**mocha测试**

1/ 全局安装mocha环境 npm install --global mocha

package.json的基本配置{ "name": "mocha-demos", "version": "1.0.0", "description": "", "main": "index.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "keywords": [ "testing", "Mocha" ], "author": "", "license": "MIT", "devDependencies": { "babel-core": "~6.2.1", "babel-preset-es2015": "~6.1.18", "chai": "~3.4.1", "mocha": "~2.3.4", "mochawesome": "~1.2.1" }, "dependencies": { "node-fetch": "~1.3.3", "superagent": "~1.4.0" } }

安装依赖 npm i

测试单元的编写-测试单元函数模块暴露 module.exports 文件名eg：index.js

测试脚本的编写－－文件名字 eg：index.test.js

测试脚本--接收require要测试的单元，引入断言库chai依赖的expect断言风格

describe块称为"测试套件"（test suite），表示一组相关的测试。它是一个函数，第一个参数是测试套件的名称（"加法函数的测试"），第二个参数是一个实际执行的函数。

it块称为"测试用例"（test case），表示一个单独的测试，是测试的最小单位。它也是一个函数，第一个参数是测试用例的名称（"1 加 1 应该等于 2"），第二个参数是一个实际执行的函数。

头部是expect方法，尾部是断言方法，比如equal、a/an、ok、match等。两者之间使用to或to.be连接。

断言，就是判断源码的实际执行结果与预期结果是否一致，如果不一致就抛出一个错误



**测试 mocha index.test.js 可以同时测试多个案例**

**通常情况下将所有 测试文件放在test文件夹下，输入mocha可以进行所有并列测试文件的执行**

**如果 test文件夹下有另外的文件夹 需要执行测试文件 输入mocha --recursive 这样不论test下多少文件，都可以被执行**

**通配符mocha test／{a,b}.js 执行测试test下的a，b文件**

**通配符mocha typ／a／＊.js 执行测试test下的a文件下的所有测试文件**

**展示效果**

mocha --reporter 默认spec 还可以设置为 tap/dot/nyan/landing/list/progress/JSON/JSON stream/min/doc等

**使用模版**

**npm install --save-dev mochawesome   
mac 专用 ../node\_modules/.bin/mocha --reporter mochawesome  
生成mochaawesome-reports目录 查看html文件即可  
tips:windows下需要安装全局环境 npm i -g mochawesome --save-dev 然后在相应文件夹下 执行 mocha --reporter mochawesome**

**监听测试： mocha --watch/mocha -w**

**--bail, -b 指定只要有一个测试用例没有通过，就停止执行后面的测试用例**

**--grep, -g用于搜索测试用例的名称（即it块的第一个参数），然后只执行匹配的测试用例。**

**--invert, -i表示只运行不符合条件的测试脚本，必须与--grep参数配合使用。eg:mocha --grep "1+1是等于2" --invert**

**配置 在test目录下新建mocha.opts文件 写入--reporter tap --recursive 配置 执行mocha 即可运行以上指令**

**es6**

**如果测试脚本使用的是es6语法编写的 按以下步骤操作**



**第一步 安装Babel依赖 npm install babel-core babel-preset-es2015 --save-dev**

**第二步 在项目目录下面，新建一个.babelrc配置文件。写入配置 {"presets": [ "es2015" ]}**

**第三步 使用--compilers参数指定测试脚本的转码器；测试 ../node\_modules/mocha/bin/mocha --compilers js:babel-core/register**

上面代码表示，运行测试之前，先用babel-core/register模块，处理一下.js文件

简便：修改package.json文件 test 将mocha --compilers js:babel-core/register 写入，执行npm run test (tips:脚本文件需放在根目录下)

**Mocha默认每个测试用例最多执行2000毫秒，如果到时没有得到结果，就报错。对于涉及异步操作的测试用例，这个时间往往是不够的，需要用-t或--timeout参数指定超时门槛。**

查看案例 demo5

**Mocha在describe块之中，提供测试用例的四个钩子:before()、after()、beforeEach()和afterEach()。它们会在指定时间执行。**

before(function() { // 在本区块的所有测试用例之前执行 });   
after(function() { // 在本区块的所有测试用例之后执行 });   
beforeEach(function() { // 在本区块的每个测试用例之前执行 });   
afterEach(function() { // 在本区块的每个测试用例之后执行 });

**大型项目有很多测试用例。有时，我们希望只运行其中的几个，这时可以用only方法。describe块和it块都允许调用only方法，表示只运行某个测试套件或测试用例。**

**it.only() 只有带有only方法的测试用例会运行, 其他不执行测试。**

**it.skip() 表示跳过指定的测试套件或测试用例，其他执行。**

**浏览器测试**

**首先，使用mocha init命令在指定目录生成初始化文件。 mocha init mydemo4 会在mydemo4中生成相应文件夹**

**然后，新建一个源码文件add.js 写入测试函数 function() 不需要模块暴露**

**接着 在index.html中引入该js文件 以及 script src="http://chaijs.com/chai.js" script**

**最后 在tests.js中写测试代码 var expect = chai.expect;**

**打开index。html运行即可**

**生成规格文件**

**mocha --recursive -R markdown > spec.md   
mocha --recursive -R doc > spec.html两种格式**