曾经我以为能为除 bug 奉献我的一生，现在我才发现已经快被手头这个庞大臃肿的项目折磨得精疲力尽即将英年早逝……

聊聊 JS 测试框架

严清 @ teambition

https://www.teambition.com, Best collaboration tool for team.

小广告：https://github.com/toajs/toa

Toa，是基于 Koa 改进的 web 框架

稳定的 API 设计，兼容过去，也兼容未来

```
const app = Toa()

app.use(function () {
  console.log(this.method, this.url)
})
app.use(function (done) {
  console.log(this.method, this.url)
  done()
})
app.use(function * () {
  yield 'some thing'
  console.log(this.method, this.url)
})
app.use(async function () {
  await Promise.resolve('some thing')
  console.log(this.method, this.url)
})
```

Koa v1 到 v2 的 breaking change：

一切都是 next 的锅

给中间件函数引入了状态，也导致不兼容升级

```
const app = koa()

app.use(function * (next) {
  yield next // next is a generator function
  console.log(this.method, this.url)
})
```

```
const app = new Koa()

app.use(async (ctx, next) => {
  await next() // next is now a normal function
  console.log(ctx.method, ctx.url)
})
```

小广告：https://github.com/toajs/quic

A QUIC server/client implementation in Node.js

QUIC, a multiplexed stream transport over UDP
Key features of **QUIC** over existing **TCP+TLS+HTTP2** include

- Dramatically reduced connection establishment time
- Improved congestion control
- Multiplexing without head of line blocking
- Forward error correction
- Connection migration

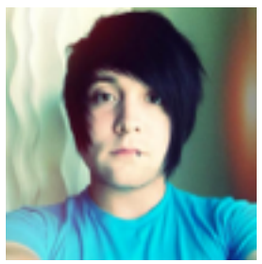
teambition

进入主题：大神们一言不合就造轮子~

mocha 流

0.9M

jasmine @pivotal, 2009



3.4M

mocha @tj, 2011

Jest @facebook, 2014

tape 流

tap @isaacs,2011

1.3M

tape @substack, 2012

ava @sindresorhus, 2014

teambition

一个现代的测试集需要的能力：

- **组织、控制并运行测试用例**，关键词：suite, describe, it, before, after, beforeEach, afterEach, only, skip, timeout
- **支持各种异步测试用例**，关键词：thunk, promise, generator, async/await, observable
- 方便的断言工具，关键词：ok, fail, equal, deepEqual, strictEqual, throws，代表作：**Should.js**, **Chai**
- 其它辅助工具，关键词：spy, stub, mock, sandbox, 代表作：**Sinon.JS**

tape 流的基本套路：

```
const test = require('tape')

test('timing test', function (t) {
  t.plan(2)

  t.equal(typeof Date.now, 'function')
  var start = Date.now()
  setTimeout(function () {
    t.equal(Date.now() - start, 100)
  }, 100)
})
```

- 只有 Test class，没有 Suite class
- Test 集成了丰富的断言方法
- 通过 plan 和断言数量决定同步或异步测试的生命周期
- 不支持第三方断言库，不直接支持异步测试实例

tape 一些基础 API：

```
test([name], [opts], cb)
test.only(name, cb)
test.skip(name, cb)
test.onFinish(fn) // 不支持 before,  after 等 Hooks

t.plan(n) // tape 流的灵魂…
t.end(err)
t.fail(msg)
t.pass(msg)
t.timeoutAfter(ms)
t.skip(msg)
t.ok(value, msg)
t.notOk(value, msg)
t.error(err, msg)
t.equal(actual, expected, msg)
t.deepEqual(actual, expected, msg)
t.throws(fn, expected, msg)
t.test(name, [opts], cb)  // 嵌套 test，不建议使用，非预期的运行时序
```

ava 在 tape 流上的改进：

```
test.cb(t => {
  t.plan(1)
  someAsyncFunction(() => {
    t.pass()
    t.end()
  })
})
test.after(async t => {
  await promiseFn()
})
```

- 支持 Hooks
- 支持各种异步测试实例，不再依赖 plan 方法
- Test 对象的断言增强，不再支持嵌套 test
- 多个测试文件使用多线程测试，文件内的 test 并发测试

ava 不可避免的串行测试需求：

```
test.beforeEach(t => { console.log('beforeEach') })
test.afterEach(t => { console.log('afterEach') })
test.serial('test 1', t => {
  return new Promise((resolve) => {
    t.pass()
    setTimeout(resolve, 100)
  })
})
test.serial('test 2', t => {
  t.pass()
})
```

- 没有全局的 before，after hooks
- 没有 suite，不方便运行指定分组的测试，组织能力弱

mocha 流的基本套路:

```
describe('db', function() {
  before(function () { return db.clear() })

  describe('#save()', function () {
    it('should save without error', function (done) {
      new User('Luna').save(done)
    })
    it('respond with matching records', function () {
      return db.find({type: 'User'})
        .should.eventually.have.length(1)
    })
  })
})
```

- 引入 Suite 层，可以自由的组合和控制测试实例
- 直接支持异步测试实例
- 断言逻辑与控制逻辑完全解耦

关于 **mocha** 流三个知名框架：

- **Jasmine**，资历最老最正统，自带断言库，Spies等，仅支持 thunk 异步测试实例

- **mocha**，知名度最高，最纯粹的测试框架，专注于组织、控制和运行测试实例，异步只支持 thunk 和 promise 的测试实例

- **Jest**，Fackbook 的专为前端打造的测试全家桶，集成断言库和各种辅助工具，React 系御用~

踏上大神的轮子路~

先定个小目标：

造个 **mocha，** 攒 10k stars

一个更纯粹的更强大的 mocha 轮子：

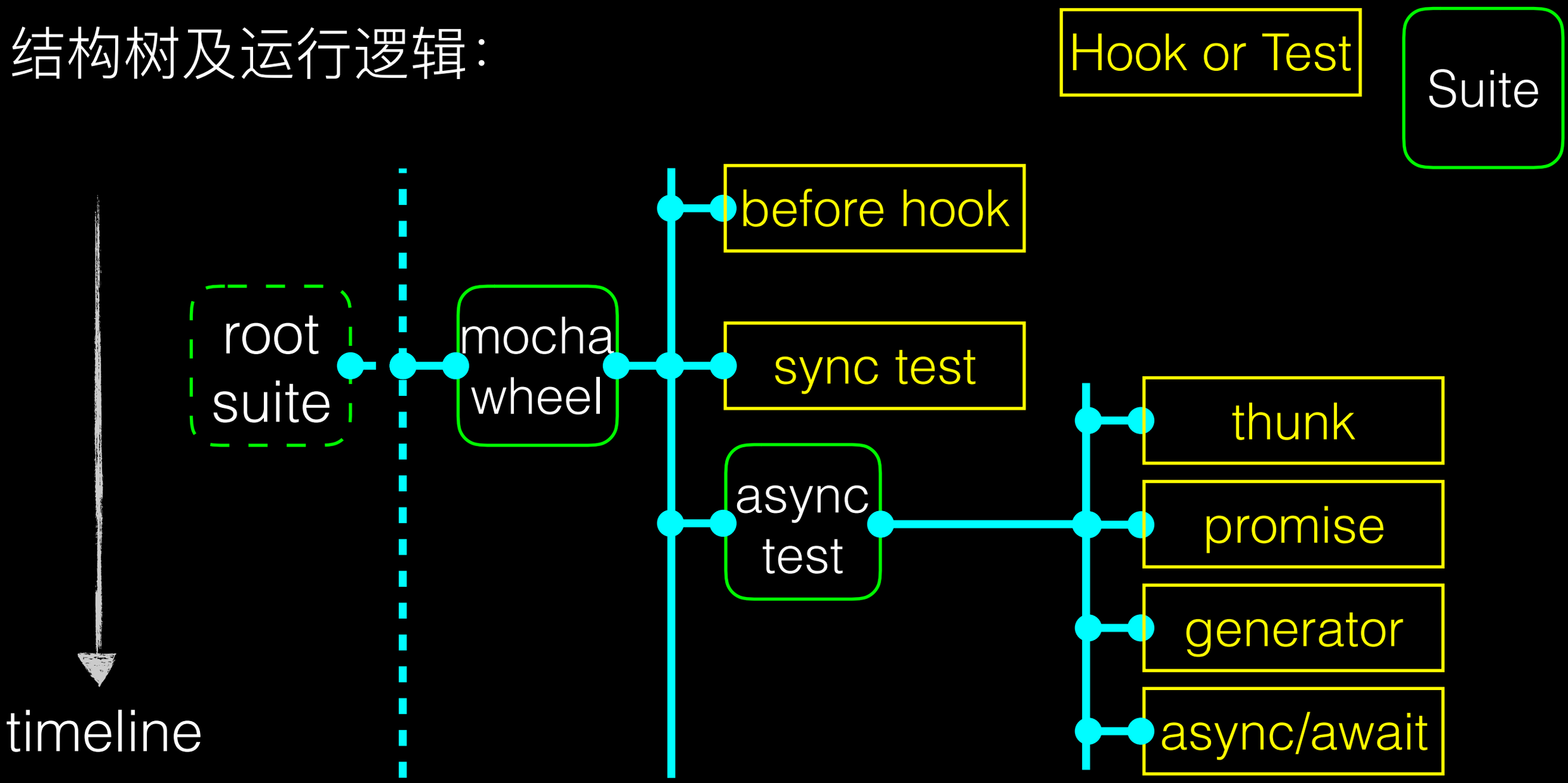
```
describe('mocha wheel', function () {
  before(function () {})

  it('sync test', function () {})

  describe('async test', function () {
    it('thunk', function (done) {
      setTimeout(done, 100)
    })
    it('promise', function () {
      return Promise.resolve('ok')
    })
    it('generator', function * () {
      yield function (done) { setTimeout(done, 100) }
      yield Promise.resolve('ok')
    })
    it('async/await', async function () {
      await Promise.resolve('ok')
    })
  })
})
```

1. 运行suite 解析结构树
2. 运行 hook 和 test
3. 处理测试结果

结构树及运行逻辑：



1. Suite 用于封装一组测试逻辑，测试初始化时生成一个 root suite
2. Suite 下可以定义 Hook，Test 和子级 Suite，从而形成树状层级关系
3. test.toThunk 和 hook.toThunk 把同步或异步的测试用例变成待运行的 thunk 函数
4. Suite.prototype.toThunk 把子级 Suite，Hook 和 Test 的 thunk 函数串行组合成 thunk 函数
5. rootSuite.toThunk() 就是我们需要运行的完整测试体，thunk 的**惰性求值**是关键

形成结构树的关键——context 状态机

```
// suite 中添加子 suite 的解析逻辑
Suite.prototype.addSuite = function (title, fn, mode) {
  var ctx = this.ctxMachine // 当前父级 suite
  var suite = new Suite(title, ctx, mode)
  ctx.children.push(suite)
  this.ctxMachine = suite // 切换到子 suite
  fn.call(suite) // 解析子 suite 的测试逻辑
  this.ctxMachine = ctx // 子 suite 解析完毕，还原之前的 context
}

// suite 中添加 test 的解析逻辑
Suite.prototype.addTest = function (title, fn, mode) {
  var ctx = this.ctxMachine // 当前父级 suite
  var test = new Test(title, ctx, fn, mode)
  ctx.children.push(test) // test 直接 push 到父级 suite
}
```

踏上大神的轮子路~

轮子已成，攒 stars：

https://github.com/thunks/tman

- 核心代码 **500** 行，browserify 打包也才 **1400+** 行
- 几乎兼容绝大部分 mocha 项目，可直接切换到 tman
- 全面支持 generator, async/await, observable 等各种形式的异步测试
- 内部接口非常灵活，可在 tman 的基础上开发出具有特色功能的自用测试框架，像基于 koa 的 egg 一样，做个基于 tman 的 stone 测试框架 @天猪~