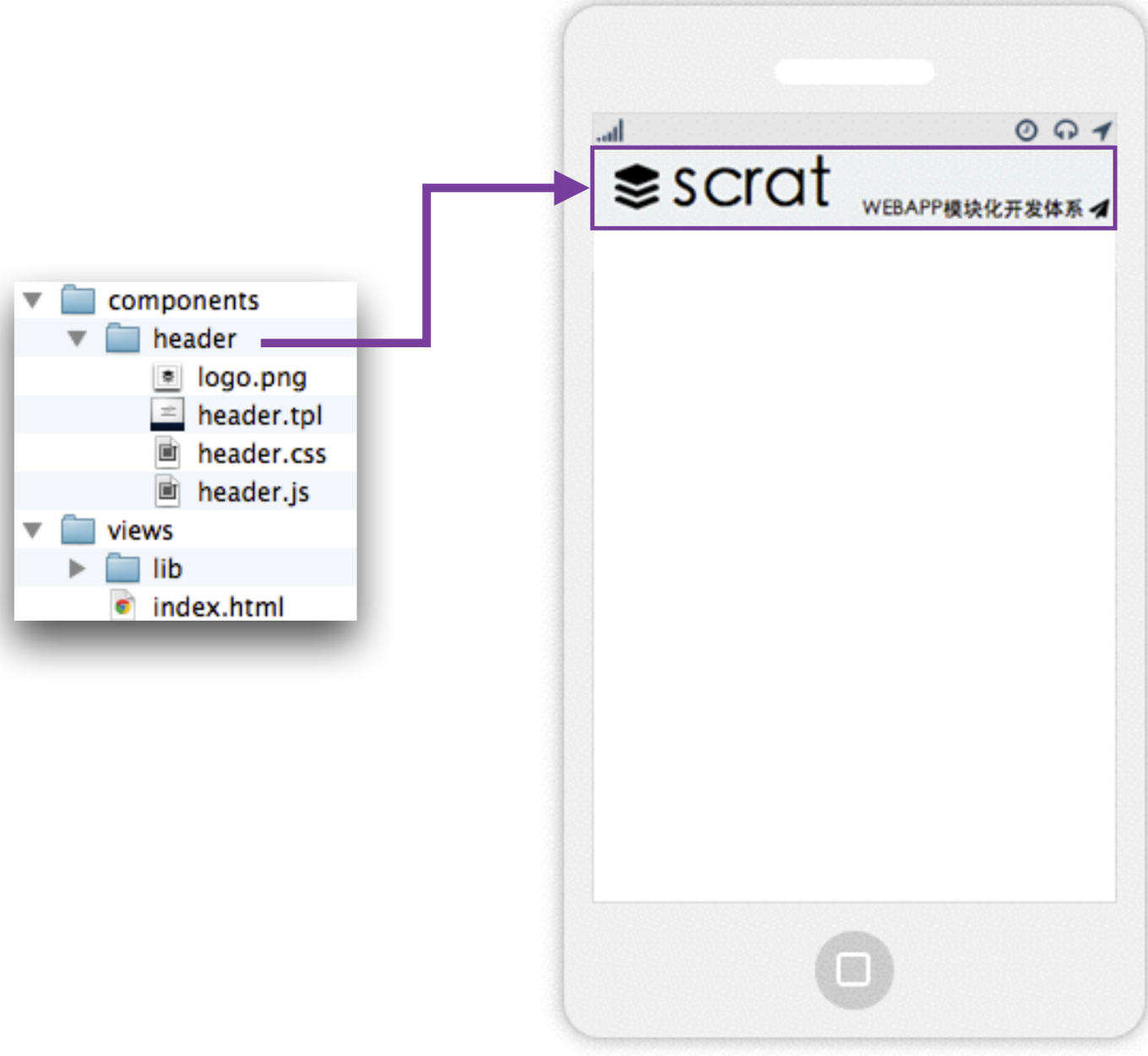
模块化开发



- 模块是可组合、可分解和更换的单元
- 模块具有一定的独立性
- 将模块所需的js、css、图片、模板维护在一起

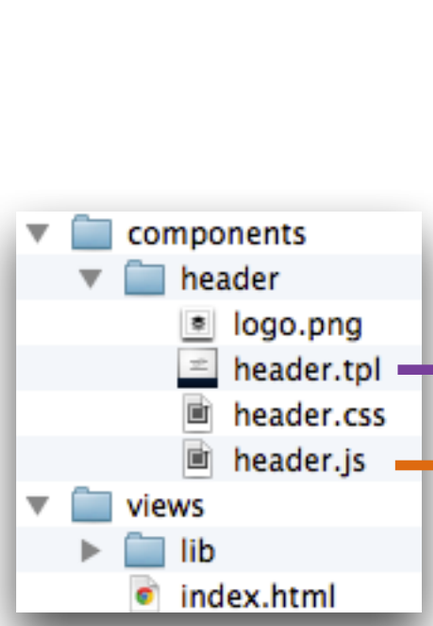
模块化开发

- 一个模块一个目录



模块化开发

- 一个模块一个目录
- 像写nodejs一样写js模块
- 将模板嵌入到js中使用



模板代码

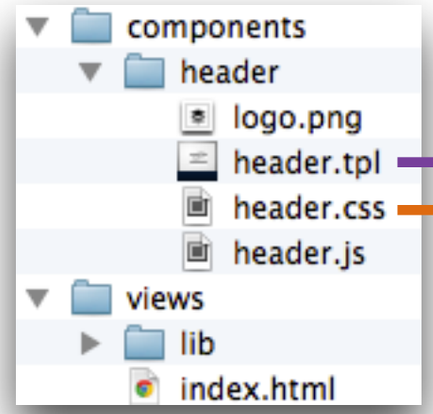
```
<div class="header">
    <h1 class="header-title">scrat</h1>
    <h3 class="header-subtitle">
        WEBAPP模块化开发体系 <i class="fa fa-paper-plane"></i>
    </h3>
</div>
```

JS模块代码

```
// 依赖zepto模块
var $ = require('zepto');

exports.render = function(selector){
    // 将模板内嵌到js中使用
    var tpl = __inline('header.tpl');
    $(selector).html(tpl);
};
```

模块化开发

- 一个模块一个目录
- 像写nodejs一样写js模块
- 将模板嵌入到js中
- css只关心模块内样式
- css也有依赖关系



模板代码

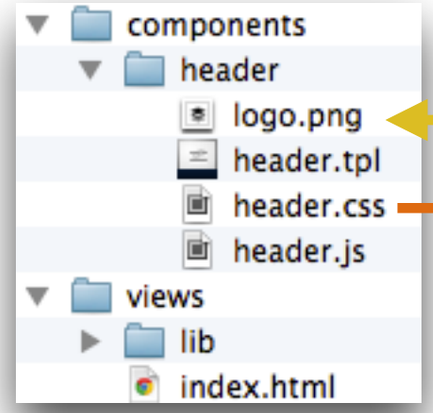
```
<div class="header">
    <h1 class="header-title">scrat</h1>
    <h3 class="header-subtitle">
        WEBAPP模块化开发体系 <i class="fa fa-paper-plane"></i>
    </h3>
</div>
```

CSS模块代码

```
/**
 * 依赖font-awesome模块
 *
 * @require font-awesome
 */

.header {
    position: relative;
    background: #ecf0f1;
    padding: 5px 10px;
}
.header-title {
    font-size: 40px;
    background: url(logo.png) no-repeat 0 center;
    padding-left: 45px;
}
.header-subtitle {
    position: absolute;
    right: 5px; bottom: 5px;
    font-size: 12px;
}
```

模块化开发

- 一个模块一个目录
- 像写nodejs一样写js模块
- 将模板嵌入到js中
- css只关心模块内样式
- css也有依赖关系
- 相对路径引用资源

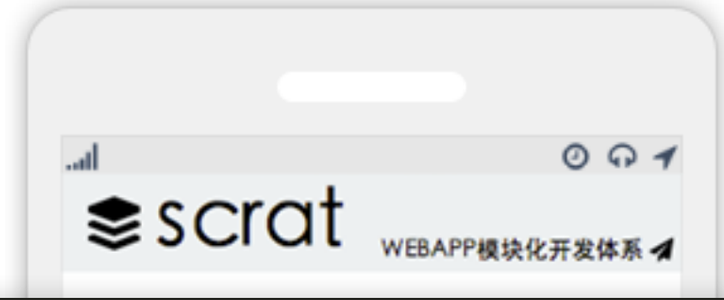


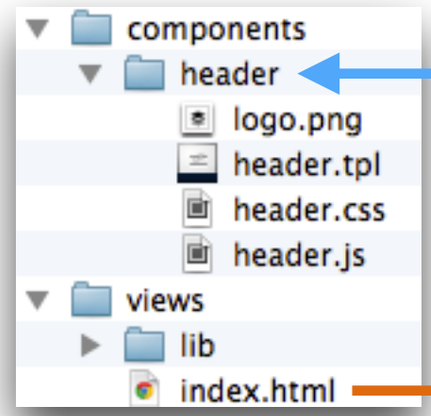
CSS模块代码

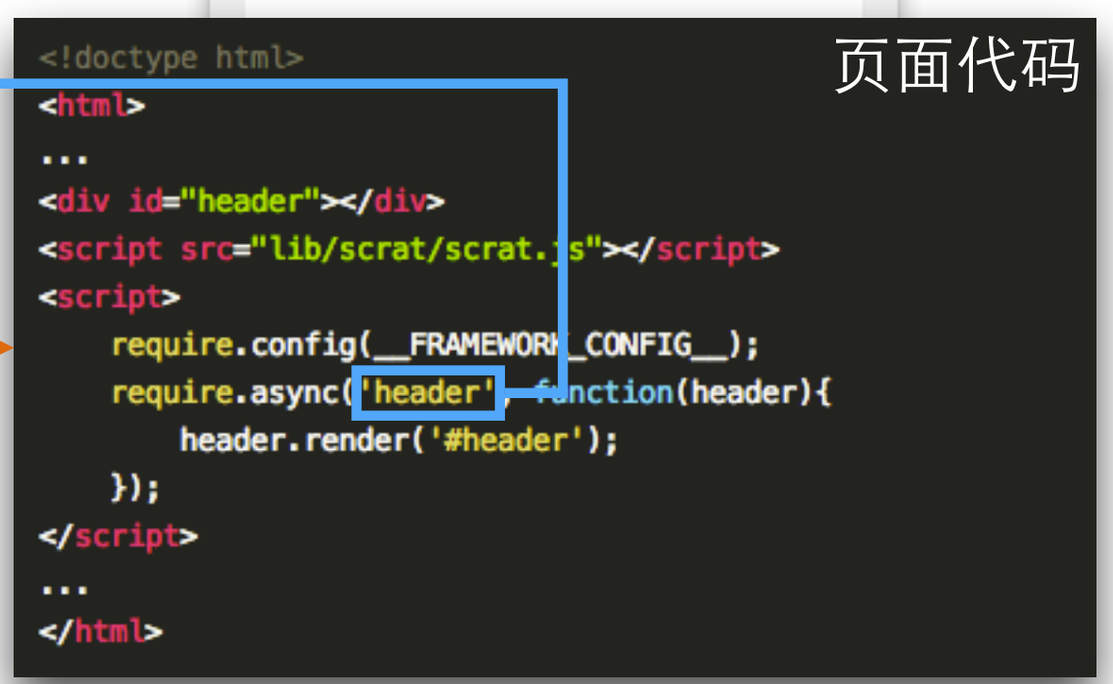
```
/**
 * 依赖font-awesome模块
 *
 * @require font-awesome
 */

.header {
    position: relative
    background: #ecf0f
    padding: 5px 10px;
}
.header-title {
    font-size: 40px;
    background: url(logo.png) no-repeat 0 center;
    padding-left: 45px;
}
.header-subtitle {
    position: absolute;
    right: 5px; bottom: 5px;
    font-size: 12px;
}
```

模块化开发

- 一个模块一个目录
- 像写nodejs一样写js模块
- 将模板嵌入到js中
- css只关心模块内样式
- css也有依赖关系
- 相对路径引用资源
- 引用模块即加载所有资源







模块化开发

- 一个模块一个目录
- 像写nodejs一样写js模块
- 将模板嵌入到js中
- css只关心模块内样式
- css也有依赖关系
- 相对路径引用资源
- 引用模块即加载所有资源

保证模块的：

独立性
可组装性
可更换性

模块化开发

- 一个模块一个目录
- 像写nodejs一样写js模块
- 将模板嵌入到js中
- css只关心模块内样式
- css也有依赖关系
- 相对路径引用资源
- 引用模块即加载所有资源

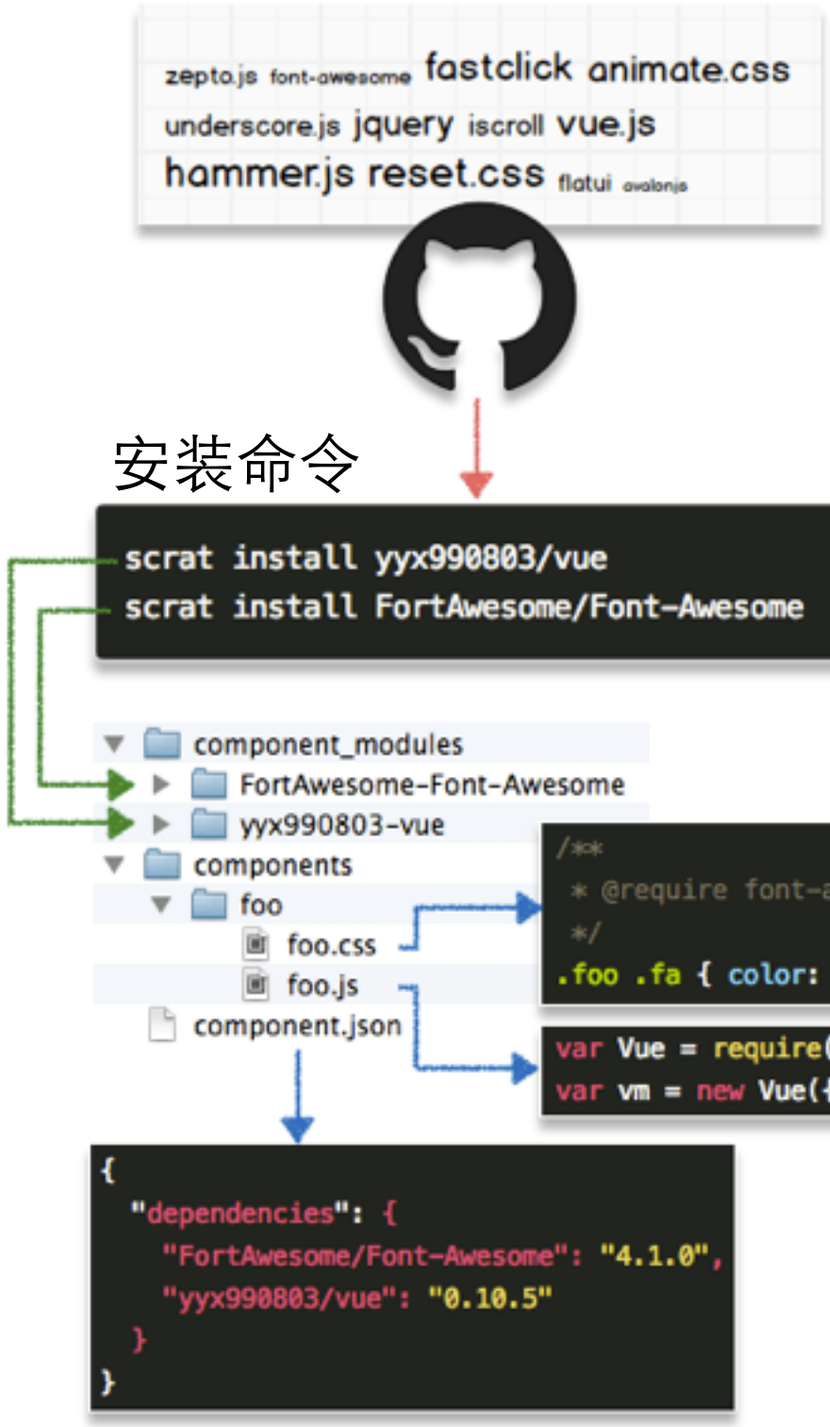
以最自然的方式写码

“我们希望每次研发新产品不是从零开始，不同团队不同项目之间能有可复用的模块沉淀下来。”

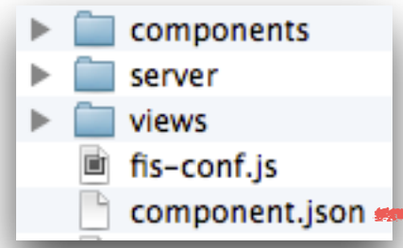
–工程师心声

模块生态

- 每个项目有工程模块和生态模块
- 生态模块基于 component 规范开发，部署到Github上
- 可以通过命令行工具将Github上的模块安装到工程中使用



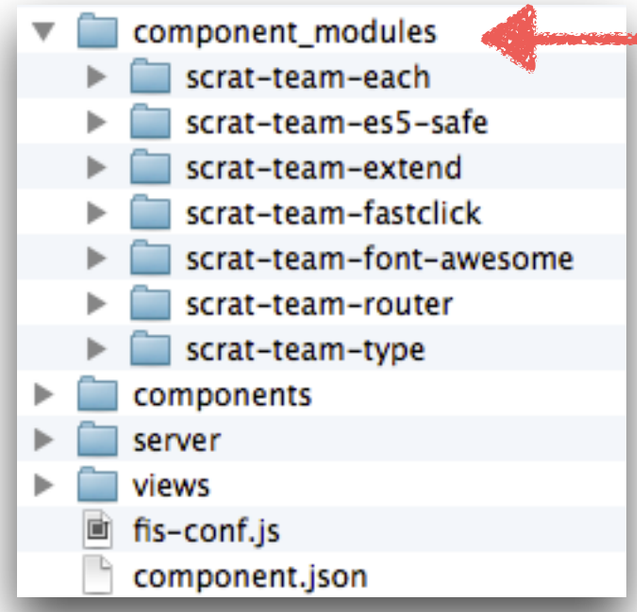
项目初始



声明生态模块依赖

```
{
  "name": "scrat-site",
  "version": "0.1.0",
  "dependencies": {
    "scrat-team/each": "0.1.0",
    "scrat-team/fastclick": "1.0.2",
    "scrat-team/router": "0.1.0",
    "scrat-team/es5-safe": "0.1.0",
    "scrat-team/font-awesome": "4.1.0"
  }
}
```

安装生态模块



命令行安装

```
→ scrat-site scrat install
install preparing...
install installing scrat-team/type@0.1.0...
install installing scrat-team/each@0.1.0...
install installing scrat-team/fastclick@1.0.2...
install installing scrat-team/extend@0.1.0...
install installing scrat-team/router@0.1.0...
install installing scrat-team/es5-safe@0.1.0...
install installing scrat-team/font-awesome@4.1.0...
install finished
→ scrat-site
```

初始项目直接基于已有生态模块，不用从零开始

世界杯项目中有 **26 个(约30%)** 组件来自NBA项目积累

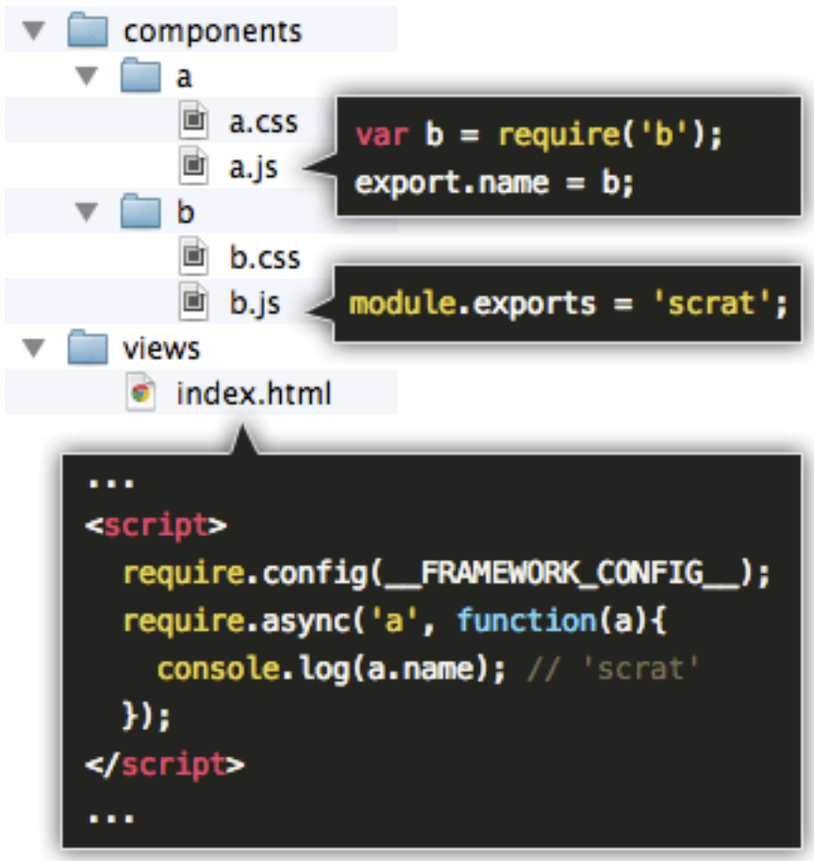
内页项目中 **31** 个频道页复用了 **21个(约 50%)** 组件

“我们希望写码时不用关心**性能优化**，但上线后资源**请求可以合并**，并且**按需加载**。”

–工程师心声

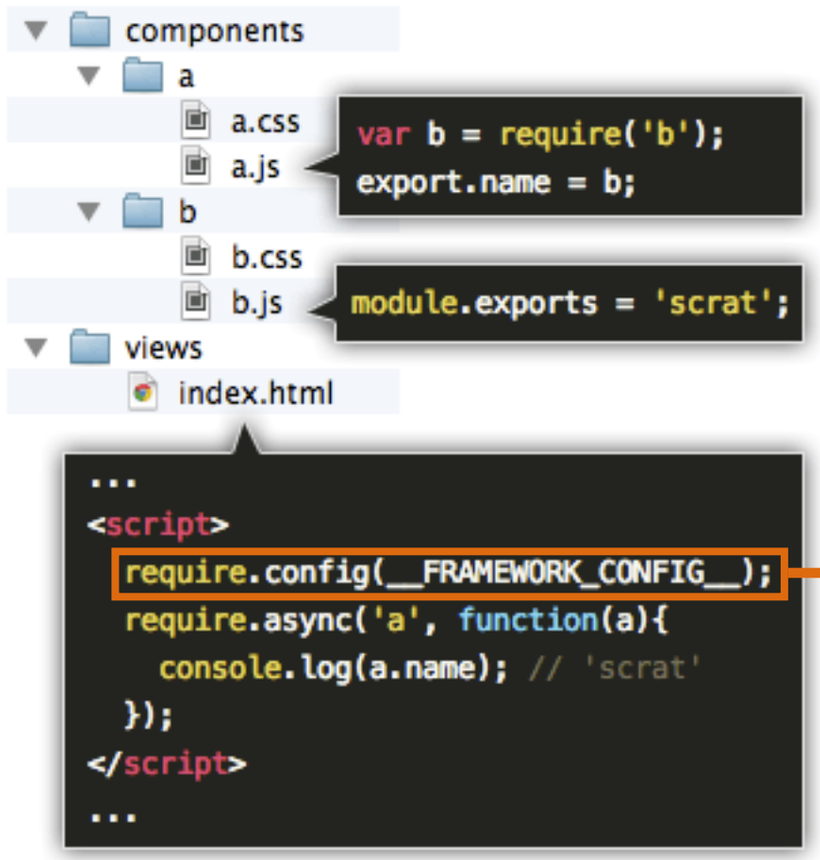
性能优化

1. 模块声明依赖关系



性能优化

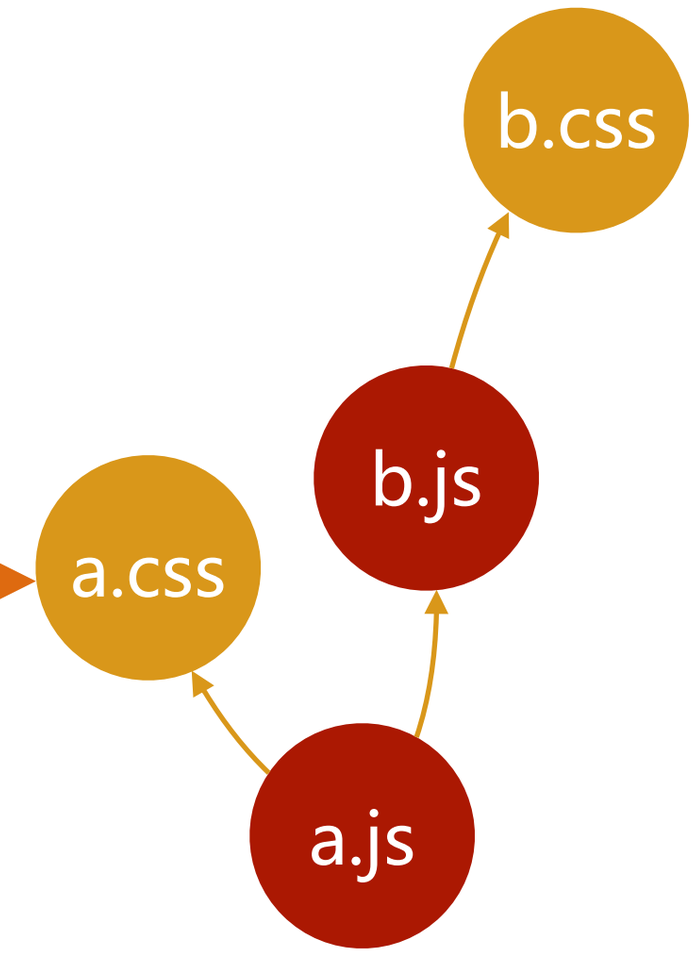
1. 模块声明依赖关系
2. 工具分析依赖关系



性能优化

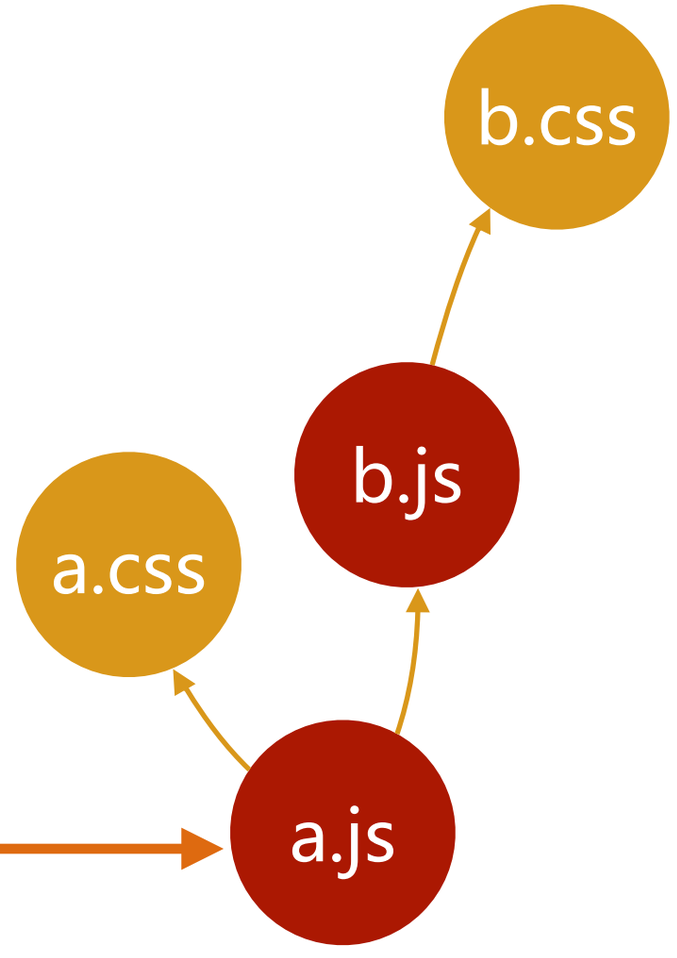
1. 模块声明依赖关系
2. 工具分析依赖关系
3. 框架获得依赖关系

```
...
<script>
  require.config({
      "cache": false,
      "version": "1.0.0",
      "name": "proj",
      "combo": true,
      "urlPattern": "/c/%s",
      "comboPattern": "/??%s",
      "hash": "a9168fe",
      "alias": {
          "a": "proj/1.0.0/a/a.js",
          "b": "proj/1.0.0/b/b.js"
      },
      "deps": {
          "proj/1.0.0/a/a.js": [
              "b",
              "proj/1.0.0/a/a.css"
          ],
          "proj/1.0.0/b/b.js": [
              "proj/1.0.0/b/b.css"
          ]
      }
  });
  require.async('a', function(a){
    console.log(a.name); // 'scrat
  });
</script>
...
```



性能优化

1. 模块声明依赖关系
2. 工具分析依赖关系
3. 框架获得依赖关系
4. 按需加载所有依赖
5. 借助combo服务合并请求

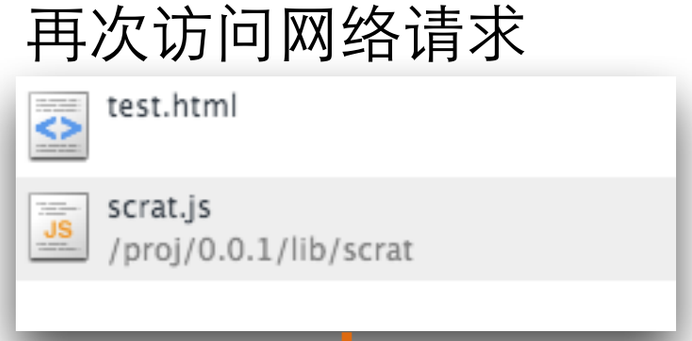



```
...
<script>
    require.config({...});
    require.async('a', function(a){
        // do something
    });
</script>
...
```

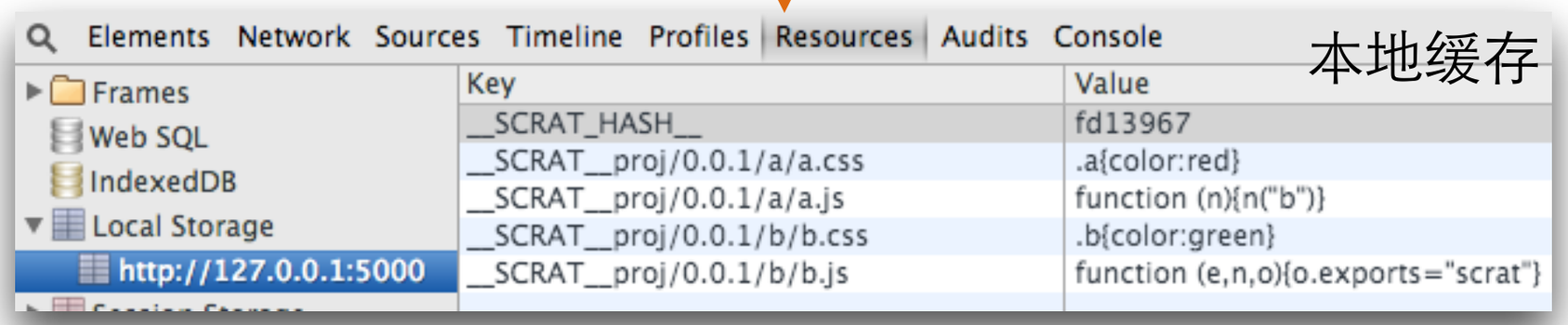
性能优化

1. 模块声明依赖关系
2. 工具分析依赖关系
3. 框架获得依赖关系
4. 按需加载所有依赖
5. 借助combo服务合并请求
6. localstorage缓存资源

再次访问网络请求



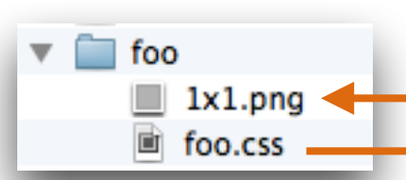
本地缓存



性能优化

1. 模块声明依赖关系
2. 工具分析依赖关系
3. 框架获得依赖关系
4. 按需加载所有依赖
5. 借助combo服务合并请求
6. localstorage缓存资源
7. 将小图片内嵌到css中

源代码




```
.foo {
    background: url('1x1.png?__inline');
}
```

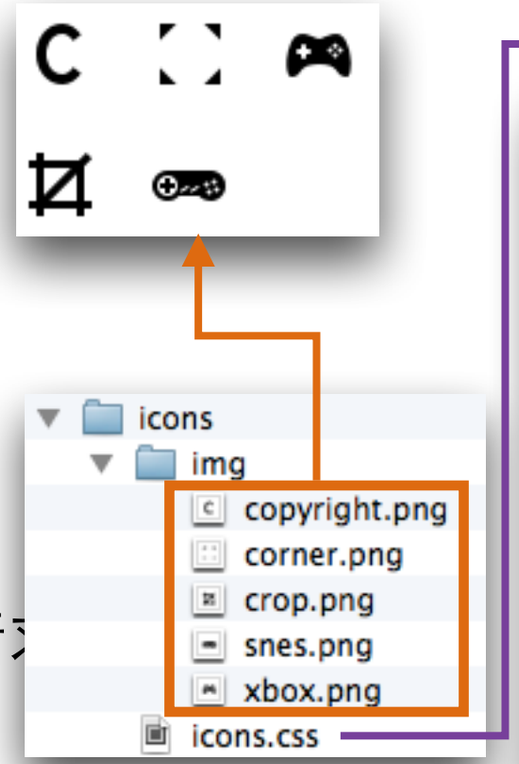
经工具构建后的代码

```
.foo {
    background:url('
SUhEUgAAAAEAAAABCAIAAACQd1PeAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZS
BJbWFnZVJlYWR5ccllPAAAAAxJREFUCNdj2LRpEwAEMAIXLOcFRwAAAABJ
RU5ErkJggg==');
}
```

性能优化

1. 模块声明依赖关系
2. 工具分析依赖关系
3. 框架获得依赖关系
4. 按需加载所有依赖
5. 借助combo服务合并请求
6. localstorage缓存资源
7. 将小图片内嵌到css中
8. 自动css雪碧图

经工具构建后的雪碧图



源代码

```
.icon {
    width: 24px;
    height: 24px;
    display: inline-block;
}
.icon-copyright {
    background: url('img/copyright.png?__sprite ) no-repeat 0 0;
    background-size: 24px 24px;
}
.icon-corner {
    background: url('img/corner.png?__sprite') no-repeat 0 0;
    background-size: 24px 24px;
}
.icon-crop {
    background: url('img/crop.png?__sprite') no-repeat 0 0;
    background-size: 24px 24px;
}
.icon-snes {
    background: url('img/snes.png?__sprite') no-repeat 0 0;
    background-size: 24px 24px;
}
.icon-xbox {
    background: url('img/xbox.png?__sprite') no-repeat 0 0;
    background-size: 24px 24px;
}
```

性能优化

1. 模块声明依赖关系
2. 工具分析依赖关系
3. 框架获得依赖关系
4. 按需加载所有依赖
5. 借助combo服务合并请求
6. localstorage缓存资源
7. 将小图片内嵌到css中
8. 自动css雪碧图

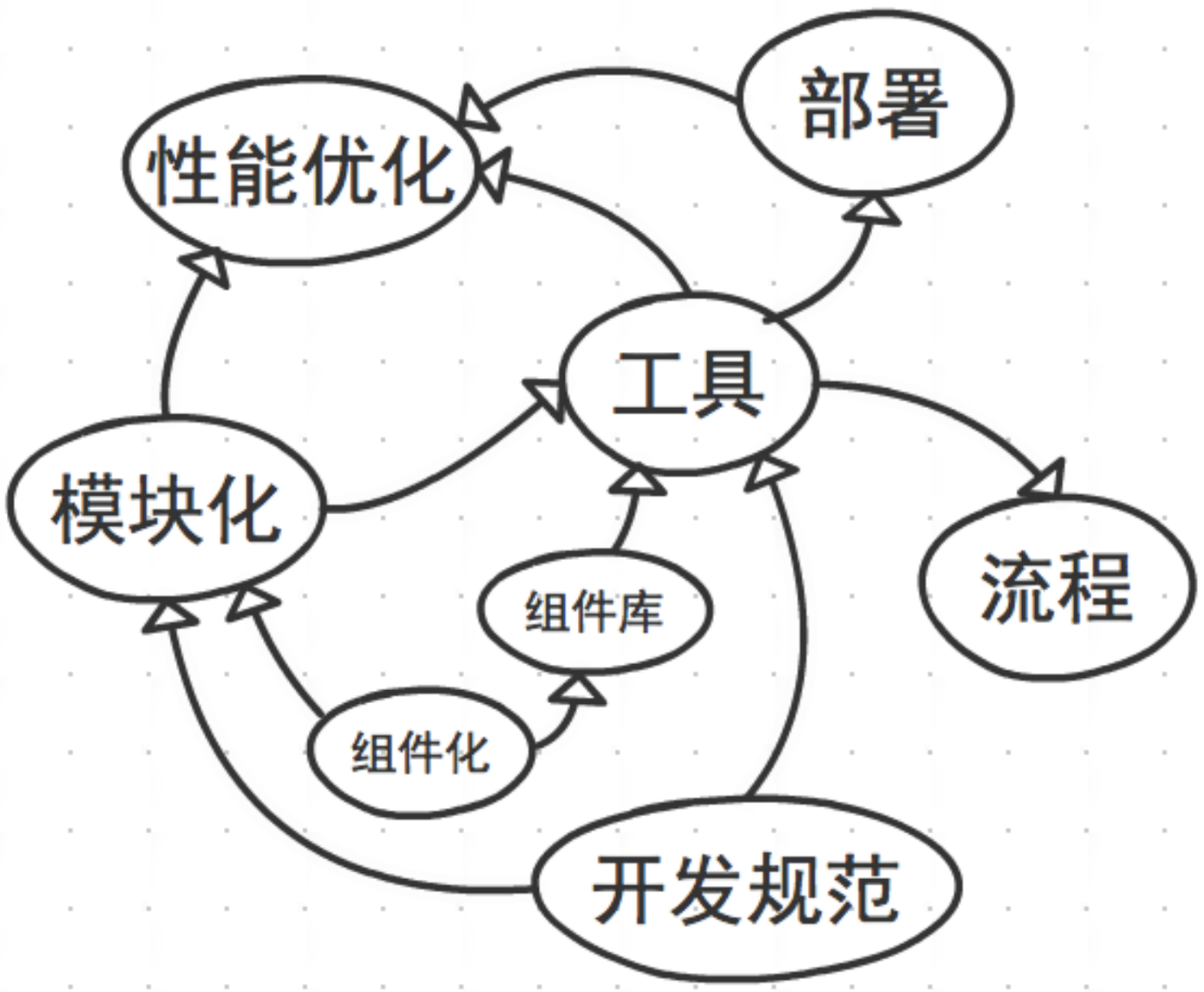
以最自然的方式写码
以最透明的形式优化

“我们希望能在自己的电脑上用最喜欢的IDE写码，并且方便移动设备开发调试。”

–工程师心声

开发利器

- 内置服务器本地开发预览
- 代理线上数据接口
- 文件监听实时构建
- 多设备自动刷新



真正的前端工程

这就是前端工程师团队的心声

项目地址：https://github.com/scrat-team/scrat
设计过程：https://github.com/fouber/blog/issues/2