

前端单元测试探索

ecmadao

Unit Test

对一个模块、一个函数或者一个类
来进行正确性检验的测试工作

代码先行，测试跟上

TDD

不同于传统软件开发，
要求在开发具体代码之前，先编写测试代码。
之后，通过编写使测试通过的代码，来驱动开发的
进行

测试先行，代码逐步完善

BDD

行为驱动开发

重点是通过与利益相关者的讨论，
取得对预期的软件行为的清醒认识

重于沟通

Something you need to know about UnitTest

需要访问数据库的测试不是单元测试

需要访问网络的测试不是单元测试

需要访问文件系统的测试不是单元测试

保持代码纯净性，降低外部依赖，向纯函数靠拢

避免太多条件判断，避免在构造函数中进行大量操作

Something you need to know about TDD

TDD以测试先行。但测试仅仅是手段而不是目的

从函数调用者的角度出发，尝试函数逻辑的各种可能性，
进而辅助性增强代码质量

快速运行、快速编写。不忽略任一个失败的测试

测试不是为了覆盖率和正确率，
而是作为实例，告诉开发人员要编写什么代码

Something you need to know about BDD

相比于其他地区，敏捷开发在亚洲更难以得到很好的执行

WHY?

- 1.有人希望有人告诉自己该干什么，也有人总想指挥别人干事，整个工作有时会过于井然有序
- 2.敏捷开发需要大家当面直言问题所在，而对于亚洲人而言很多人 too shy to 撕逼
- 3.从教育上而言，亚洲的教育是为了考高分、定级别，而不是为了尝试、自我发现和试错，可这些却正是敏捷实践的目的所在

TDD workflow

Step1.需求分析，思考实现

考虑如何“使用”产品代码，
是一个实例方法还是一个类方法，
是从构造函数传参还是从方法调用传参，
方法的命名，返回值等。

这时其实就是在做设计，而且设计以代码来体现

此时测试为红

TDD workflow

Step2.实现代码

根据写测试时思考的流程进行函数的编写

但注意，随着测试的增多，要保持编写代码的设计，
以及测试的简洁有效

直至测试为绿

TDD workflow

Step3.重构

保持当前测试的完整性，并重构代码
消除冗余与重复，提高内聚度

最终要保证每个概念都被清晰的表达，没有多余的东西
并能够通过测试

之后重复进行第一步（测试）

BDD workflow

Step1

从业务的角度定义具体的，以及可衡量的目标

找到一种可以达到设定目标的、
对业务最重要的那些功能的方法

然后像故事一样描述出一个个具体可执行的行为
其描述方法基于一些通用词汇，这些词汇具有准确无误的表达能力和一致的含义。例如，expect, should, assert

BDD workflow

Step2

寻找合适语言及方法，对行为进行实现

BDD workflow

Step3

测试人员检验产品运行结果是否符合预期行为
最大程度的交付出符合用户期望的产品，避免表达不一致带来的问题

JavaScript UnitTest based on Mocha

1.setup

```
$ npm install mocha --save
```

```
$ npm install chai --save
```

```
$ npm install --save-dev babel-core
```

```
$ npm install --save coffee-script
```

JavaScript UnitTest based on Mocha

2.config

```
{  
  "scripts": {  
    "test": "./node_modules/mocha/bin/mocha --compilers  
js:babel-core/register",coffee:coffee-script/register"  
  }  
}
```

JavaScript UnitTest based on Mocha

3.chai

```
import chai from 'chai';  
  
const assert = chai.assert;  
const expect = chai.expect;  
const should = chai.should();
```

```
foo.should.be.a('string');  
foo.should.equal('bar');  
list.should.have.length(3);  
  
expect(foo).to.be.a('string');  
expect(foo).to.equal('bar');  
expect(list).to.have.length(3);  
  
assert.typeOf(foo, 'string');  
assert.equal(foo, 'bar');  
assert.lengthOf(list, 3);
```


JavaScript UnitTest based on Mocha

4.Test

```
describe('describe a test', () => {  
  it('should return true', () => {  
    let example = true;  
    expect(example).not.to.equal(false);  
    expect(example).to.equal(true);  
  });  
});
```

JavaScript UnitTest based on Mocha

5.AsynTest

mocha无法自动监听异步方法的完成
需要我们在完成之后手动调用done()方法

而如果要在回调之后使用异步测试语句，则需要使用try/catch进行捕获
成功则done()，失败则done(error)

```
// 普通的测试方法
it("should work", () =>{
  console.log("Synchronous test");
});

// 异步的测试方法
it("should work", (done) =>{
  setTimeout(() => {
    try {
      expect(1).not.to.equal(0);
      done(); // 成功
    } catch (err) {
      done(err); // 失败
    }
  }, 200);
});
```

JavaScript UnitTest based on Mocha

6.Redux

```
// 测试actions
import * as ACTIONS from '../redux/actions';

describe('test actions', () => {
  it('should return an action to create a todo',
    () => {
      let expectedAction = {
        type: ACTIONS.NEW_TODO,
        todo: 'this is a new todo'
      };
      expect(ACTIONS.addNewTodo('this is a new
      todo')).to.deep.equal(expectedAction);
    });
});
```

```
// 测试reducer
import * as REDUCERS from '../redux/reducers';
import * as ACTIONS from '../redux/actions';

describe('todos', () => {
  let todos = [];
  it('should add a new todo', () => {
    todos.push({
      todo: 'new todo',
      complete: false
    });
    expect(REDUCERS.todos(todos, {
      type: ACTIONS.NEW_TODO,
      todo: 'new todo'
    })).to.deep.equal([
      {
        todo: 'new todo',
        complete: false
      }
    ]);
  });
});
```

```
// 一旦注册就会时刻监听state变化
const subscribeListener = (result, done) => {
  return AppStore.subscribe(() => {
    expect(AppStore.getState()).to.deep.equal(result);
    done();
  });
};

describe('use store in unittest', () => {
  it('should create a todo', (done) => {
    // 首先取得我们的期望值
    state.todos.append({
      todo: 'new todo',
      complete: false
    });

    // 注册state监听
    let unsubscribe = subscribeListener(state, done);
    AppStore.dispatch(ACTIONS.addNewTodo('new todo'));
    // 结束之后取消监听
    unsubscribe();
  });
});
```

测试是手段

测试是手段

测试是手段



**Use Python
to create your workflow**