代码覆盖率工具 Istanbul 入门教程

作者： [阮一峰](http://www.ruanyifeng.com/)

日期： [2015年6月 8日](http://www.ruanyifeng.com/blog/2015/06/)

测试的时候，我们常常关心，是否所有代码都测试到了。

这个指标就叫做["代码覆盖率"](http://en.wikipedia.org/wiki/Code_coverage" \t "_blank)（code coverage）。它有四个测量维度。

* **行覆盖率**（line coverage）：是否每一行都执行了？
* **函数覆盖率**（function coverage）：是否每个函数都调用了？
* **分支覆盖率**（branch coverage）：是否每个if代码块都执行了？
* **语句覆盖率**（statement coverage）：是否每个语句都执行了？

[Istanbul](https://github.com/gotwarlost/istanbul) 是 JavaScript 程序的代码覆盖率工具，本文介绍它的用法。



这个软件以土耳其最大城市伊斯坦布尔命名，因为土耳其地毯世界闻名，而地毯是用来覆盖的。

一、安装

Istanbul 是一个 npm 模块，安装非常简单，就一行命令。

$ npm install -g istanbul

二、覆盖率测试

来看一个例子，怎么使用 Istanbul 。下面是脚本文件 simple.js 。

var a = 1;

var b = 1;

if ((a + b) > 2) {

console.log('more than two');

}

使用 **istanbul cover** 命令，就能得到覆盖率。

$ istanbul cover simple.js

===== Coverage summary =====

Statements : 75% ( 3/4 )

Branches : 50% ( 1/2 )

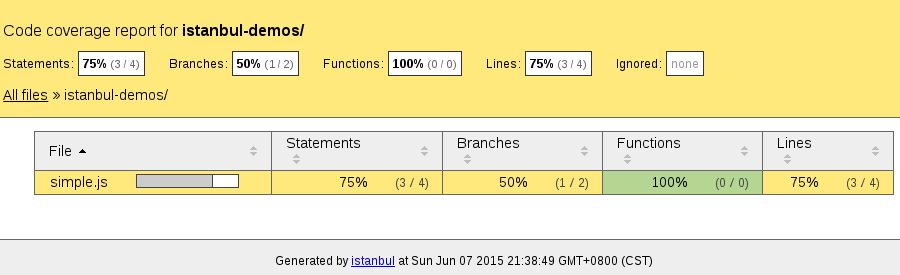
Functions : 100% ( 0/0 )

Lines : 75% ( 3/4 )

=============================

返回结果显示，simple.js 有4个语句（statement），执行了3个；有2个分支（branch），执行了1个；有0个函数，调用了0个；有4行代码，执行了3行。

这条命令同时还生成了一个 coverage 子目录，其中的 coverage.json 文件包含覆盖率的原始数据，coverage/lcov-report 是可以在浏览器打开的覆盖率报告，其中有详细信息，到底哪些代码没有覆盖到。



三、覆盖率门槛

完美的覆盖率当然是 100%，但是现实中很难达到。需要有一个门槛，衡量覆盖率是否达标。

**istanbul check-coverage** 命令用来设置门槛，同时检查当前代码是否达标。

$ istanbul check-coverage --statement 90

ERROR: Coverage for statements (75%) does not meet global threshold (90%)

上面命令设置语句覆盖率的门槛是 90% ，结果就报错了，因为实际覆盖率只有75%。

除了百分比门槛，我们还可以设置绝对值门槛，比如只允许有一个语句没有被覆盖到。

$ istanbul check-coverage --statement -1

上面命令使用负数，表示绝对值门槛。这样一来，上面的例子就通过了覆盖率测试，不会再报错了。

百分比门槛和绝对值门槛，可以结合使用。

$ istanbul check-coverage --statement -5 --branch -3 --function 100

上面命令设置了3个覆盖率门槛：5个语句、3个 if 代码块、100%的函数。注意，这三个门槛是"与"（and）的关系，只要有一个没有达标，就会报错。

四、与测试框架的结合

实际开发时，istanbul 总是与测试框架结合使用，下面以常用的 [Mocha 框架](http://mochajs.github.io/mocha/)为例。

sqrt.js 是一个计算平方根的脚本。

var My = {

sqrt: function(x) {

if (x < 0) throw new Error("负值没有平方根");

return Math.exp(Math.log(x)/2);

}

};

module.exports = My;

它的测试脚本 test.sqrt.js 放在 test 子目录。

var chai = require('chai');

var expect = chai.expect;

var My = require('../sqrt.js');

describe("sqrt", function() {

it("4的平方根应该等于2", function() {

expect(My.sqrt(4)).to.equal(2);

});

it("参数为负值时应该报错", function() {

expect(function(){ My.sqrt(-1); }).to.throw("负值没有平方根");

});

});

然后，执行下面的命令得到代码覆盖率。

$ istanbul cover \_mocha

// or

$ istanbul cover \_mocha test/test.sqrt.js

sqrt

✓ 4的平方根应该等于2

✓ 参数为负值时应该报错

2 passing (7ms)

===== Coverage summary =====

Statements : 100% ( 5/5 )

Branches : 100% ( 2/2 )

Functions : 100% ( 1/1 )

Lines : 100% ( 4/4 )

=============================

上面命令中，istanbul cover 命令后面跟的是 \_mocha 命令，前面的下划线是不能省略的。

因为，[mocha 和 \_mocha 是两个不同的命令](https://github.com/gotwarlost/istanbul/issues/44" \t "_blank)，前者会新建一个进程执行测试，而后者是在当前进程（即 istanbul 所在的进程）执行测试，只有这样， istanbul 才会捕捉到覆盖率数据。其他测试框架也是如此，必须在同一个进程执行测试。

如果要向 mocha 传入参数，可以写成下面的样子。

$ istanbul cover \_mocha -- tests/test.sqrt.js -R spec

上面命令中，两根连词线后面的部分，都会被当作参数传入 Mocha 。如果不加那两根连词线，它们就会被当作 istanbul 的参数（参考链接[1](http://www.clock.co.uk/blog/npm-module-code-coverage-in-2-simple-steps" \t "_blank)，[2](http://www.vapidspace.com/coding/2014/10/29/code-coverage-metrics-with-mocha-and-istanbul/" \t "_blank)）。

如果想在浏览器运行 Istanbul ，可以参考[这篇文章](http://ariya.ofilabs.com/2013/12/code-coverage-of-mocha-tests-using-istanbul-and-karma.html" \t "_blank)。

五、忽略某些代码

istanbul 提供[注释语法](https://github.com/gotwarlost/istanbul/blob/master/ignoring-code-for-coverage.md" \t "_blank)，允许某些代码不计入覆盖率。

var object = parameter || /\* istanbul ignore next \*/ {};

上面代码是为 object 指定默认值（一个空对象）。如果由于种种原因，没有为 object 为空对象的情况写测试，可以用注释，不将这种情况计入覆盖率。注意，注释要写在"或"运算符的后面。

/\* istanbul ignore if \*/

if (hardToReproduceError)) {

return callback(hardToReproduceError);

}

上面代码的 if 语句块，在计算覆盖率的时候会被忽略。

**如果失败:**

**D:\myEg2\myEg2\日常\覆盖率工具Istanbul\例子2与mocha框架>istanbul cover C:/Users/**

**Administrator/AppData/Roaming\npm/node\_modules/mocha/bin/\_mocha -- -R spec**