

# 单元测试搭建

- [单元测试搭建](#)
  - [1. 基础概念](#)
  - [2. 选型调研](#)
    - [2.1 基础测试框架对比分析](#)
    - [2.2 断言库对比分析](#)
  - [3. 项目demo](#)
    - [3.1 Mocha + Chai 测试demo搭建](#)
    - [3.2 思考优化](#)
  - [4. 自动化测试搭建](#)
    - [4.2 基础环境配置](#)
    - [4.2 目录结构](#)
    - [4.3 运行测试](#)
    - [4.4 存在问题](#)

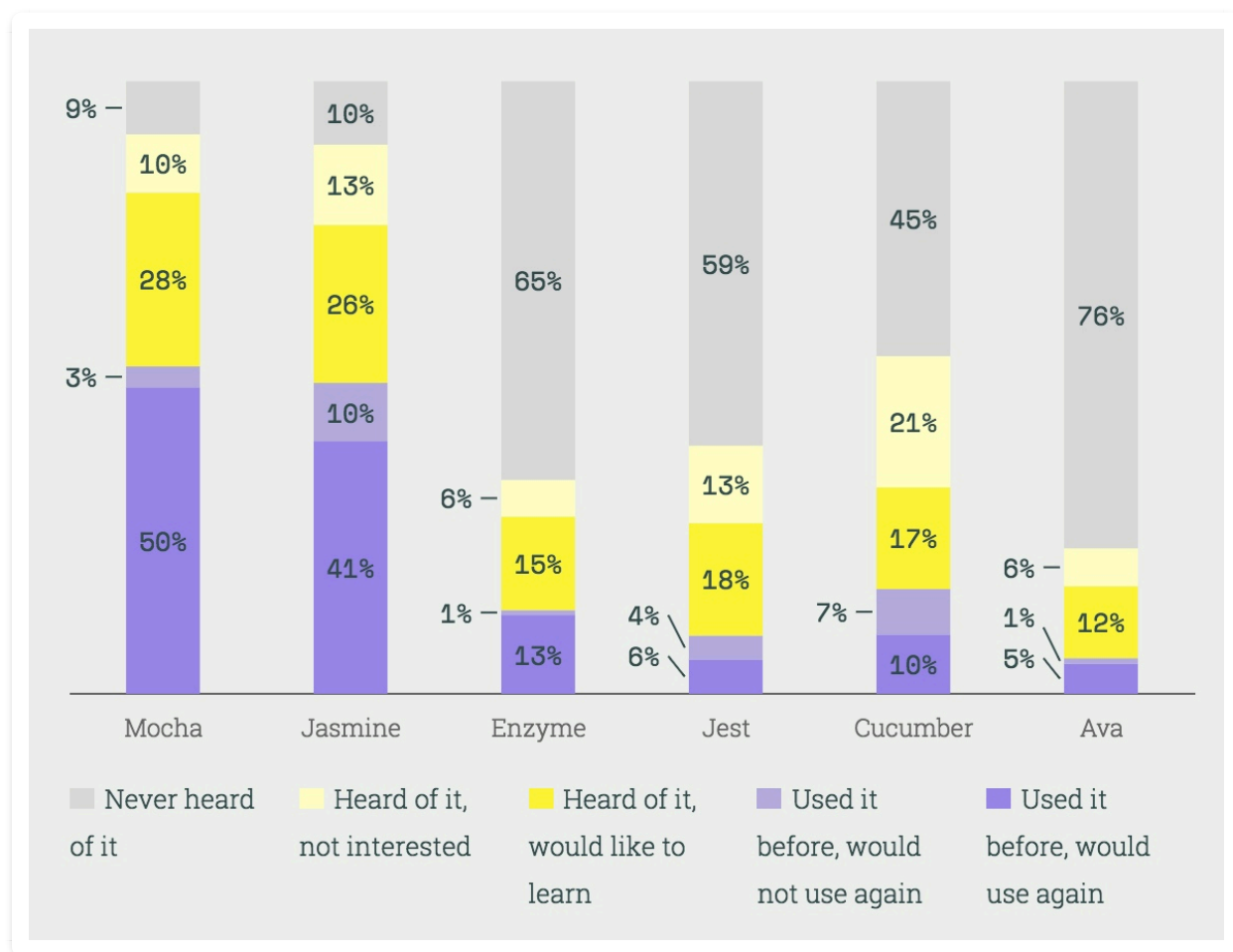
## 1. 基础概念

概念	解释
TDD	测试驱动开发，在开发功能代码之前，先编写单元测试用例代码，测试代码确定需要编写什么产品代码
BDD	行为驱动开发，使用一种通用模板来描述行为
断言	捕捉代码中的假设

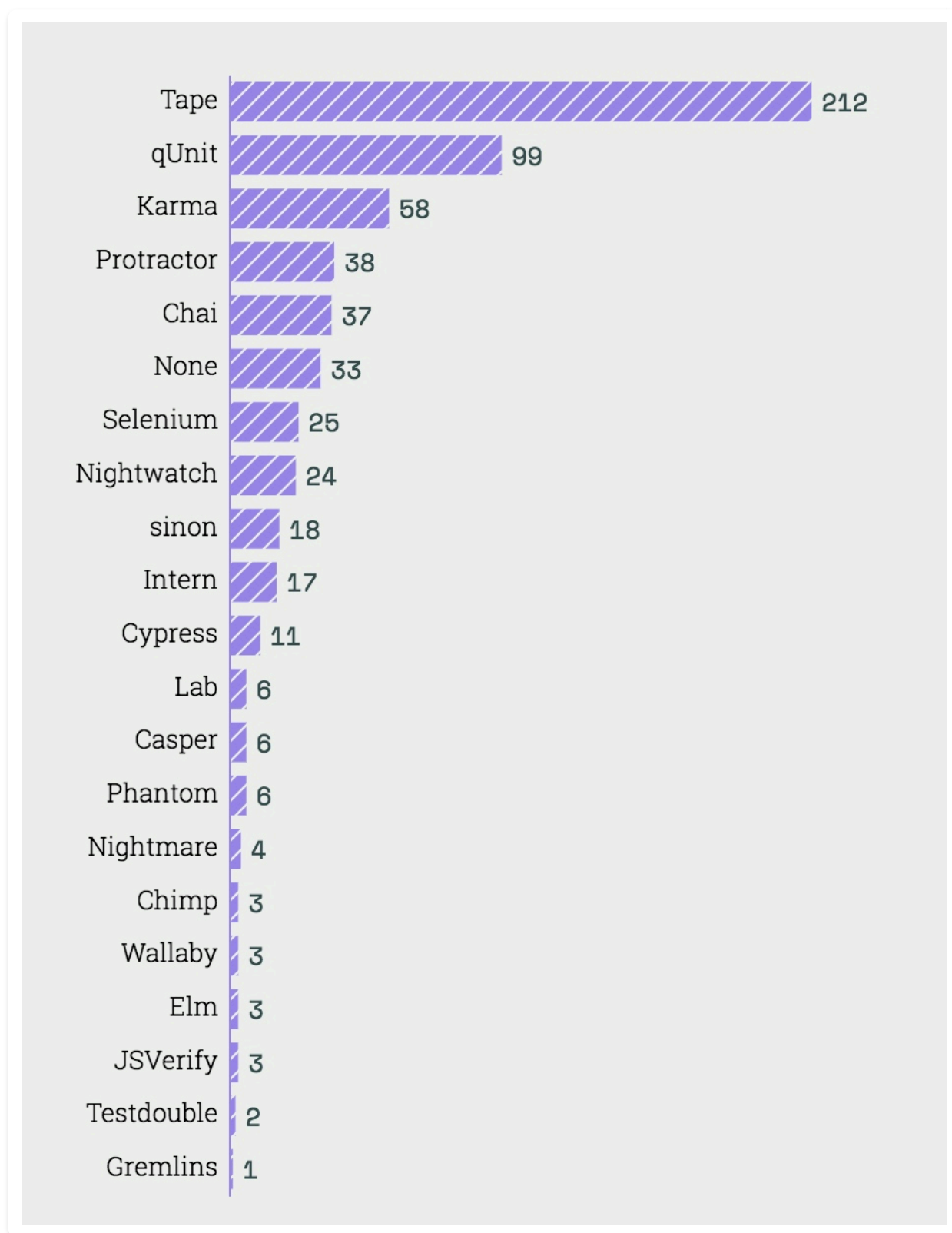
## 2. 选型调研

### 2.1 基础测试框架对比分析

[stateofjs Testing Frameworks](#)官网调查的热门框架结论：



其他框架：



这是官网列出的调研结论，各个测试框架之间的，对测试环节的关注点会因为侧重点不同，而有所差异，可以划分为几类：

类型	示例
基础测试框架	<a href="#">Mocha</a> 、 <a href="#">Ava</a>
断言库和功能库	<a href="#">Chai</a> 、 <a href="#">Sinon</a>

全功能测试框架	<a href="#">Jasmine</a>
集成环境框架	<a href="#">Karma</a>

在选择框架时，主要对比了框架的稳定性、使用口碑，官方文档来对比。社区方面，可作为参照条件，但框架使用过程中遇到问题，通常解决方案会优先考虑查阅官方文档，再到参考github相关项目demo，再到社区查阅。这里，优先考虑官方文档的使用难以程度。

这里先选择目前口碑最佳和最具发展潜力的三个基础框架（参考官方注释说明）对比，Jasmine相比功能更齐全，但本身更多作为基础框架，使用其他第三方库扩展，这里可用来一起比较。

框架	Mocha	Jasmine	Ava
特点总结	使用人数目前最高，稳定	使用人数较高多，稳定	新兴的框架，使用人数有上升的趋势，对ES6支持最好，有部分ES7概念引入
官方文档使用	文档脉络更贴近于快速入门类型，从0搭建到框架提供的每种功能，循序渐进式的介绍，查阅容易	文档组织形式与习惯不一致，文档查阅上稍有难度	快速入门式的循序渐进介绍，文档查阅容易
实际安装	安装容易，按照官网介绍安装即可	安装容易，按照官网介绍安装即可	安装容易，按照官网介绍安装即可

综合考虑，首选Mocha框架作为基础框架，Ava保持持续跟进关注，Jasmine可以做简单了解。

## 2.2 断言库对比分析

[better-assert](#) 【C-style TDD 断言库】

👁 Watch ▼

5

★ Star

226

🔗 Fork

21

[should.js](#) 【BDD 风格断言库】

👁 Watch ▼

38

★ Star

1,485

🔗 Fork

99

[expect.js](#) 【追求极简的 BDD 风格断言库，基于 should.js 简化】

👁 Watch ▼

200

★ Star

1,746

🔗 Fork

182

[chai.js](#) 【BDD/TDD 双模，支持 should / expect / assert 三种风格的断言库，强大插件机制】

👁 Watch ▼

114

★ Star

4,420

🔗 Fork

411

根据github上各断言库使用数据，chai在使用人数与维护人数方面，更有优势，且功能更强大，可以优先考虑。

### 3. 项目demo

#### 3.1 Mocha + Chai 测试demo搭建

这里以image-upload.js为实际项目搭建自动化测试项目。遇到以下问题及解决方案：

遇到问题	问题解决
项目文件为ES6语法编写，需要babel转义	通过发布脚本编译参数赋值指向npm环境中的babel执行文件
无界面运行	

document对象报错问题	发布命令指定编译运行环境，这里选择无界面浏览器PhantomJS
项目文件使用基于Vue编写，需要解决Vue npm包默认指向问题	1. 推荐使用项目中的引入的vue.js而不是node_modules中的vue，要保证版本的一致性，且node_modules中默认使用的vue指向为不含有模板引擎的vue.common.js； 2. 也可以通过webpake配置修改默认路径（小佑对此问题提出的解决方案），考虑到暂未引入webpake，在实际操作中直接import node_modules中vue模块的指定文件；
上传功能模拟网络请求问题	引入第三方库 <a href="#">Sinon</a> 解决

## 3.2 思考优化

在测试demo搭建过程中，思考的优化点：

- 对于需要编译的文件，需要配置相同的参数，若项目中还有多种需要编译的文件(比如vue单文件组件、less预编译等)，则package.json文件中需要配置更多的并行执行命令，发布命令配置复杂，思考是否能够通过类似webpake、fis这类构建工具来优化配置。
- 测试报告产出文本化。
- 是否同时可以支持UI界面展示和无界面展示。

针对这些思考，尤其是配置问题，是否有更多的集成工具库来简化搭建。这里调研了[Vue官方推荐](#)的测试集成环境框架[Karma](#)，参考了[ELementUI](#)单元测试的实现。选定了一套基于Karma集成的Mocha + Chai + Sinon的搭建方案。

## 4. 自动化测试搭建

选用Karma + Mocha + Chai + Sinon的方案，具体实现如下：

### 4.2 基础环境配置

项目的基础依赖包已经在package.json文件配置完毕，直接执行 `npm install` 自动全部安装即可。

### 4.2 目录结构

```
├── coverage
├── karma.conf.js
├── lib
│   ├── jquery
│   └── vue
├── node_modules
├── package.json
├── test
│   ├── unit
│   │   └── image-upload.spec.js
│   └── util.js
└── widget
    └── image-upload.js
```

`coverage/` 目录下主要存放测试报告产出文档（HTML文件），`karma.conf.js` 文件为Karma相关配置，`lib/` 目录下主要用于存放组件使用的js框架或库（项目用引入了Vue2和JQuery），`test/unit/` 目录下存放测试脚本，`test/` 目录下的可以定义 `util.js` 等文件来管理测试脚本公用的工具方法。`widget/` 目录下为组件文件。

Karma配置可参考 `karma.conf.js` 的实现及注释。测试脚本命名方式要求为目标文件名 + `.spec`，如 `image-upload.js` 对应的测试脚本为 `image-upload.spec.js`。若文件标识后缀名或文件目录有调整，请对应的修改的 `karma.conf.js` 文件的 `preprocessors` 配置。

## 4.3 运行测试

执行命令及结果：

```
npm run test
```

```

> karma-test@1.0.0 test /Users/huangyi/work/fe-auto-test/mocha/karma-test
> karma start ./karma.conf.js --single-run

18 08 2017 13:50:41.176:INFO [karma]: Karma v1.7.0 server started at http://0.0.0.0:9876/
18 08 2017 13:50:41.179:INFO [launcher]: Launching browser Chrome with unlimited concurrency
18 08 2017 13:50:41.209:INFO [launcher]: Starting browser Chrome
18 08 2017 13:50:43.659:INFO [Chrome 60.0.3112 (Mac OS X 10.12.5)]: Connected on socket noRjm2B5vX04Y4vAAAAA with id 85486938
18 08 2017 13:50:44.325:WARN [web-server]: 404: /undefined
INFO LOG: 'Download the Vue Devtools extension for a better development experience:
https://github.com/vuejs/vue-devtools'
INFO LOG: 'You are running Vue in development mode.
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html'

  图片上传组件测试
    Vue实例创建
      ✓ 创建成功
    ajax测试
      ✓ 请求成功, 返回成功信息
      ✓ 请求失败
      ✓ 请求成功, 返回后端报错信息
    上传功能测试
      ✓ 上传成功处理
18 08 2017 13:50:44.461:WARN [web-server]: 404: /undefined
      ✓ 上传失败处理

Chrome 60.0.3112 (Mac OS X 10.12.5): Executed 6 of 6 SUCCESS (0.3 secs / 0.259 secs)
TOTAL: 6 SUCCESS

===== Coverage summary =====
Statements : 100% ( 0/0 )
Branches   : 100% ( 0/0 )
Functions  : 100% ( 0/0 )
Lines      : 100% ( 0/0 )

```

在终端打印测试过程及结果的过程中，会调起配置的默认浏览器（项目制定chrome为默认浏览器），在浏览器端可以看到自动测试过程的UI变化及交互。

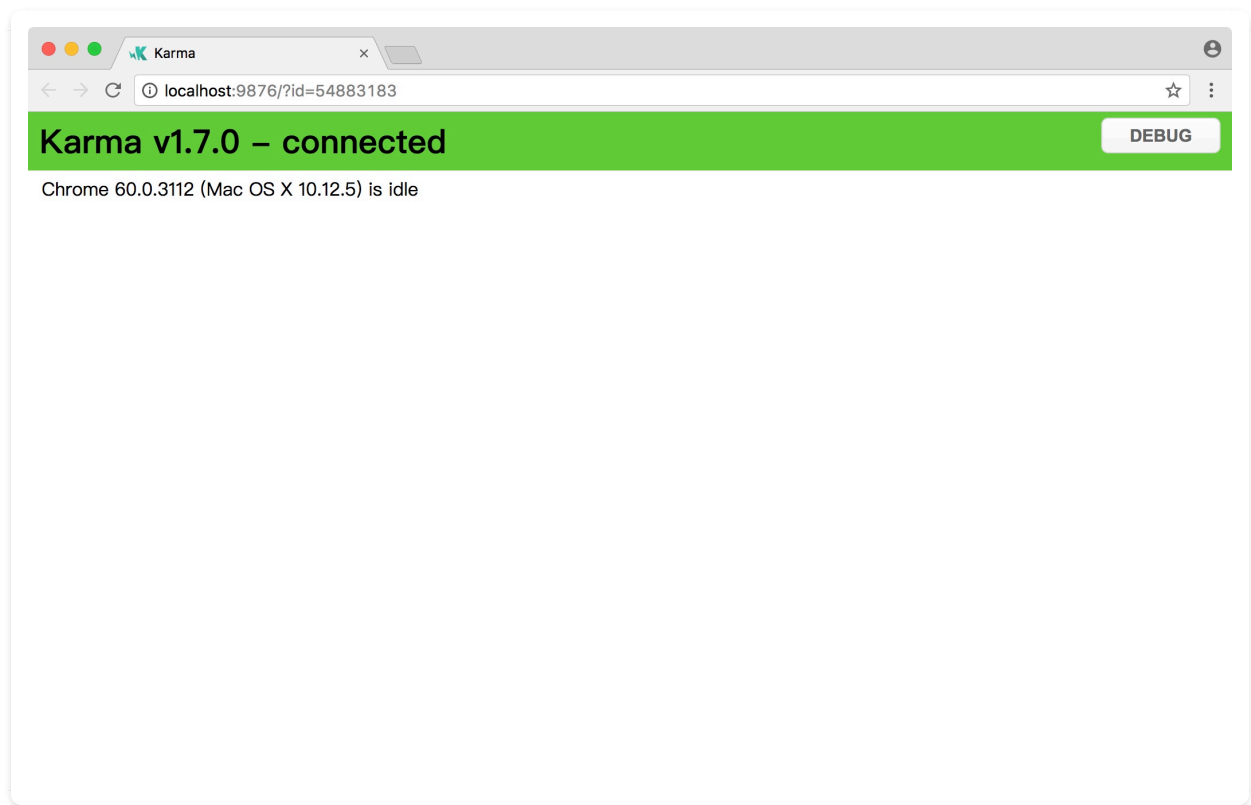
在开发测试脚本的过程中，也可以实时监听调试：

```
npm run debug
```

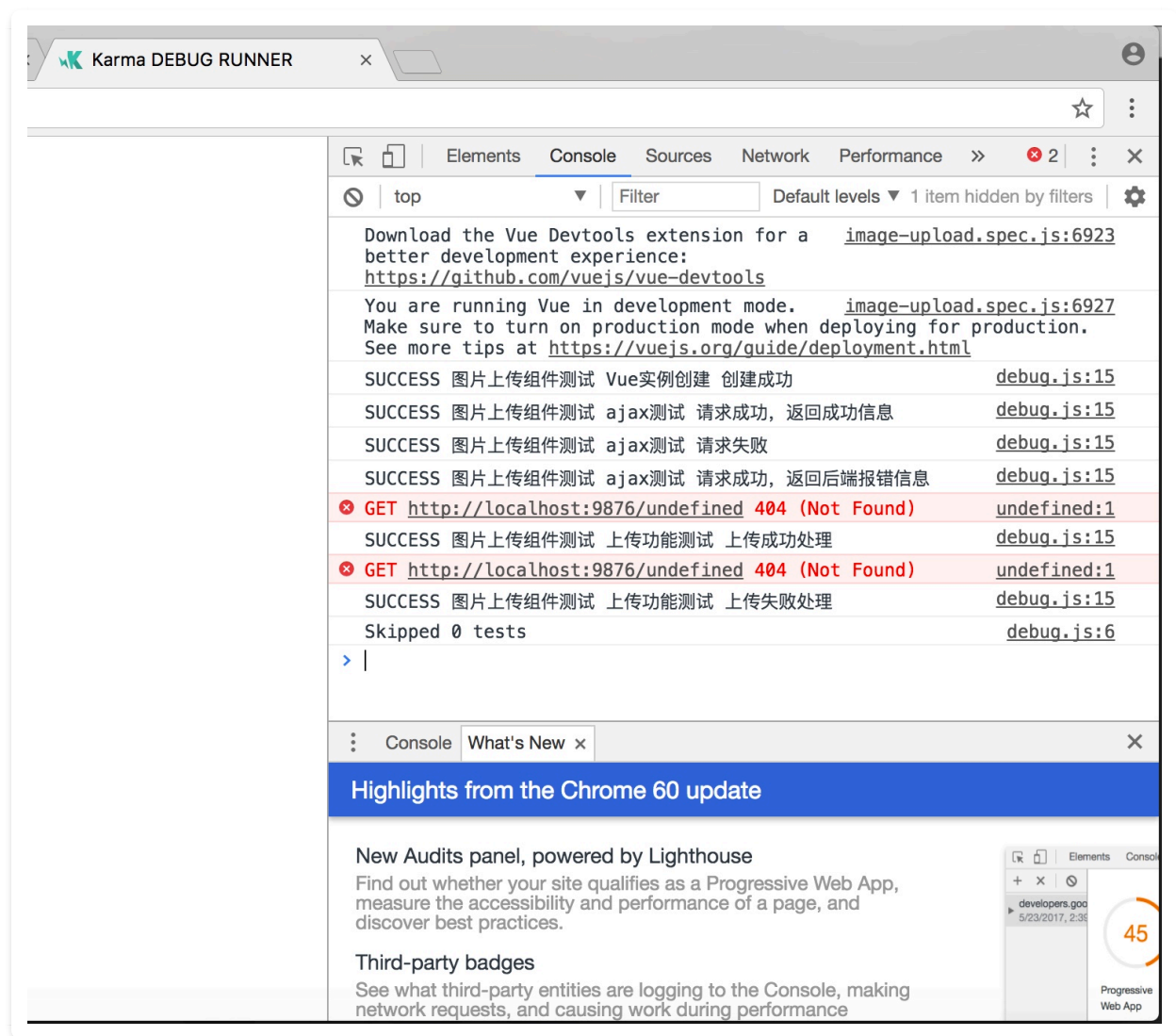
则调起默认浏览器访问

`localhost+指定port(默认先占用9876，若端口被占用则另外选用)`，在打开浏览器中可以实时调试测试脚本，如下：





点击debug按钮进入调试界面



## 4.4 存在问题

目前存在测试报告Coverage summary显示结果数据统计不正确的问题, 暂未解决。

