Prerequisites

- git
- Node
- npm
- **bit.ly/ng-ptor**
- `npm install`
- protractor (`npm i protractor -g`)
- webdriver-manager update
- serve (`npm i serve -g`)
- **bit.ly/ng-ptor-slides**

bit.ly/ng-ptor-slides

Functional Testing with Protractor

Ben Clinkinbeard
twitter.com/bclinkinbeard
github.com/bclinkinbeard

1. **Concepts and components**
2. **Beyond the Basics**
3. **Going to Production**

Selenium

- Selenium 1.0
- Selenium RC
- WebDriver
- Selenium 1.0 + WebDriver = Selenium 2.0

Multiple language bindings

- Java
- C#
- Ruby
- Python
- JavaScript (Node)

How It Works

- Your tests call Protractor APIs
- Protractor extends/calls selenium-webdriver (running in Node)
- selenium-webdriver calls a (local or remote) Selenium server
- Selenium server calls a browser driver
- Browser driver uses your app ***as a user would***

How It Works

- **Your tests call Protractor APIs**

- **Protractor extends/calls selenium-webdriver (running in Node)**

- selenium-webdriver calls a (local or remote) Selenium server

- Selenium server calls a browser driver

- Browser driver uses your app as a user would

Terminology is a mess

- selenium-webdriver
  - https://www.npmjs.com/package/selenium-webdriver
  - Sometimes also referred to as WebdriverJS
  - Official JS bindings
- WebdriverIO
  - https://www.npmjs.com/package/webdriverio
  - Formerly known as webdriverjs
- wd
  - https://www.npmjs.com/package/wd

Protractor

- Best JS Selenium API available
- Literally extends selenium-webdriver
- Adds Angular-specific APIs
- Understands Angular's run loop
- Provides helpers

Async all the things

- Commands sent as HTTP calls
- Abstracted by WebDriver
- Abstracted even further by Protractor

```
element(by.css('button.myclass')).click();
```

- POST /session/:sessionId/execute_async

- POST /session/:sessionId/element

- POST /session/:sessionId/element/:id/click

```
var webdriver = require('selenium-webdriver'),
    By = require('selenium-webdriver').By,
    until = require('selenium-webdriver').until;

var driver = new webdriver.Builder()
  .forBrowser('firefox')
  .build();

driver.get('http://www.google.com/ncr');
driver.findElement(By.name('q')).sendKeys('webdriver');
driver.findElement(By.name('btnG')).click();
driver.wait(until.titleIs('webdriver - Google Search'), 1000);
driver.quit();
```

```
// browser configured externally

browser.get('http://www.google.com/ncr');
element(by.name('q')).sendKeys('protractor');
element(by.name('btnG')).click();

expect(browser.getTitle()).toBe('protractor - Google Search');
```

WebDriver Control Flow

- Each command adds a promise to the stack/chain
- Protractor adapts Jasmine to wait for empty stack
- Other test libs like can be used
  - Mocha with Chai as Promised
  - Cucumber

WebDriver Control Flow

```
browser.get('/signin');

var username = element(by.model('username'));
username.clear();
username.sendKeys('Jane Doe');

var name = element(by.binding('username'));
expect(name.getText()).toEqual('Jane Doe');

// NOTHING HAS HAPPENED YET
```

Protractor API

- browser
- element()
- element.all()
- by
- driver

API – Locators

- `by` extends `webdriver.By`
- Angular-specific locators
- `by.binding()`, `by.model()`, etc.

API – Element Methods

- `click()`
- `getText()`
- `sendKeys()`
- `clear()`
- `isPresent()`
- …

Config

protractor.conf.js

```
var config = {
  specs: [
    './e2e/**/*.spec.js'
  ],

  baseUrl: 'http://localhost:3000',

  browserName: 'chrome',
  directConnect: true
};

exports.config = config;
```

Config

protractor.conf.js

```
var config = {
  specs: [
    './e2e/**/*.spec.js'
  ],

  baseUrl: 'http://localhost:3000',

  seleniumAddress: 'http://localhost:4723/wd/hub',
  capabilities: {
    name: 'IE 9',
    browserName: 'internet explorer',
    version: '9.0',
    tags: ['ie']
  }
};

exports.config = config;
```

Config

```
var config = {
  specs: [
    './e2e/**/*.spec.js'
  ],

  baseUrl: 'http://localhost:3000',

  seleniumAddress: 'http://hub.browserstack.com/wd/hub',
  multiCapabilities: [
    {
      name: 'IE 9',
      browserName: 'internet explorer',
      version: '9.0',
      tags: ['ie']
    },
    {
      name: 'Firefox',
      browserName: 'firefox'
    }
  ]
};

exports.config = config;
```

Hands On

- **http://bit.ly/ng-ptor**
- `npm install`
- `git checkout exercise-1`
- `serve` to spin up http://localhost:3000/
- `npm test`
- `Verify app runs and tests pass`

Exercise #1

- Verify submit label is “Sign in”
- Verify the forgot password link leads to /forgot

- `git checkout exercise-1-answer`

Page Objects

- ~~Useful~~ Essential for maintainable test code

- Softens the blow of app changes

Page Objects

```
function IndexPage () {
  this.emailInput = element(by.model('user.email'));
  this.passwordInput = element(by.model('user.password'));
  this.submitBtn = element(by.buttonText('Sign in'));
  this.forgotLink = element(by.linkText('Forgot your password?'));

  this.get = function () {
    browser.get('/');
  };

  this.fillEmail = function (email) {
    this.emailInput.sendKeys(email);
  };

  this.getFilledEmail = function () {
    return this.emailInput.getAttribute('value');
  };

  this.fillPassword = function (password) {
    this.passwordInput.sendKeys(password);
  };
}

module.exports = IndexPage;
```

Page Objects

- `page.submitButton` **YES**
- `page.submitButton.click()` **YES**
- `page.clickSubmitButton()` **NO**

Advanced Config

- Reading CLI params
- Extracting browser definitions
- Extracting provider configs
- Testing multiple browsers

Extracted browsers

```
exports.chrome = {
  browserName: 'chrome'
};

exports.firefox = {
  browserName: 'firefox'
};

exports.safari = {
  browserName: 'safari'
};

exports.ie9 = {
  browserName: 'internet explorer',
  version: '9.0'
};

exports.ie10 = {
  browserName: 'internet explorer',
  version: '10.0'
};
```

```
var browsers = require('./browsers');

var config = {
  specs: [
    './e2e/**/*.spec.js'
  ],

  baseUrl: 'http://localhost:3000',

  capabilities: browsers.firefox,
  directConnect: true
};

exports.config = config;
```

Extracted browsers

```
exports.chrome = {
  browserName: 'chrome'
};

exports.firefox = {
  browserName: 'firefox'
};

exports.safari = {
  browserName: 'safari'
};

exports.ie9 = {
  browserName: 'internet explorer',
  version: '9.0'
};

exports.ie10 = {
  browserName: 'internet explorer',
  version: '10.0'
};
```

```
var browsers = require('./browsers');

var config = {
  specs: [
    './e2e/**/*.spec.js'
  ],

  baseUrl: 'http://localhost:3000',

  multiCapabilities: [
    browsers.chrome,
    browsers.firefox
  ],
  directConnect: true
};

exports.config = config;
```

Extracted browsers

```
exports.ie9 = {
  browserName: 'internet explorer',
  version: '9.0'
};

exports.ie10 = {
  browserName: 'internet explorer',
  version: '10.0'
};

exports.all = [
  exports.chrome,
  exports.firefox,
  exports.safari,
  exports.ie8,
  exports.ie9,
  exports.ie10,
  exports.ie11,
  exports.iOS7
];
```

```
var browsers = require('./browsers');

var config = {
  specs: [
    './e2e/**/*.spec.js'
  ],

  baseUrl: 'http://localhost:3000',

  multiCapabilities: browsers.all
};

exports.config = config;
```

Advanced Locators

```
element.all(by.binding('item.name')).first();

element(by.css('.list')).all(by.tagName('li')).get(2);
```

Hands On

- `git checkout exercise-2`
- Review `protractor.conf.js`
- Review `index.spec.js`
- Review `IndexPage.js`

Exercise #2

- Verify correct credentials lead to /landing
- Verify incorrect credentials display error state

- `git checkout exercise-2-answer`

Test suites

- Divide tests into subsets
- Good for organization
- Good for dev speed

Test suites

```
var config = {
  suites: {
    basics: './e2e/index.spec.js',
    signin: './e2e/signin.spec.js'
  },

  baseUrl: 'http://localhost:3000',

  multiCapabilities: [
    browsers.chrome,
    browsers.firefox
  ],
  directConnect: true
};

exports.config = config;
```

```
protractor protractor.conf.js --suite=signin
```

Screenshots

- browser.takeScreenshot().then(callback)
- Can run in afterEach
- Only on failure for debugging
- On all tests for visual regression testing

Screenshots

```
var fs = require('fs');

function capture (spec) {
  var name = spec.description.split(' ').join('_');
  name = name.split('/').join('');

  browser.takeScreenshot().then(function (png) {
    var stream = fs.createWriteStream('screenshots/' + name + '.png');
    stream.write(new Buffer(png, 'base64'));
    stream.end();
  });
}

exports.takeScreenshot = function (spec) {
  capture(spec);
};

exports.takeScreenshotOnFailure = function (spec) {
  if (spec.results().passed()) return;

  capture(spec);
};
```

Screenshots

```
var page = new IndexPage();

beforeEach(function () {
  page.get();
});

afterEach(function () {
  capture.takeScreenshot(jasmine.getEnv().currentSpec);
});

describe('basics', function () {
  it('should display the correct page title', function () {
    expect(browser.getTitle()).toBe('Sign In');
  });
```

Appium

- Mobile Safari
- Android
- Requires XCode, etc.
- Lots of device and OS combos
- Can be a pain
- Use the GUI

Providers

- Sauce Labs
- BrowserStack
- Others…

Why Providers?

- Managed Selenium infrastructure
- Hundreds of platforms
- Dashboards
- CI Integration

Hands On

- `git checkout exercise-3`

Exercise #3

- Divide tests into suites (one file per)
- Verify you can run suites individually
- Add `afterEach` that takes screenshots
- Note: must first create `screenshots` dir in project root

- `git checkout exercise-3-answer`