INTRODUCTION TO E2E TESTING

using Web Driver IO, Mocha, Should and Selenium

Node.js Cincy Meetup
August 12th, 2015

Presented By: Tony Keith

Presentation online at:
http://tlkeith.com/e2e-presentation.pdf

AGENDA

1. Introduction
2. Background
3. Objectives
4. Technologies
5. Architecture
6. Several Examples
7. Testing Possibilities
8. Demo
9. Tutorial and Working Test Site
10. Installation
11. Conclusion

INTRODUCTION

Tony Keith - Independent Consultant

www.tlkeith.com

- Developer (C, Java, PHP languages)
- Industrial Automation (Warehouse Management, Inventory Control, etc…)
- Online Payments (Credit Cards and Prepaid Cards)
- Security Specialist - PCI-DSS
- Many titles and many years of experience but no automated Q/A experience.
    - But I have a passion for Technology…
        - This is how I became interested in Node technologies

BACKGROUND

- I recently had an interesting challenge presented to me. I needed to introduce automated testing to a Q/A department with very little technical experience and no programming background.

- This was really two (2) separate challenges. The first was to research the technologies to use to do the automated testing. The second was to train the Q/A department.

- This presentation will only address the technologies used and what I learned in the process.

- The technologies worked well but I really had to search for information and I spent many hours figuring out issues. I had a hard time finding information on the Internet about these technologies all working together.

- I wanted to share this information, along with working example test scripts and a test web site to run the scripts against.

OBJECTIVES

Use Technologies That:

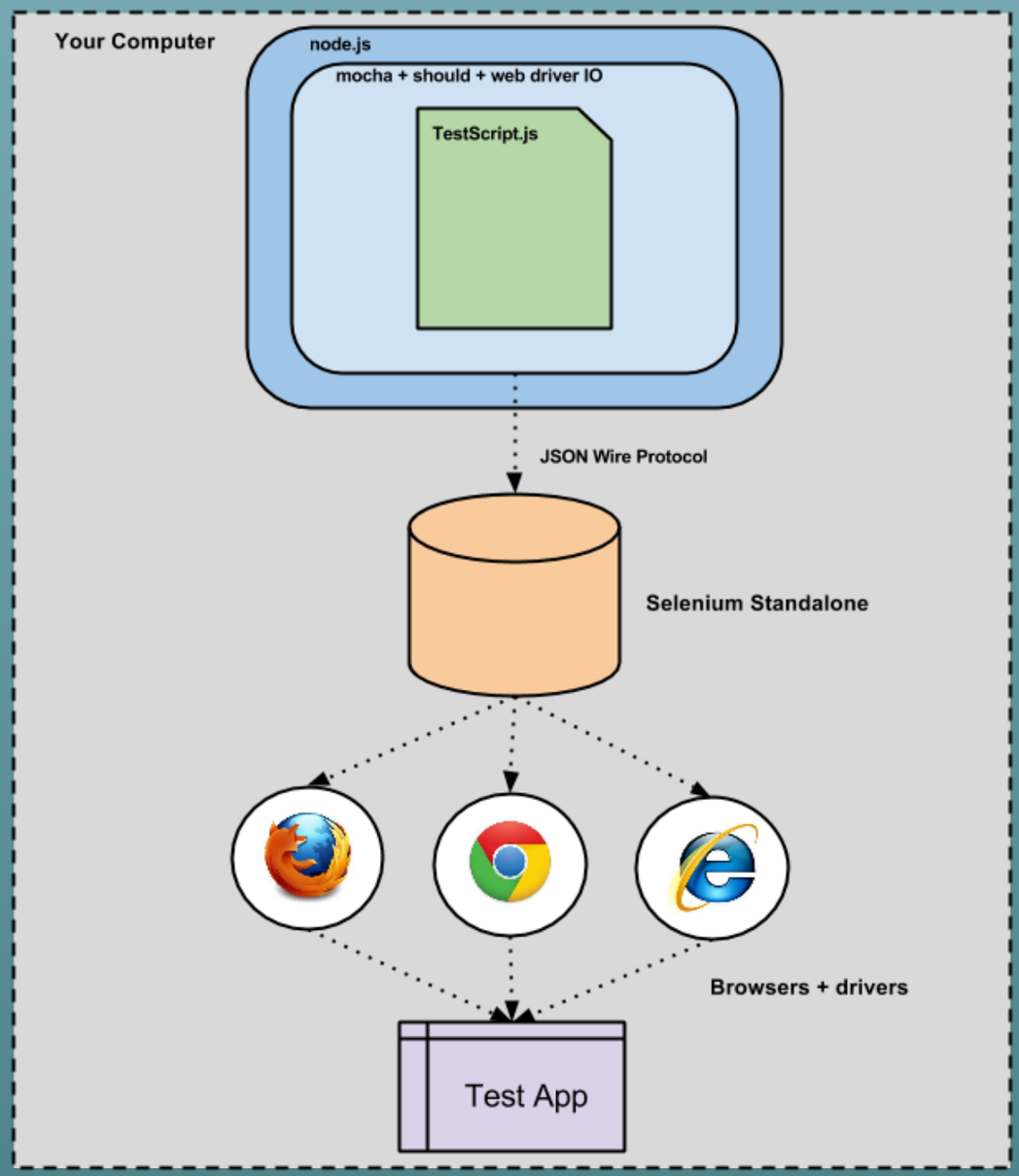
- **Can test web site functionality**
- **Can test JavaScript functionality**
- **Can be run manually from command line**
- **Can be run automatically using CI**
- **** Have an easy to learn language for non programmers ****
  - **Q/A personnel has basic knowledge of HTML and JavaScript**
- **Uses open source software only (except cloud based testing platforms)**
- **Can run tests on multiple OS/Browser versions and combinations**

TECHNOLOGIES

- **Node** - JS runtime environment
- **Mocha** – test runner framework and executes the test scripts (runner)
- **Should** – assertion library
- **Web Driver IO** – browser control bindings (JS programming language bindings - communicates with Selenium)
- **Selenium** – browser abstraction and running factory (starts and communicates with browser)
- **Grunt** - javascript task runner
- **Grunt-WebDriverIO** - plugin for Grunt with Mocha + Web Driver IO as test runner
- **Browser/Mobile drivers** + **Browsers** (IE, Chrome, Firefox, Safari)

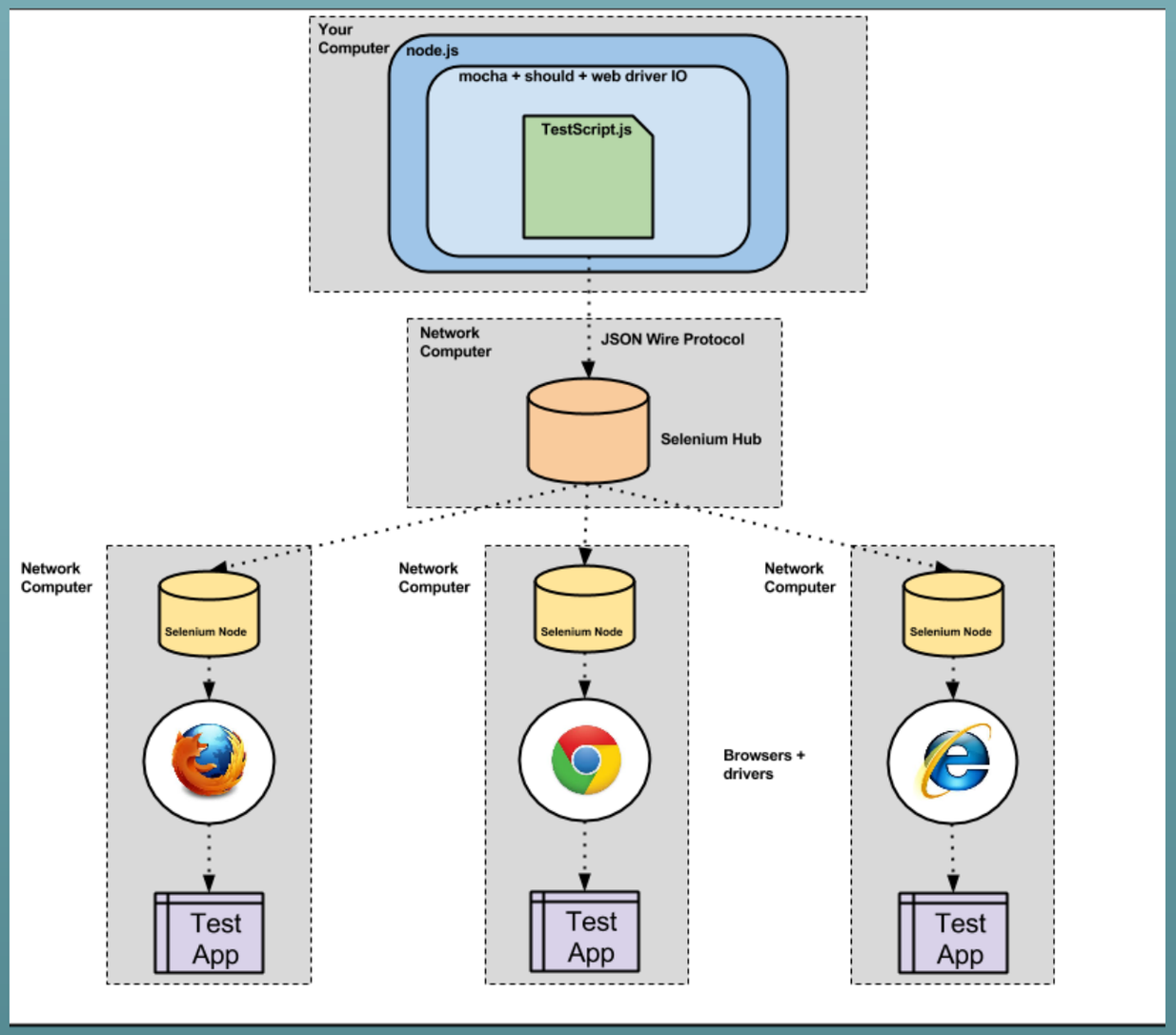
LOCAL SELENIUM SETUP

- **Node** runs **Mocha** test framework & runner of test script.
- **Should** is the assertion library.
- **Web Driver IO** communicates with **Selenium** using JSON Wire Protocol.
- **Selenium** invokes local browser using a driver to test the web application.



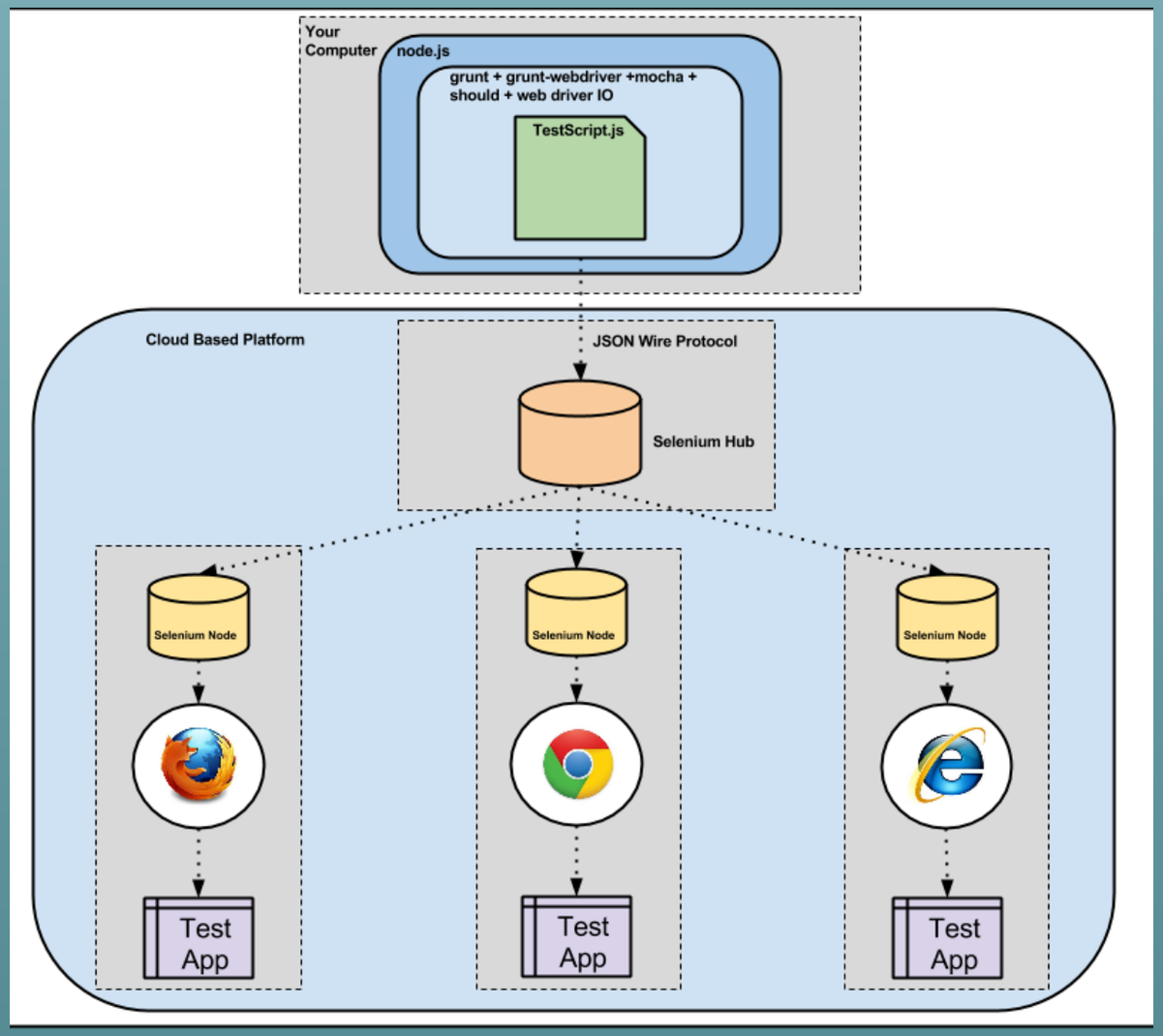
SELENIUM GRID SETUP

- **Node** runs **Mocha** test framework and runner of test script.
- **Should** is the assertion library.
- **Web Driver IO** communicates with **Selenium Hub** using JSON Wire Protocol.
- **Selenium Hub** routes requests to **Selenium Nodes** with different OS/Browsers combinations to test the web application on.



CLOUD BASED TESTING PLATFORM SETUP

- **Node** runs **Grunt** and **Grunt-Webdriver** plug-in runs **Mocha** the test framework and test script.
- **Should** is the assertion library.
- **Web Driver IO** communicates with cloud based testing platform using JSON Wire Protocol. (Saucelabs, Browserstack, …)
- Cloud based testing platform will automatically setup the correct OS/Browser combination to test your web application on.



SIMPLE EXAMPLE

```
// required libraries
var webdriverio = require('webdriverio'),
  should = require('should');

// a test script block or suite
describe('Title Test for Web Driver IO - Tutorial Test Page Website', function() {

  // set timeout to 10 seconds
  this.timeout(10000);
  var driver = {};

  // hook to run before tests
  before( function (done) {
    // load the driver for browser
    driver = webdriverio.remote({ desiredCapabilities: {browserName: 'firefox'} });
    driver.init(done);
  });

  // a test spec - "specification"
  it('should be load correct page and title', function () {
    // load page, then call function()
    return driver
      .url('http://www.tlkeith.com/WebDriverIOTutorialTest.html')
      // get title, then pass title to function()
      .getTitle().then( function (title) {
        // verify title
        (title).should.be.equal("Web Driver IO - Tutorial Test Page");
        // uncomment for console debug
        // console.log('Current Page Title: ' + title);
      });
  });

  // a "hook" to run after all tests in this block
  after(function(done) {
    driver.end(done);
  });
});
```

- Load the required libraries: web driver IO and should.
- Define the test suite. This suite it is called: "Title Test for Web Driver IO - Tutorial Test Page Website".
- Set the timeout to 10 seconds so the script doesn't timeout when loading the page.
- Hook to load the browser driver before running the specifications "specs". The Firefox driver is loaded in this example.
- Define the test specification.
- Load the website page.
- Verify the title using the should assertion library.
- Hook to quit and cleanup the driver when finished.

ANOTHER EXAMPLE

```
// a test script block or suite
describe('Loop Data Form Field Test for Web Driver IO - Tutorial Test Page Website', function() {

// data array - firstName and lastName
var dataArray = [
{"firstName" : "Tony", "lastName" : "Keith"},
{"firstName" : "John", "lastName" : "Doe"},
{"firstName" : "Jane", "lastName" : "Doe"},
{"firstName" : "Don", "lastName" : "Johnson"}
];
```

- Define the test suite. This suite it is called: “Loop Data Form Field Test for Web Driver IO - Tutorial Test Page Website”.
- Create static data array of JSON objects.

```
// loop through each dataArray
dataArray.forEach(function(d) {
  it('should populate fields, submit form, wait for results and verify data', function() {
    return driver
    // make sure you are on the starting page
    .url('http://www.tlkeith.com/WebDriverIOTutorialTest.html')
    .getTitle().then( function (title) {
      // verify title
      (title).should.be.equal("Web Driver IO - Tutorial Test Page");
    })
    .setValue("#fname", d.firstName)
    .getValue("#fname").then( function (e) {
      (e).should.be.equal(d.firstName);
      console.log("First Name: " + e);
    })
    .setValue("#lname", d.lastName)
    .getValue("#lname").then( function (e) {
      (e).should.be.equal(d.lastName);
      console.log("Last Name: " + e);
    })
    .submitForm("#search-form").then( function() {
      console.log('Submit Search Form');
    })
    .waitForVisible("#search-results", 10000).then(function () {
      console.log('Result Page Found');
    })
    .getText("//h1").then(function (link) {
      console.log('Text found: ' + link);
      (link).should.equal("Welcome " + d.firstName + " " + d.lastName + ".");
    });
  });
});
```

- Loop through each data array.
- Load test page.
- Set / Verify first name input field.
- Set / Verify last name input field.
- Submit form.
- Wait for results page to be loaded.
- Verify data on results page.

TESTING POSSIBILITIES

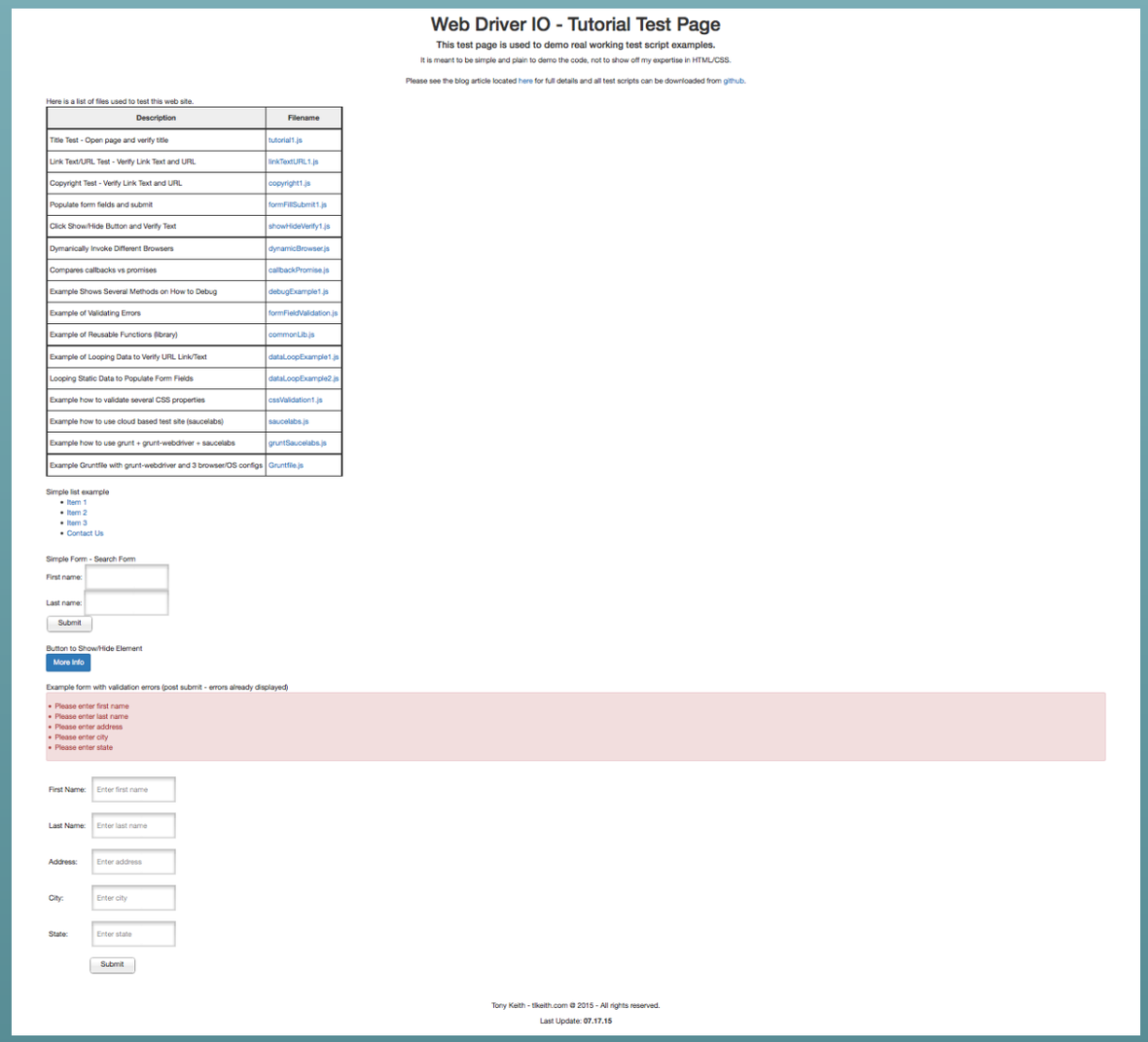
- **Verify elements on a page**
- **Verify URL and link text**
- **Verify header/footer/copyright text**
- **Populate form data and submit**
- **Validate form errors**
- **Validate CSS properties**
- **Reusable commands (library)**
- **Test applications on multiple OS/browsers combinations + mobile**

DEMO TIME

- Run tutorial1.js locally using mocha as the runner (Firefox only)
  - `$ mocha tutorial1.js`
- Run dataLoopExample2.js locally using mocha as the runner (Firefox only)
  - `$ mocha dataLoopExample2.js`
- Run dataLoopExample2.js remotely using grunt + grunt-webdriver as the runner on saucelabs (IE, Chrome, Firefox, iPhone)
  - `$ grunt —gruntfile Gruntfile-dataLoopExample2.js webdriver`

TUTORIAL & WORKING TEST SITE

- When I was researching the technologies, I didn't find many real-world examples.
- The examples either used a simple Google search or a Wikipedia search.
- So I created a tutorial with a working test site to run the scripts against.
- **Tutorial:** https://github.com/onewithhammer/web-driver-io-tutorial
- **Test Web Site:** http://www.tlkeith.com/WebDriverIOTutorialTest.html
- **My Blog:** http://webdriveriotutorial.blogspot.com/



Web Driver IO - Tutorial Test Page

This test page is used to demo real working test script examples.

It is meant to be simple and plain to demo the code, not to show off my expertise in HTML/CSS.

Please see the blog article located here for full details and all test scripts can be downloaded from github.

Here is a list of files used to test this web site.

Description	Filename
Title Test - Open page and verify title	tutorial1.js
Link Text/URL Test - Verify Link Text and URL	linkTextURL1.js
Copyright Test - Verify Link Text and URL	copyright1.js
Populate form fields and submit	formFillSubmit1.js
Click Show/Hide Button and Verify Text	showHideVerify1.js
Dymanically Invoke Different Browsers	dynamicBrowser.js
Compares callbacks vs promises	callbackPromise.js
Example Shows Several Methods on How to Debug	debugExample1.js
Example of Validating Errors	formFieldValidation.js
Example of Reusable Functions (library)	commonLib.js
Example of Looping Data to Verify URL Link/Text	dataLoopExample1.js
Looping Static Data to Populate Form Fields	dataLoopExample2.js
Example how to validate several CSS properties	cssValidation1.js
Example how to use cloud based test site (saucelabs)	saucelabs.js
Example how to use grunt + grunt-webdriver + saucelabs	gruntSaucelabs.js
Example Gruntfile with grunt-webdriver and 3 browser/OS configs	Gruntfile.js

Simple list example

- Item 1
- Item 2
- Item 3
- Contact Us

Simple Form - Search Form

First name:

Last name:

Submit

Button to Show/Hide Element

More Info

Example form with validation errors (post submit - errors already displayed)

- Please enter first name
- Please enter last name
- Please enter address
- Please enter city
- Please enter state

First Name: Enter first name

Last Name: Enter last name

Address: Enter address

City: Enter city

State: Enter state

Submit

Tony Keith - tlkeith.com @ 2015 - All rights reserved.

Last Update: **07.17.15**

INSTALLATION

- **Assumptions: Node and Java are installed.**
- **Install selenium server standalone:**
  - Create directory for selenium: **$ mkdir selenium**
  - Open browser and go to **http://www.seleniumhq.org/download/**
  - Download jar file. Save/move into the "**selenium**" directory.
- **Start Selenium Server :**
  - **$ java -jar selenium-server-standalone-2.47.1.jar**
- **Install Firefox (if not installed):**
- **Install Tutorial and required Software (mocha, should, web driver IO, grunt, grunt-webdriver):**
  - **$ git clone https://github.com/onewithhammer/web-driver-io-tutorial.git**
    - **OR**
  - Download and unzip from **https://github.com/onewithhammer/web-driver-io-tutorial**
  - **$ cd WebDriverIOTutorial**
  - **$ npm install** OR **$ sudo npm install**
- **Run the test scripts:**
  - **$ mocha tutorial1.js**

CONCLUSION

- I spent some time researching the technologies to use. I originally started with Selenium Web Driver but switched to using Web Driver IO. Web Driver IO seemed to be easier to use, better support and much easier to extend (at least the documentation for extending - reusable commands was better).

- When I first started looking at the technologies it was hard to find good examples that were relative to anything I was trying to do. This is the reason I wanted to share this information and knowledge with you.

- These technologies worked much better than I expected however there was learning curve involved. Once I understood how all the components worked together, I was able to write complicated test scripts in a very short time. The most difficult scripts were JS based components such as a date picker and modal selectors.

- I hope I educated you a little about E2E testing and provided enough information to motivate you into looking at these technologies.

THANK YOU

Special thanks to Node.js Cincy Meetup for organizing this meetup, The Brandery for allowing us to use this awesome space and Modulus for providing the pizza and beer.

I hope this presentation was useful in providing an Introduction to E2E Web Site Testing.

Please let me know if you have any questions.

REFERENCES

- **Tutorial on Github:** https://github.com/onewithhammer/web-driver-io-tutorial
- **My Blog:** http://webdriveriotutorial.blogspot.com/
- **Test Web Site:** http://tlkeith.com/WebDriverIOTutorialTest.html
- **Web Driver IO:** http://webdriver.io/
- **Mocha:** http://mochajs.org/
- **Should Assertion Library:** https://github.com/shouldjs/should.js
- **Grunt Task Runner:** http://gruntjs.com/
- **Grunt-WebDriver Plug-in:** http://webdriver.io/guide/plugins/grunt-webdriver.html
- **Selenium:** http://www.seleniumhq.org/
- **Saucelabs:** https://saucelabs.com
- **Browserstack:** https://www.browserstack.com/
- **Travis CI:** https://travis-ci.org/
- **Web Driver IO Gitter Room:** https://gitter.im/webdriverio/webdriverio
- **Mocha Gitter Room:** https://gitter.im/mochajs/mocha
- **Chai Assertion Library:** http://chaijs.com/
- **Appium Mobile Framework:** http://appium.io/
- **Phanton JS:** http://phantomjs.org/