# SPIES, MOCKS, AND STUBS

Fake it 'til you make it

♦ FULLSTACK 1 TESTING

### **PREMISE**

- Pure functions are easy to test
  - Example: addition
    - Input (1, 2, 3) => reliable output (6), the same every time
- Non-pure functions are trickier
  - Example 1: readFile
    - Input (filename) => output depends on file contents, which can change even if the filename does not
  - Example 2: func depends on other funcs (e.g. randomized output)
- We want isolation!

♦ FULLSTACK
2 COURSE TITLE GOES HERE

Pure: no side effects (doesn't talk to the outside world, ajax...), and deterministic (same output for same input, reliably)



### **SPIES**

- An object that "spies" on a function to record metadata
  - Whether the function was called
  - How many times it was called
  - Arguments it was called with
  - etc...
- Can also be a dummy function to pass around
  - Output of this function doesn't matter, we only care about how it's being used by another function (e.g. callback with right args)

♦ FULLSTACK

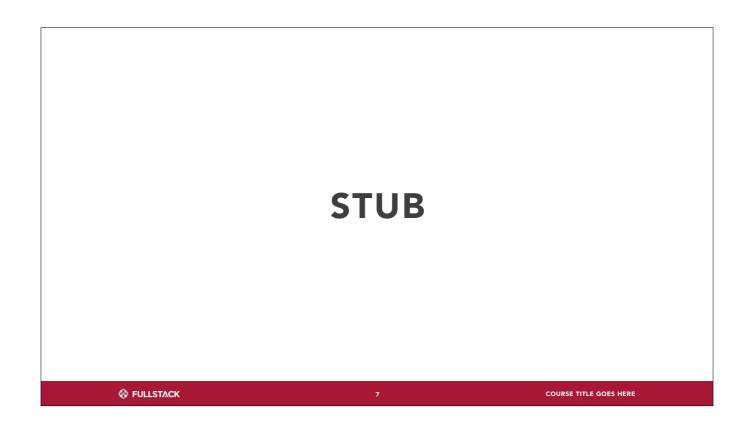
4 COURSE TITLE GOES HERE

### **SPIES - ANONYMOUS FUNCTION**

♦ FULLSTACK
5 COURSE TITLE GOES HERE

### **SPIES - WRAP AN EXISTING METHOD**

♦ FULLSTACK
6 COURSE TITLE GOES HERE



### **STUBS**

- Like a spy... but fake!
- Spies "spy" on a function (calls original/same behavior)
- Stubs *replace* a function entirely, "stubbing" their output
- Use cases:
  - Replacing external requests (e.g. AJAX, db queries) takes too long and not predictable outcome
  - Testing behavior tracks based on function output (e.g. coin flip: if Math.random gives >0.5, return heads; else tails)

♦ FULLSTACK

8 COURSE TITLE GOES HERE

### **STUBS**

```
let stub = sinon.stub(Math, 'random');
stub.returns(/* something specific */);
```

♦ FULLSTACK 9 COURSE TITLE GOES HERE



♦ FULLSTACK

10

COURSE TITLE GOES HERE

### **MOCKS**

- Like a stub... but with more functionality!
  - Allows you to stub all methods in an object at once
  - Comes with built-in assertions

♦ FULLSTACK 11 COURSE TITLE GOES HERE

## "LEAVE IT HOW YOU FOUND IT"

♦ FULLSTACK

12

COURSE TITLE GOES HERE

### **RESTORE**

- When using mocks and stubs, you are replacing behavior of functions that have other important tasks to get back to once you're done with testing!
- It's important to restore things to their original state whenever appropriate

♦ FULLSTACK 13 COURSE TITLE GOES HERE

### WHERE TO FIND THESE COOL TOOLS?!

- Spies/mocks/stubs often come packaged with your testing framework of choice
  - Each may have slightly different descriptions of "mock"/"stub"/"spy"
  - No need to get too hung up about terminology consult specific documentation and find the right tool for your task!
- You can also pick standalone tools (much like how we chose chai for assertions)
  - · Sinon.js is a popular choice for Mocha

♦ FULLSTACK 14 COURSE TITLE GOES HERE

https://learn.fullstackacademy.com/workshop/5a7b5735310c5c0004037021/content/5a7b7084c34ae00004ac6c2e/text