CS454 Node.js & Angular.js

Cydney Auman
CSULA

Testing - Week 9

Testing

Testing is an important aspect of writing good quality, maintainable code that does what you expect.

Unit Testing

At a high-level, unit testing refers to the practice of testing certain functions and areas – or units – of our code. This gives us the ability to verify that our functions work as expected.

That is to say that for any function and given a set of inputs, we can determine if the function is returning the proper values and will gracefully handle failures during the course of execution should invalid input be provided.

Ultimately, this helps us to identify failures in our algorithms and/or logic to help improve the quality of the code that composes a certain function.

Test Driven Development (TDD)

1. First the developer writes some tests.

2. The developer then runs those tests and (obviously) they fail because none of those features are actually implemented.

3. Next the developer actually implements those tests in code.

4. If the developer writes their code well, then the in next stage they will see the tests pass.

5. The developer can then refactor their code, add comments, clean it up, as they wish because the developer knows that if the new code breaks something, then the tests will alert them by failing

Behavior-Driven Development (BDD)

Combines the general techniques and principles of **TDD** with a focus on the behavioral aspects of the system.

In contrast to TDD, BDD gives a clearer understanding as to what the system should do from the perspective of the developer and the customer.

Tools

Mocha

Mocha is a JavaScript test framework running on **node.js** and the browser. It was designed to make asynchronous testing simple.

Mocha tests run serially, allowing for flexible and accurate testing, while mapping uncaught exceptions to the correct test cases.

Tools

Mocha

describe()

- A grouping - typically the top level is file, then method.

it()

- The beginning of a test case.

before(), after()

- hooks to run before/after it() or describe().

Tools

Chai

A BDD / TDD assertion library for node and the browser that can be paired with any javascript testing framework

Chai has several interfaces that allow the developer to choose the most comfortable. The chain-capable BDD styles provide an expressive language & readable style, while the TDD assert style provides a more classical feel.

Tools

Chai

Should

```
chai.should();

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

Visit Should Guide

Expect

```
var expect = chai.expect;

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

Additional Tools

supertest

- module that allows you to test restful APIs. written by the same team that wrote superagent.

sinon/nock

- create mock HTTP requests. stub out data and simulate callbacks.

rewire

- module that allows you to easily test dependencies. rewire adds a special setter and getter to modules so you can modify their behaviour for better unit testing.