学习 单元测试

# Sinon

Sinon提供了一些方法，允许我们替换掉代码中复杂的代码简化测试。通过创建”测试替身”(Test Doubles)的方式降低测试的复杂度。
主要提供了：

## Spies

测试间谍（test spy）是一个监听函数，通过创建匿名函数或者包裹已有的函数的方式创建，当它被调用的时候会记录传入参数、返回值、`this`以及抛出的异常，不会改变原函数的行为。
**create Spy**

- 匿名函数

```
1. sion.spy()
```

- 包裹function
  sion.spy(myFun)
- 包裹对象的方法

```
1. sion.spy(jQuery,"ajax");
2. jQuery.ajax.restore(); //取消spy包裹
```

**Spy API**

- `spy.calledWith(arg1,arg2,...)`：测试特定参数的调用，如果被调用返回`true`
- `spy.withArgs(arg1,[,arg2,...])`：记录特定参数的调用
- `spy.callCount`：纪录的调用次数
- `spy.called`：返回`true`如果被调用
- `spy.calledOnce`：返回`true`如果只被调用一次
- `spy.calledTwice`:同上
- `spy.calledTrice`：同上
- `spy.firstCall spy.secondCall spy.thirdCall spy.lastCall`

- `spy.getCall(n)`：某次特定的调用
- **其他有用的API**

# Stubs

测试桩（test stubs）是预定义行为的spy，支持所有的spy的API并且附加了一些改变桩行为的方法，会完全替换目标函数。
以下情况考虑使用测试桩：

- 控制方法以特定路径执行，例如控制返回，控制抛出异常。
- 避免特定方法被调用

example1

```
// 控制函数在接收特定参数的时的返回
const callback= sinon.stub();
callback().withArgs('name').returns('tom')
callback('name') //'tom'
callback() // 没有返回
```

example2

```
// 控制函数在接收特定参数的时的返回
const Obj={
    prop:()=>'test'
}
sinon.stub(Obj,'prop').callsFake(()=>'fake')
Obj.prop(); //'fake'
```