## **Intro to Testing**

In a perfect world you would be able to write code, deploy it and never have to touch it again because it functions flawlessly. The reality of course is that things can, and often do go wrong. Whether its your own mistakes that have caused a bug or some unforeseen circumstances, you need a way to ensure code integrity as well as confidence in your apps.

The first common solution is manual-testing, but as applications grow in size and complexity it becomes unrealistic to rely on just manual testing to verify new features, catch bugs and notice regressions. Unit tests are a great next step towards bug-free apps but there are issues that can occur that simply can't be caught within the scope of a unit-test. What else can be done?

Best practices dictate that a combination of unit testing and end-to-end testing (commonly referred to as E2E testing), should be implemented. If you're not familiar with what E2E testing is you can think of it as automated manual testing. Rather than looking at a unit or a specific component of an application, an end-to-end test will test the flow of an application from start to finish. An example of this might be a user logging into your application, adding an item to their cart and checking out.

## **The OrderCloud.io Framework of Choice**

Our E2E testing framework of choice is Protractor which was built by the Angular team. This framework is built on top of [Jasmine](http://jasmine.github.io/2.4/introduction.html) and [Selenium](http://www.seleniumhq.org/docs/) and extends functionality from those frameworks to cover Angular specific testing scenarios such as page synchronization and Angular specific locators. These are features key to testing Angular apps that you won't find anywhere else.

Our seed comes with Protractor already configured and ready to run and we've leveraged the automation power of Gulp to make running your tests easier than ever. All of our E2E tests are conveniently located in an E2E folder in the root of the project directory. Our intelligent build system picks up any files within this directory ending in spec.js so you will want to make sure to put any of your own Protractor tests within this directory.

The other thing to note within the project structure is the configuration file for Protractor titled protractor.config.js in the root of the project directory. The configuration file is set up to meet the needs of most of our customers but if you would like to change the way tests are run you certainly can.

## **Who Can Write Protractor Tests**

A common belief is that writing automated tests is a very technical process best left to developers. For E2E testing this is actually not true. Besides a very basic knowledge of CSS and HTML, even non-developers can write good automated tests. This allows your QA and Testing members to take away some of the burden from the developers.

## **Learning to Write Your Own Protractor Tests**

OrderCloud.io recommends taking some time to learn more about Protractor before getting started with writing tests. Pluralsight's [Introduction to Protractor](https://www.pluralsight.com/courses/protractor-introduction) course is very hands-on and provides an in-depth overview without being overwhelming, the subsequent courses also contain a lot of great information and tips to make your tests more dynamic.

Other resources:

* [Jasmine](http://jasmine.github.io/2.4/introduction.html) - A behavior-driven development framework for testing Javascript code. Protractor is built on top of this framework so being fluent with it is crucial. Luckily their documentation and syntax makes learning very easy.
* [Protractor](http://www.protractortest.org) - If you don't have access to Pluralsight, Protractor's Quick Start guide is certainly worthwhile. They also have great examples and of course a reference to their API.
* [Syle Guide](http://www.protractortest.org/#/style-guide) - This is a great reference for best practices as well as strategies for organization of tests. By breaking your tests out into easily identifiable and navigable sections you are able to make swift and precise error detection.

## **Running your Tests**

To run your tests simply open a terminal window in your project directory and type gulp build to start your app. Now, open a new terminal window also in your project directory and type gulp test:e2e

That's it! Now, you can watch in amazement as your tests run in the browser. Any failing tests will be made immediately apparent to you and you can be confident your app is running as it should.