



# TESTING: HANDS-ON

PHILOSOPHY & NODEJS & FUN

# WHO AM I? MEET DANIEL...




 Backend Developer @ MoD

 B.Sc. in Computer Science @ Bar-Ilan University

 Handball &  Weightlifting

 Tech Enthusiast

 Backend Technologies  
Testing  
Open Source Projects  
Anomaly Detection

 Meetups

---

# AGENDA

- 
- What?
  - Why?
  - How?
    - Test Runners
      - Mocha
    - Assertion Libraries
      - Chai
  - Sinon
  - Rewire
  - Mockery
  - Tips & Tricks
  - A spoiler
  - Personal Note
  - Meme

# WHAT IS SOFTWARE TESTING?

01

Software testing is defined as an activity to check whether the **actual** results match the **expected** results and to ensure that the software system is **Defect free**.

02

It involves execution of a **software component or system component** to evaluate one or more properties of interest.

# PHILOSOPHY



- Much like we gain knowledge about the behavior of the physical universe via the scientific method, we gain knowledge about the behavior of our **software via a system of assertion, observation**, and experimentation called “**testing**.”
- The purpose of a test is to deliver us knowledge about the system, and **knowledge has different levels of value**. For example, testing that  $1 + 1$  still equals two no matter what time of day it's doesn't give us valuable knowledge. However, knowing that my code still works despite possible **breaking changes in APIs**.
- There are many ways to gain knowledge about a system, and testing is just one of them. We could also read its **code**, look at its **documentation**, talk to its **developers**, etc., and each of these would give us a **belief** about how the system behaves. However, testing **validates** our beliefs, and thus is particularly important out of all these methods.

# WHY SHOULD I BOTHER?

In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.

Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.

Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.

Some of the Amazon's third party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.

Vulnerability in Window 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.

In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.

China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent lives

In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.

In April of 1999, a software bug caused the failure of a \$1.2 billion military satellite launch, the costliest accident in history

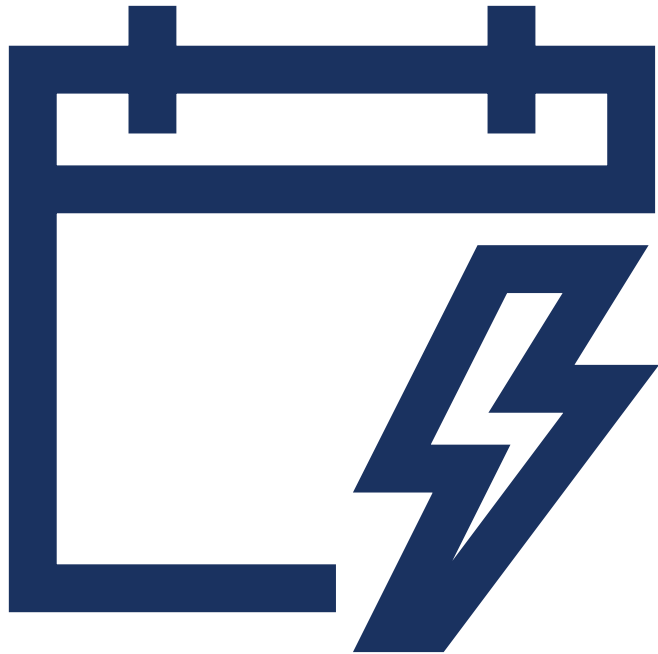
In May of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.



OK DANIEL WE GET IT – CODE CAN KILL



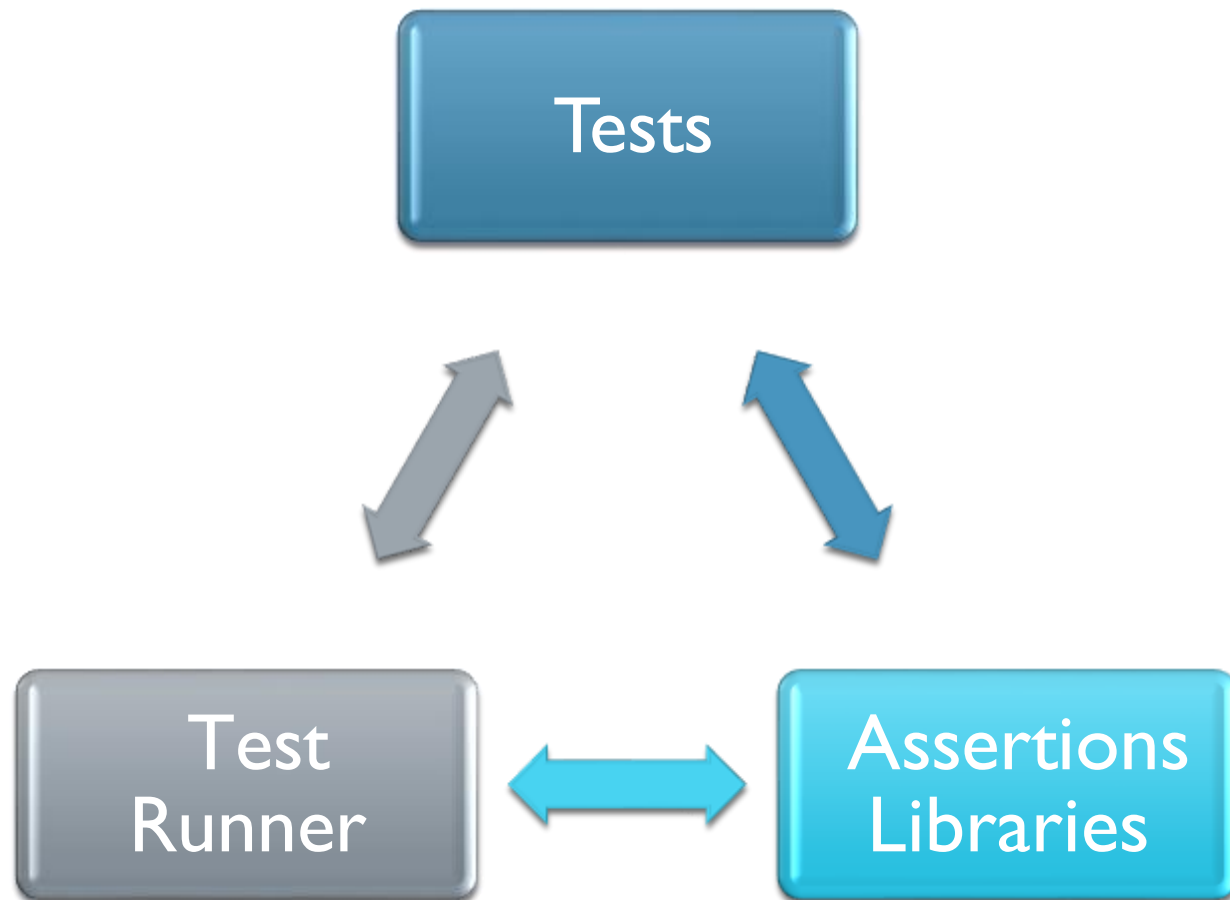
# WRITING TESTS VS SCHEDULE



- “Quality is the ally of schedule and cost, not their adversary. If we have to sacrifice quality to meet schedule, it’s because we are doing the job wrong from the very beginning.” – James A. Ward
- “The bitterness of poor quality remains long after the sweetness of meeting the schedule has been forgotten.” – Anonymous
- “Quality is free, but only to those who are willing to pay heavily for it.” – T. DeMarco and T. Lister



# HOW CAN WE TEST OUR CODE?



# TEST RUNNERS

A **test runner** is the library or tool that picks up an assembly (or a source code directory) that contains **tests**, and a bunch of settings, and then executes them and writes the **test** results to the console or log files.



there are many **runners** for different languages.



Each has its own syntax and behavior

---

# MOCHA SYNTAX

- 
- BDD (MOST COMMON):
    - `describe()` (Alias `context`)
    - `it()` (Alias `specify`)
    - `before()` & `beforeEach()`
    - `after()` & `afterEach()`
  - TDD:
    - `suite()`
    - `test()`
    - `suiteSetup()`
    - `suiteTeardown()`
    - `setup()`
    - `teardown()`

```
Daniel is
```

```
✓ cool
```

```
✓ smart
```

```
✓ modest
```

```
✓ in the best team in the panda (GO TESLA!)
```

```
Daniel should
```

```
1) stop drinking coffee
```

```
4 passing (24ms)
```

```
1 failing
```

```
1) Daniel should
```

```
stop drinking coffee:
```

```
Error: NO!!!!!!!
```

```
at Context.<anonymous> (tests\daniel-testing.js:14
```

# MOCHA CONSOLE OUTPUT

# SYNTAX.TEST.JS

THIS IS WHERE THE FUN BEGINS



---


# MOCHA CLI

- 
- All you need to know about test running is explained in
    - `mocha --help` (for global installation)
    - `node .\node_modules\mocha\bin\_mocha --help` (for local installation)

Let's cover some of its features together!

## ASSERTION LIBRARIES

Assertion libraries are tools to verify that things are correct.  
This makes it a lot easier to test your code, so you don't have to do  
**thousands** of if statements.



Provides the way to write custom assertions closer to the business language.

# CHAI

- Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with **any** javascript testing framework.





# CHAI SYNTAX SAMPLES

## Should

```
chai.should();

foo.should.be.a('string');
foo.should.equal('bar');
foo.should.have.lengthOf(3);
tea.should.have.property('flavors')
  .with.lengthOf(3);
```

[Visit Should Guide](#)

## Expect

```
var expect = chai.expect;

expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.lengthOf(3);
expect(tea).to.have.property('flavors')
  .with.lengthOf(3);
```

[Visit Expect Guide](#) →

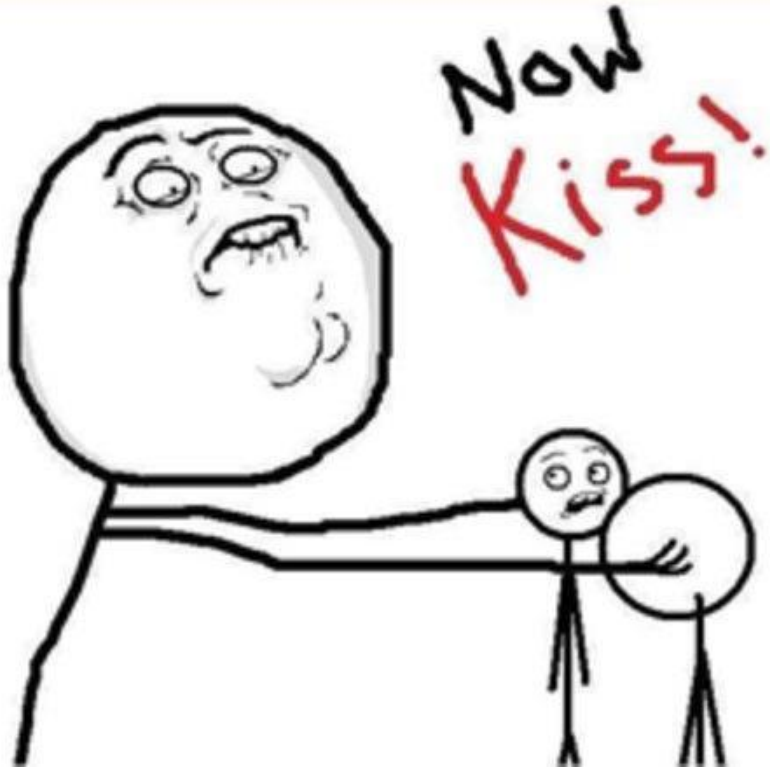
## Assert

```
var assert = chai.assert;

assert.typeOf(foo, 'string');
assert.equal(foo, 'bar');
assert.lengthOf(foo, 3);
assert.property(tea, 'flavors');
assert.lengthOf(tea.flavors, 3);
```

[Visit Assert Guide](#)

# MOCHA + CHAI = LOVE



Having Mocha and Chai combined eases your pain writing tests.

Let's see it in action.

HERE WE GO

**Mocha-chai.test.js**

# SINON + REWIRE

"**Sinon.** JS - Standalone test fakes, spies, stubs and mocks for JavaScript. " - Sinon official website

- Works with any unit testing framework.
- Documentation can be found here @ <https://sinonjs.org/releases/v7.5.0/>

"Rewire.js - Easy monkey-patching for node.js unit tests"

Rewire adds a special setter and getter to modules so you can modify their behavior for better unit testing.

- inject mocks for other modules or globals like process
- inspect private variables
- override variables within the module.

SINON-GAME.TEST.JS



# MOCKERY

Mockery lets you work more easily with your framework of choice (or no framework) to get your mocks hooked in to all the right places in the code you need to test.

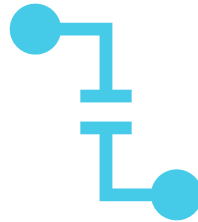
Mockery is used more for ***stubs*** since it replaces the ***module***

MOCKERY-SINON.TEST.JS

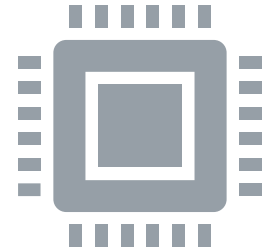
# NOCK



HTTP server mocking and expectations library for Node.js



Nock can be used to test modules that perform HTTP requests in isolation.



For instance, if a module performs HTTP requests to a Elasticsearch server or makes HTTP requests to the Amazon API, you can test that module in isolation.

NOCK-REQUEST.TEST.JS





# SUPERTEST

Small progressive client-side HTTP request library, and Node.js module with the same API, supporting many high-level HTTP client features

**SUPERTEST-EXPRESS.TEST.JS**

**TIPS & TRICKS**



SPOILER ALERT



PERSONAL NOTE



# AGENDA

- ✓ What?
- ✓ Why?
- ✓ How?
  - ✓ Test Runners
    - ✓ Mocha
  - ✓ Assertion Libraries
    - ✓ Chai
- ✓ Sinon
- ✓ Rewire
- ✓ Mockery
- ✓ Nock
- ✓ Supertest
- ✓ Tips & Tricks
- ✓ A spoiler
- ✓ Personal Note
- ✗ Meme

Me randomly  
meowing to  
my cat

My cat  
wondering why I  
said the n word



# AGENDA

- ✓ What?
- ✓ Why?
- ✓ How?
  - ✓ Test Runners
    - ✓ Mocha
  - ✓ Assertion Libraries
    - ✓ Chai
- ✓ Sinon
- ✓ Rewire
- ✓ Mockery
- ✓ Nock
- ✓ Supertest
- ✓ Tips & Tricks
- ✓ A spoiler
- ✓ Personal Note
- ✓ Meme



## MUST READ

- [best practices for spies stubs and mocks in sinonjs](#)
- [Excelling with Sinon.js](#)
- [Sinon.js Tutorial - How to Write Unit Tests Using Sinon.js \(Video\)](#)
- [How to Stub Dependencies with Sinon | JavaScript Testing Tutorials \(Video\)](#)

## MORE ON TESTING....

- [How to Unit Test with NodeJS?](#)
- [Node.js & JavaScript Testing Best Practices \(2019\)](#)
- [Advanced Node.js testing: beyond unit & integration tests \(2019\) \(Video\)](#)
- [Mocking HTTP requests with Nock](#)
- [NodeJS Best Practices](#)
- Google it!



# THANK YOU

DANIEL HERMON

[DANIELHERMONSYNC@GMAIL.COM](mailto:DANIELHERMONSYNC@GMAIL.COM)

DANIEL HERMON @ LINKEDIN

SYNCUSH @ GITHUB

SYNCUSH @ KAGGLE