Software Test Professionals 2018!

WebDriverJS - Protractor

Peter Kim
Director of Quality Engineering
Kinetica

Hello

- Jennifer
- Peter

“Future You” in the next 2.5 hrs

- Basics of Node.js (nodejs.org)
- Test framework
- Selenium WebDriverJs
- Automate a real website
- Test Reporting

Goals ...

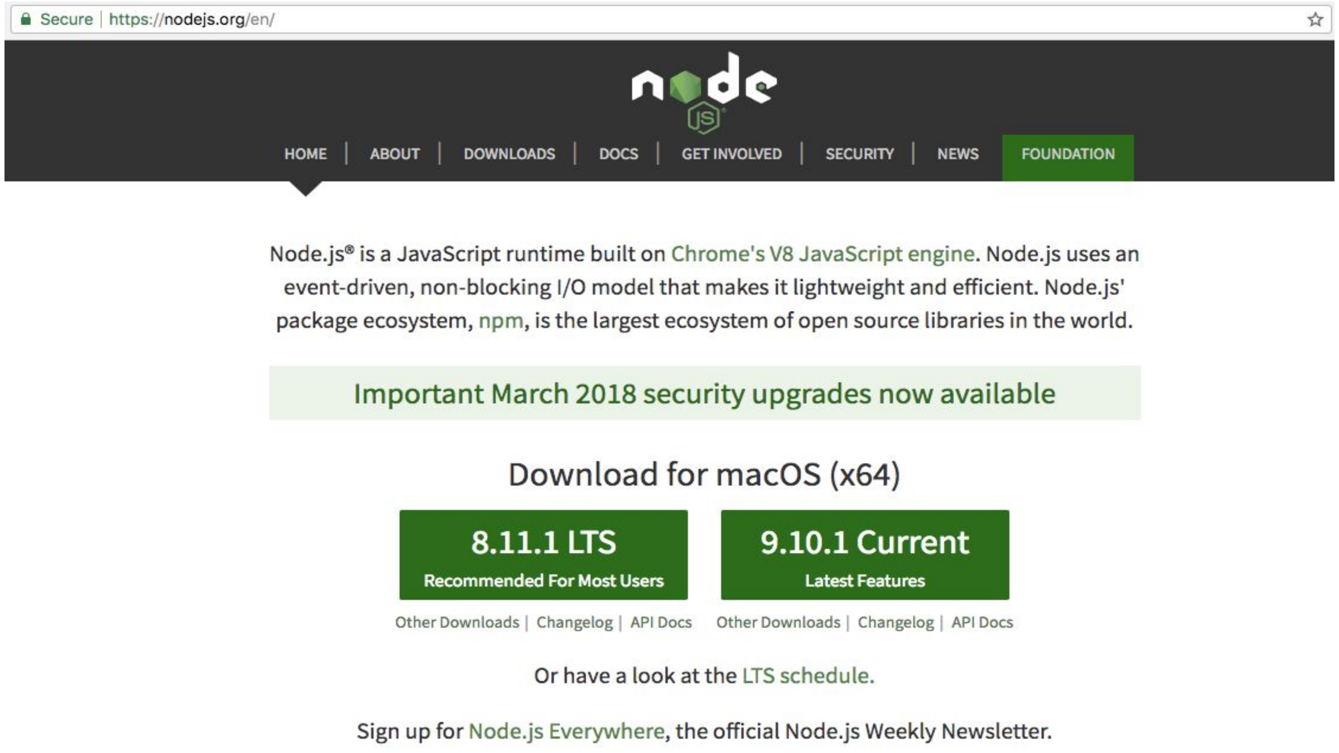
Nodejs (~30 min)
Installation
Basic programming
Promises
Builds
Examples

Test Frameworks (~30 min)
mochajs/jasmine/chai
Installation
Examples

Selenium-WebDriverjs (~60 min)
Protractor
Installation
Examples

Exercises (~30 min)
.. applying what you've learned.

Node.js::Installation – https://nodejs.org Mac OS



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Important March 2018 security upgrades now available

Download for macOS (x64)

8.11.1 LTS
Recommended For Most Users

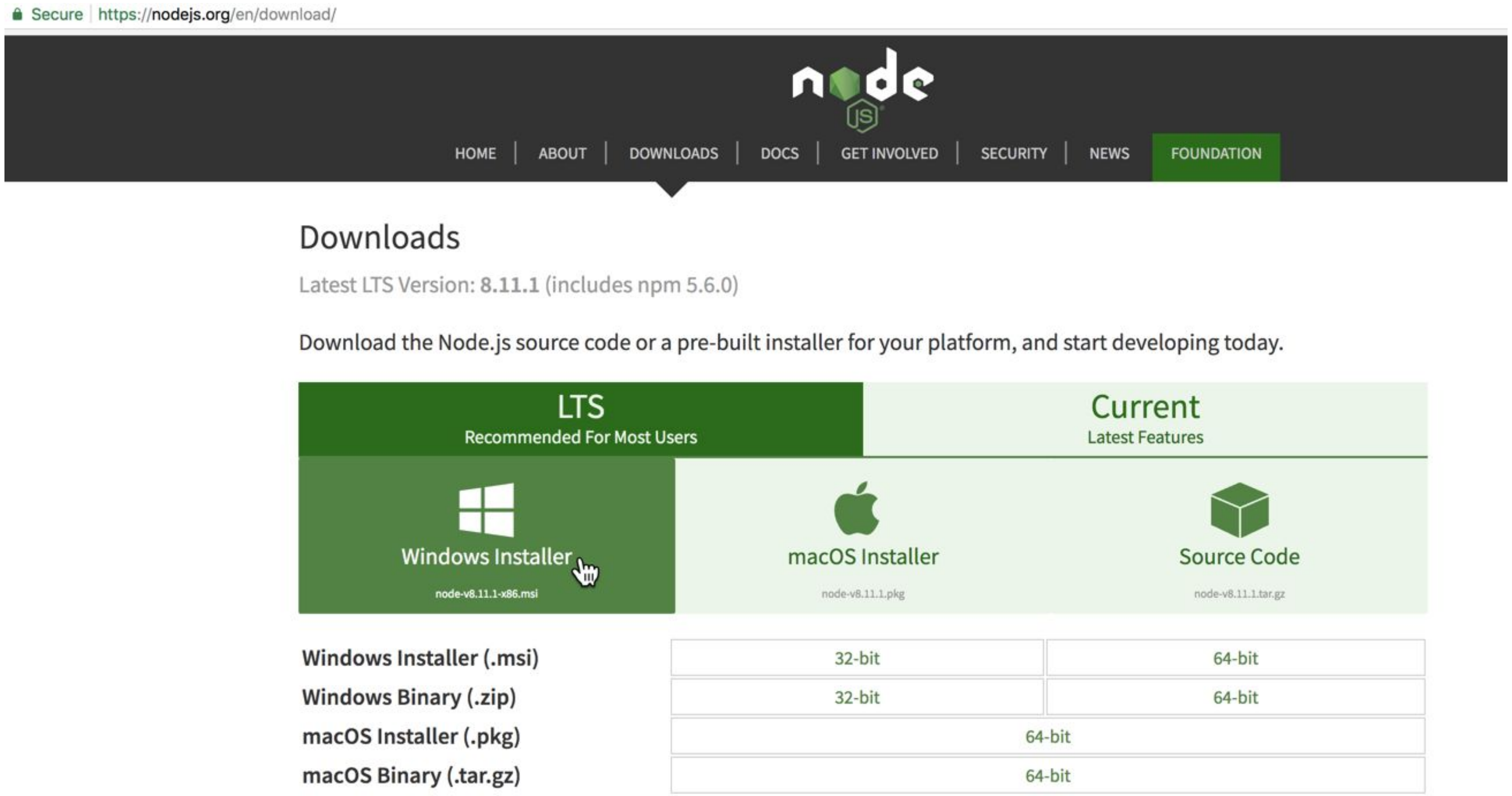
9.10.1 Current
Latest Features

Other Downloads | Changelog | API Docs   Other Downloads | Changelog | API Docs

Or have a look at the LTS schedule.

Sign up for Node.js Everywhere, the official Node.js Weekly Newsletter.

Node.js::Installation – https://nodejs.org Windows


Secure https://nodejs.org/en/download/
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS | FOUNDATION

Downloads

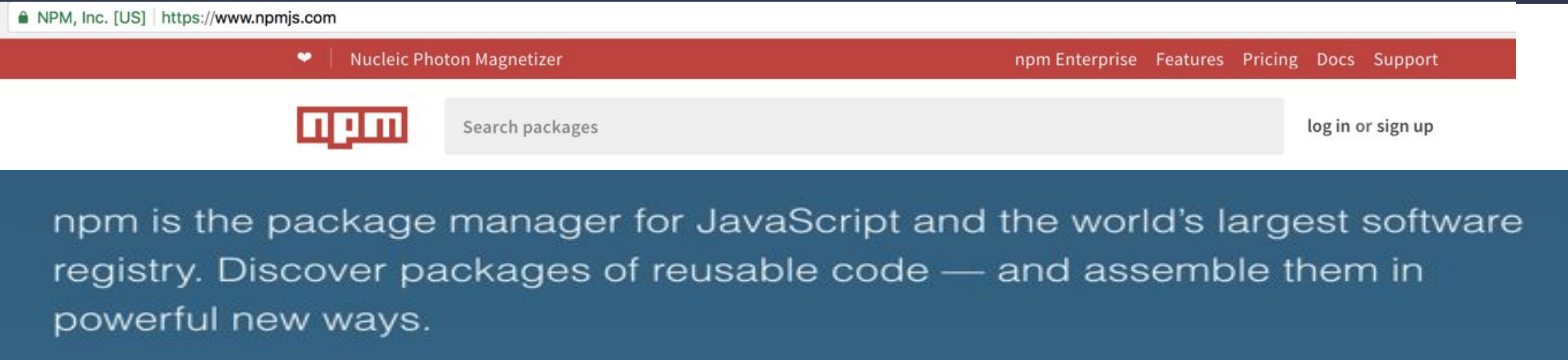
Latest LTS Version: **8.11.1** (includes npm 5.6.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users		Current Latest Features
Windows Installer node-v8.11.1-x86.msi	macOS Installer node-v8.11.1.pkg	Source Code node-v8.11.1.tar.gz

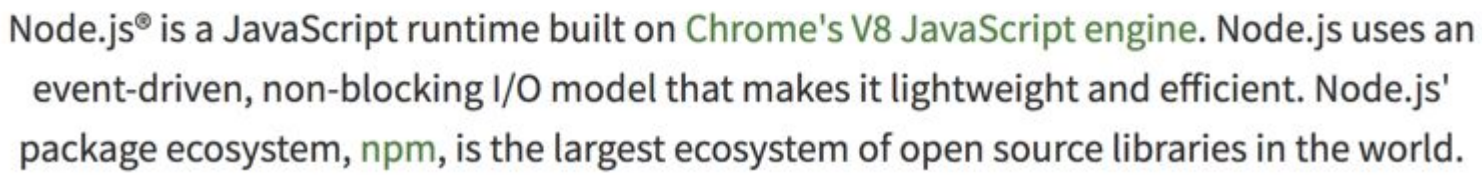
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	

Node.js::npm

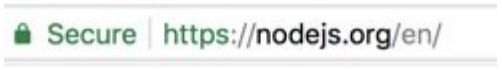

NPM, Inc. [US] | https://www.npmjs.com

npm is the package manager for JavaScript and the world's largest software registry. Discover packages of reusable code — and assemble them in powerful new ways.

Node.js::Basics

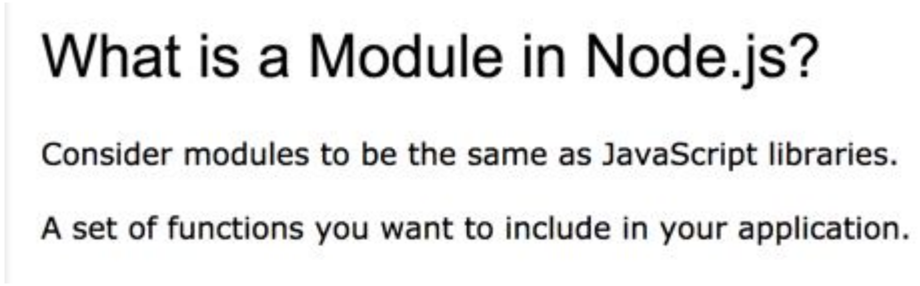


Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.



Secure | https://nodejs.org/en/

Node.js::Modules

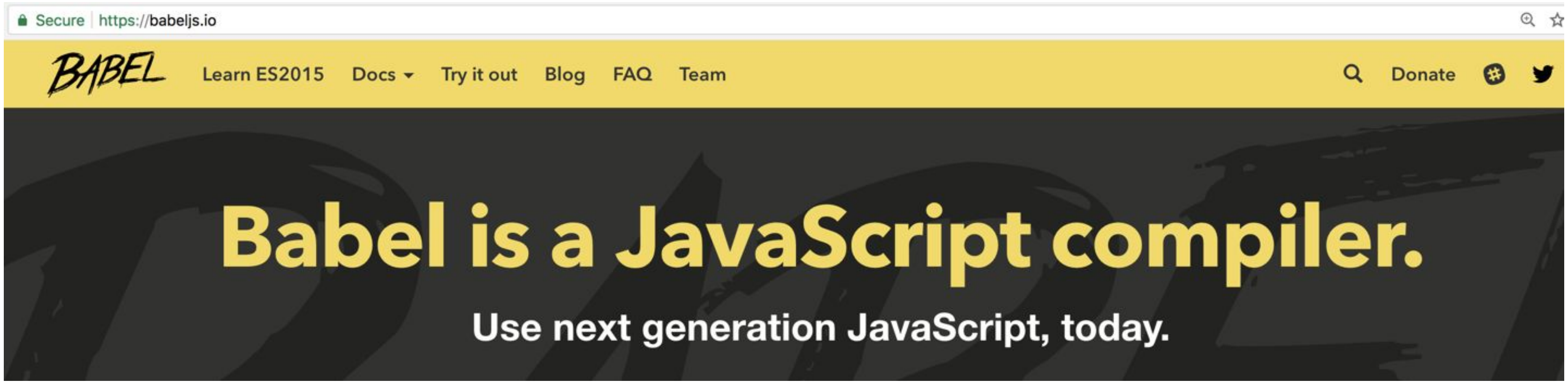


What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

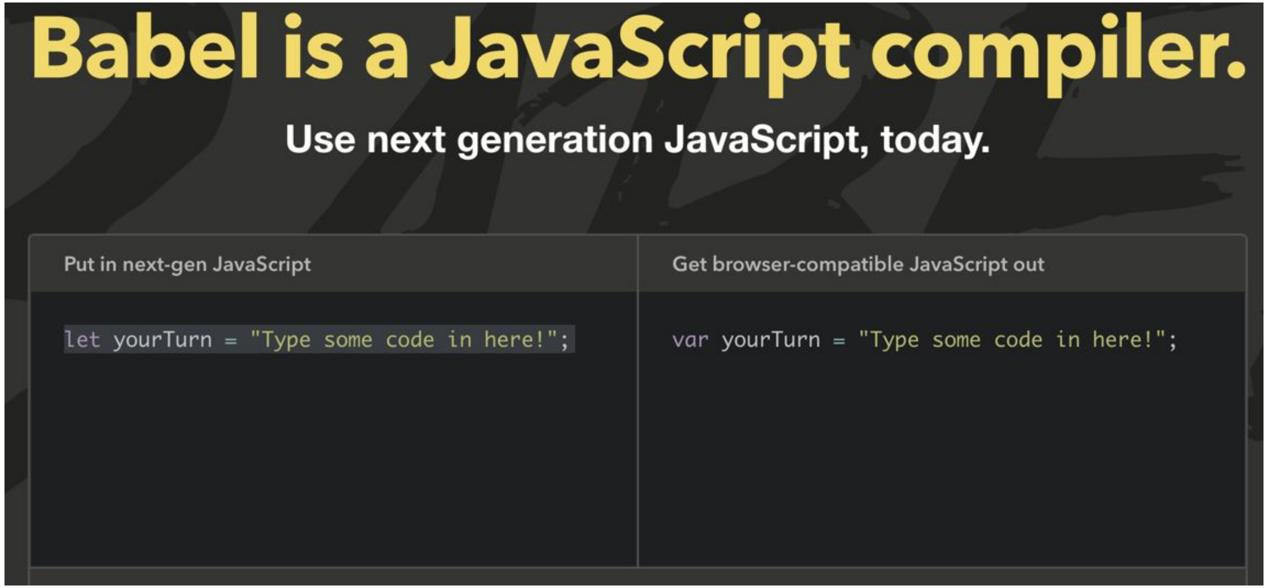
Node.js::Builds



Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Node.js::Builds::Compiling



Babel is a JavaScript compiler.

Use next generation JavaScript, today.

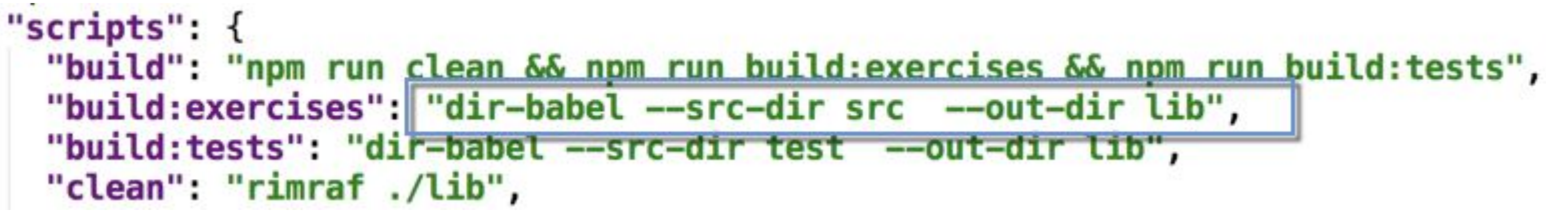
Put in next-gen JavaScript

```
let yourTurn = "Type some code in here!";
```

Get browser-compatible JavaScript out

```
var yourTurn = "Type some code in here!";
```

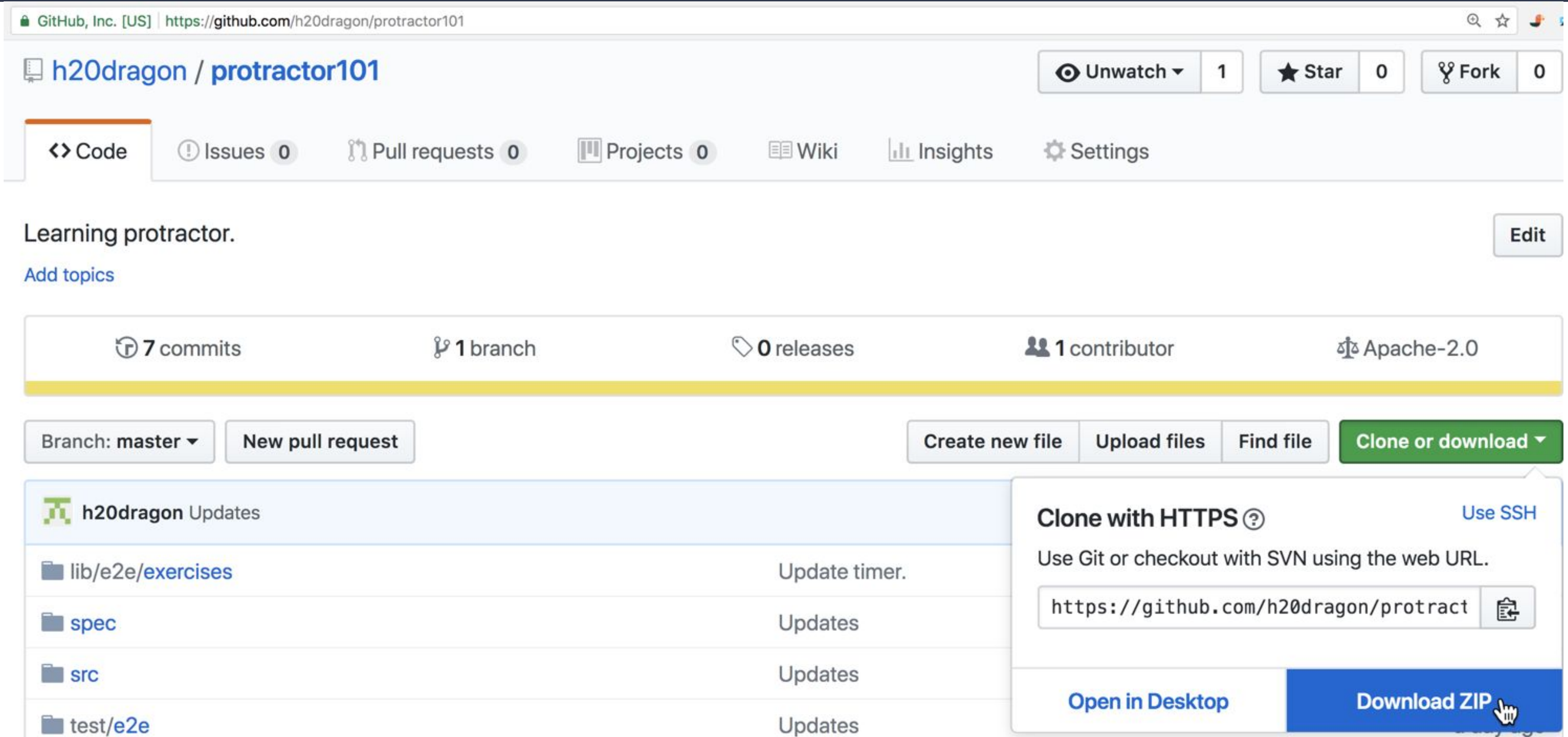
Node.js::Builds::Compiling



```
"scripts": {
  "build": "npm run clean && npm run build:exercises && npm run build:tests",
  "build:exercises": "dir-babel --src-dir src  --out-dir lib",
  "build:tests": "dir-babel --src-dir test  --out-dir lib",
  "clean": "rimraf ./lib",
```

Examples - Download source code

https://github.com/h20dragon/protractor101



Learning protractor.

h20dragon Updates

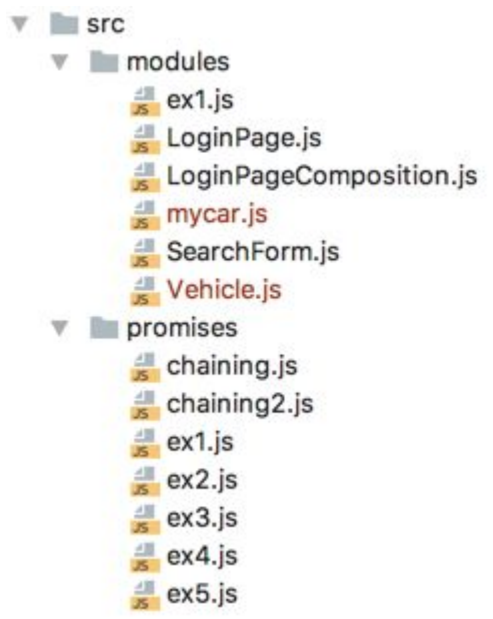
lib/e2e/exercises	Update timer.
spec	Updates
src	Updates
test/e2e	Updates

Clone with HTTPS

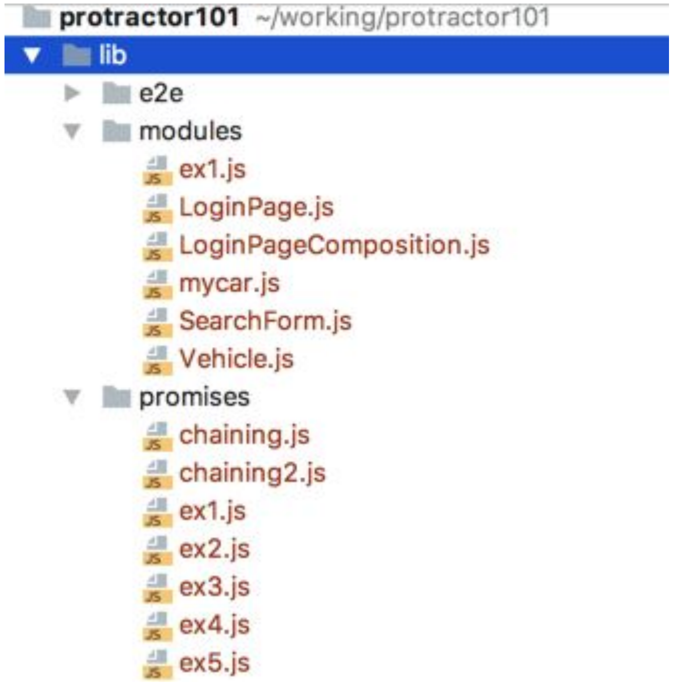
Use Git or checkout with SVN using the web URL.

```
https://github.com/h20dragon/protract
```

Node.js::Builds::Compiling (ES6)

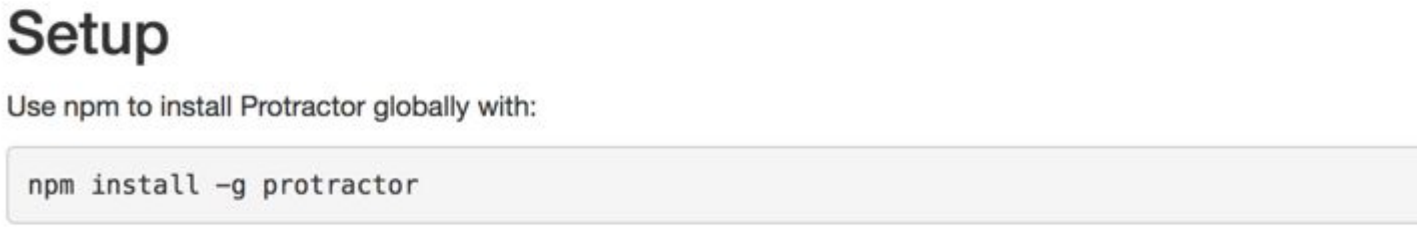


```
▼ src
  ▼ modules
      ex1.js
      LoginPage.js
      LoginPageComposition.js
      mycar.js
      SearchForm.js
      Vehicle.js
  ▼ promises
      chaining.js
      chaining2.js
      ex1.js
      ex2.js
      ex3.js
      ex4.js
      ex5.js
```



```
protractor101  ~/working/protractor101
▼ lib
  ▶ e2e
  ▼ modules
      ex1.js
      LoginPage.js
      LoginPageComposition.js
      mycar.js
      SearchForm.js
      Vehicle.js
  ▼ promises
      chaining.js
      chaining2.js
      ex1.js
      ex2.js
      ex3.js
      ex4.js
      ex5.js
```

Builds::Install Protractor



Setup

Use npm to install Protractor globally with:

```
npm install -g protractor
```

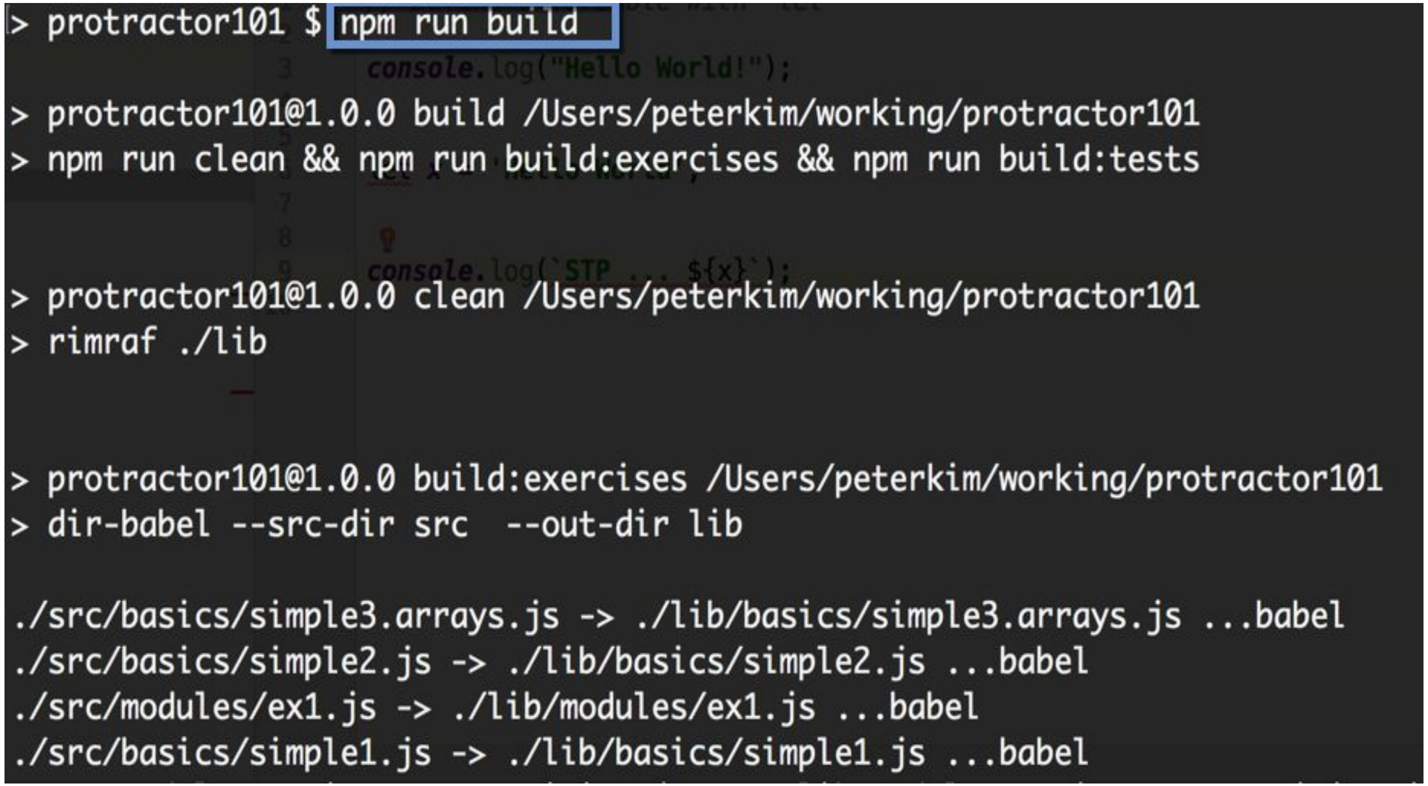
Builds::Let's install the dependencies

```
> protractor101 =>  npm install
```

Builds::Let’s install Selenium WebDriverjs

```
npm  run update
```

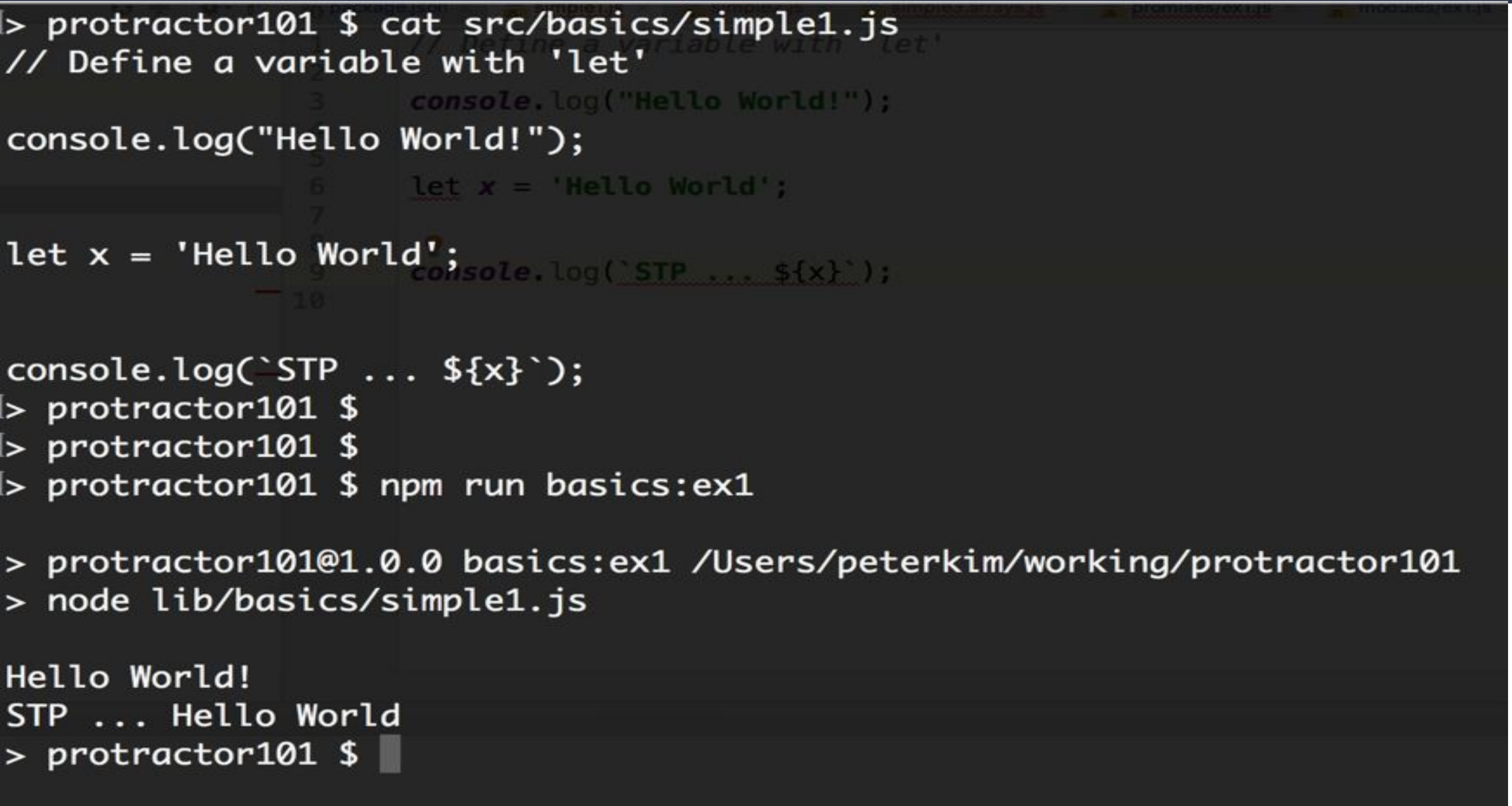
Builds::Let’s Build the Examples



```
> protractor101 $ npm run build

> protractor101@1.0.0 build /Users/peterkim/working/protractor101
> npm run clean && npm run build:exercises && npm run build:tests


> protractor101@1.0.0 clean /Users/peterkim/working/protractor101
> rimraf ./lib


> protractor101@1.0.0 build:exercises /Users/peterkim/working/protractor101
> dir-babel --src-dir src  --out-dir lib

./src/basics/simple3.arrays.js -> ./lib/basics/simple3.arrays.js ...babel
./src/basics/simple2.js -> ./lib/basics/simple2.js ...babel
./src/modules/ex1.js -> ./lib/modules/ex1.js ...babel
./src/basics/simple1.js -> ./lib/basics/simple1.js ...babel
```

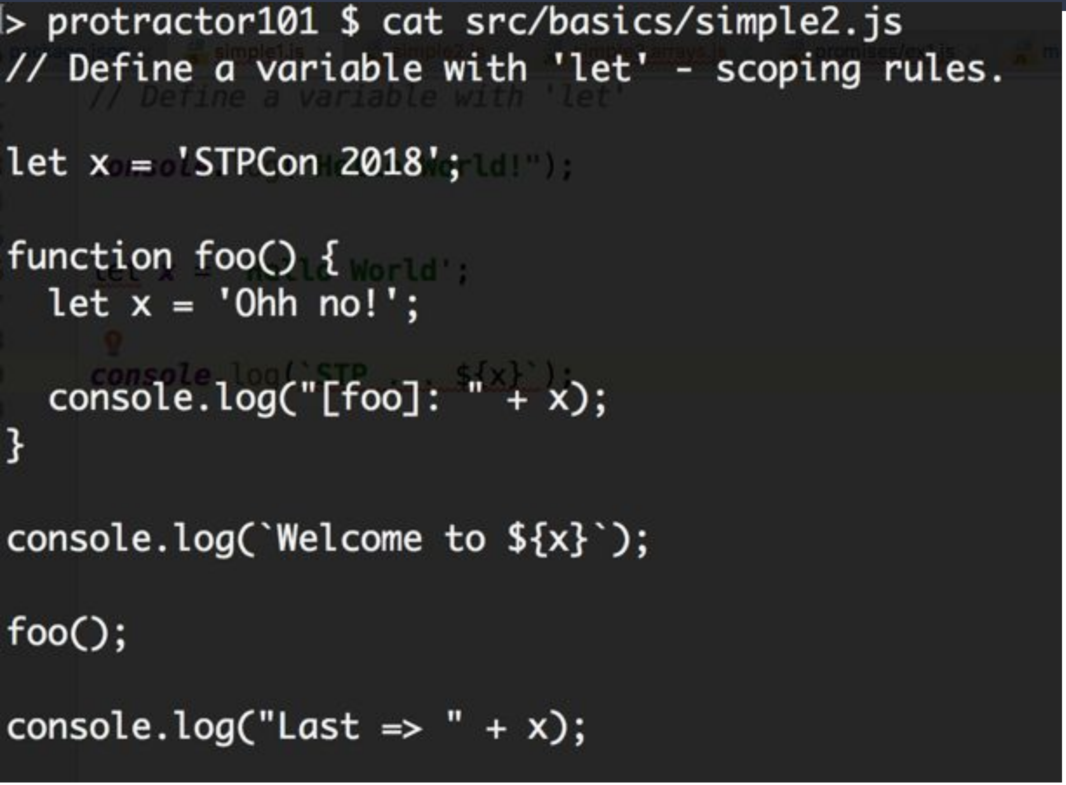
Node.js::Exercises

Node.js::Exercise 1:: npm run basics:ex1



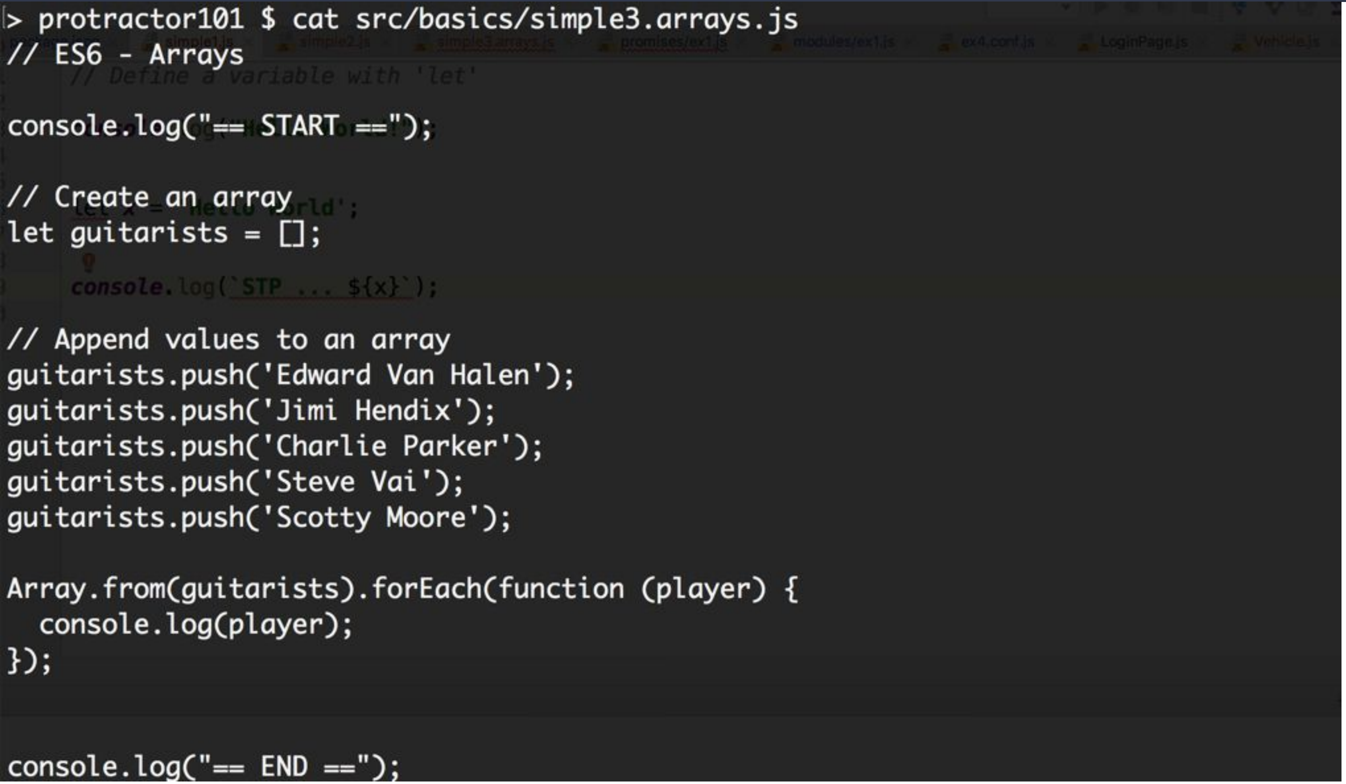
```
> protractor101 $ cat src/basics/simple1.js
// Define a variable with 'let'

console.log("Hello World!");


let x = 'Hello World';


console.log(`STP ... ${x}`);
> protractor101 $
> protractor101 $
> protractor101 $ npm run basics:ex1

> protractor101@1.0.0 basics:ex1 /Users/peterkim/working/protractor101
> node lib/basics/simple1.js

Hello World!
STP ... Hello World
> protractor101 $
```

Node.js::Exercise 2:: npm run basics:ex2



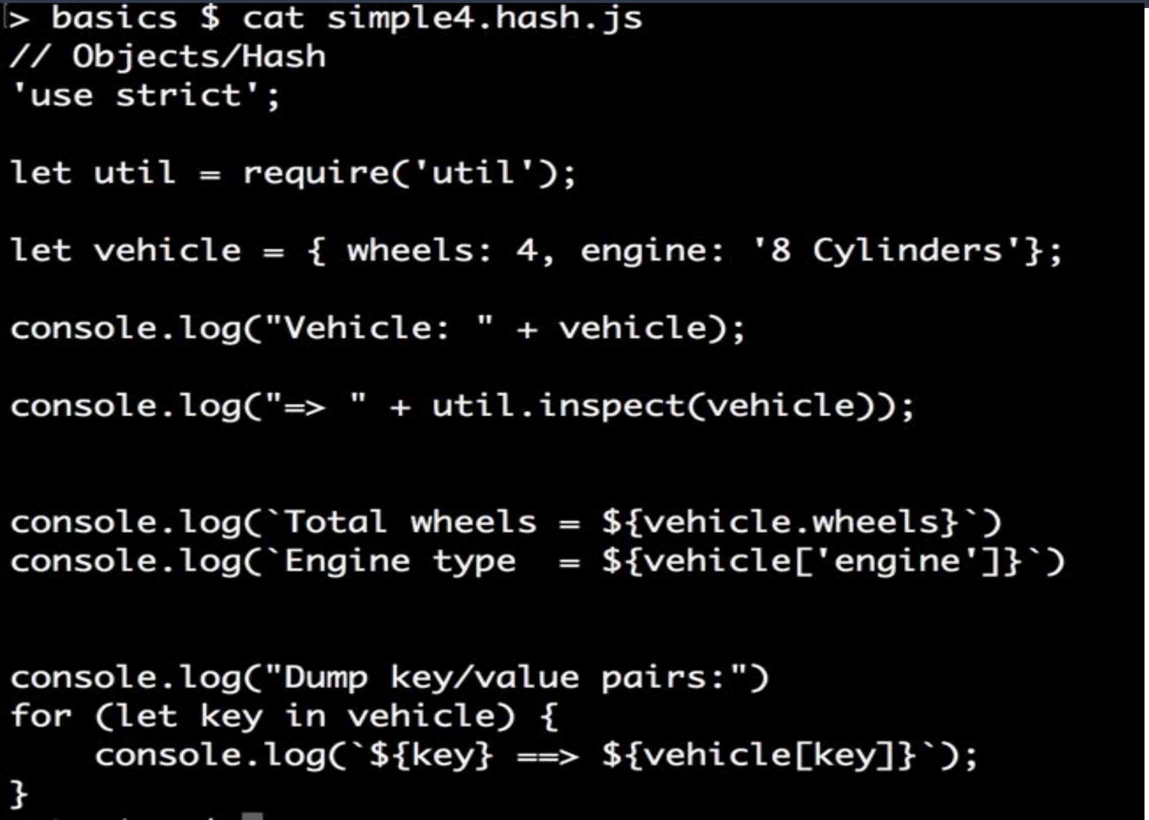
```
> protractor101 $ cat src/basics/simple2.js
// Define a variable with 'let' - scoping rules.

let x = 'STPCon 2018';

function foo() {
  let x = 'Ohh no!';

  console.log("[foo]: " + x);
}

console.log(`Welcome to ${x}`);

foo();

console.log("Last => " + x);
```

Node.js::Exercise 3:: npm run basics:ex3



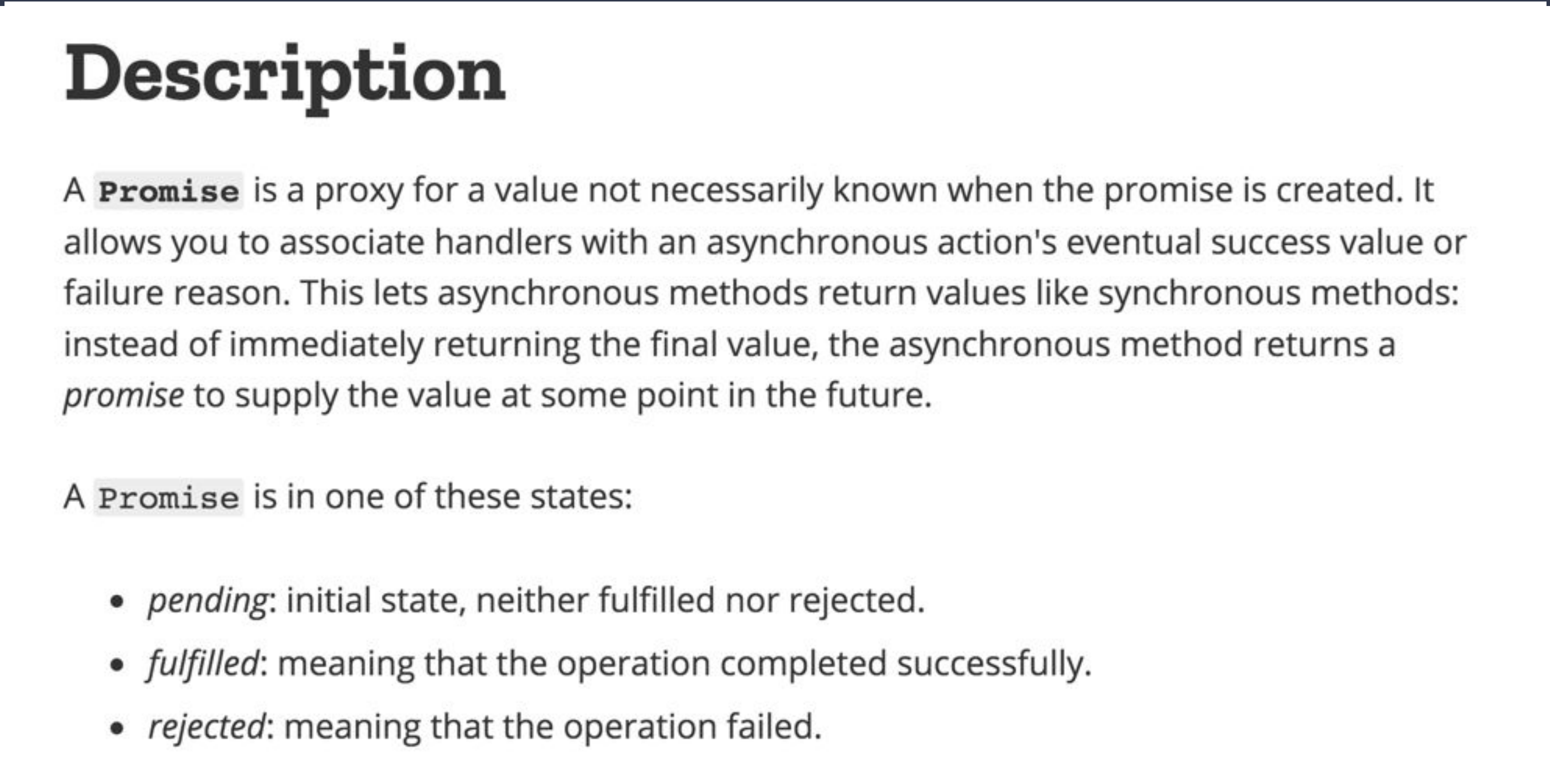
```
> protractor101 $ cat src/basics/simple3.arrays.js
// ES6 - Arrays

console.log("== START ==");

// Create an array
let guitarists = [];


// Append values to an array
guitarists.push('Edward Van Halen');
guitarists.push('Jimi Hendix');
guitarists.push('Charlie Parker');
guitarists.push('Steve Vai');
guitarists.push('Scotty Moore');

Array.from(guitarists).forEach(function (player) {
  console.log(player);
});


console.log("== END ==");
```

Node.js::Exercise 4:: npm run basics:ex4



```
> basics $ cat simple4.hash.js
// Objects/Hash
'use strict';

let util = require('util');

let vehicle = { wheels: 4, engine: '8 Cylinders'};

console.log("Vehicle: " + vehicle);

console.log("=> " + util.inspect(vehicle));


console.log(`Total wheels = ${vehicle.wheels}`)
console.log(`Engine type  = ${vehicle['engine']}`)


console.log("Dump key/value pairs:")
for (let key in vehicle) {
    console.log(`${key} ==> ${vehicle[key]}`);
}
```

Node.js::Promises



Description

A `Promise` is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with an asynchronous action's eventual success value or failure reason. This lets asynchronous methods return values like synchronous methods: instead of immediately returning the final value, the asynchronous method returns a *promise* to supply the value at some point in the future.

A `Promise` is in one of these states:

- *pending*: initial state, neither fulfilled nor rejected.
- *fulfilled*: meaning that the operation completed successfully.
- *rejected*: meaning that the operation failed.

https://developer.mozilla.org

Node.js::Promises::Create

```
let promise = new Promise(function(resolve, reject) {
        //
        // Execute Async call
        //

        if (/* all is well */) {
          resolve("My Result");
        }
        else {
          reject(Error("Yikes!"));
        }
});
```

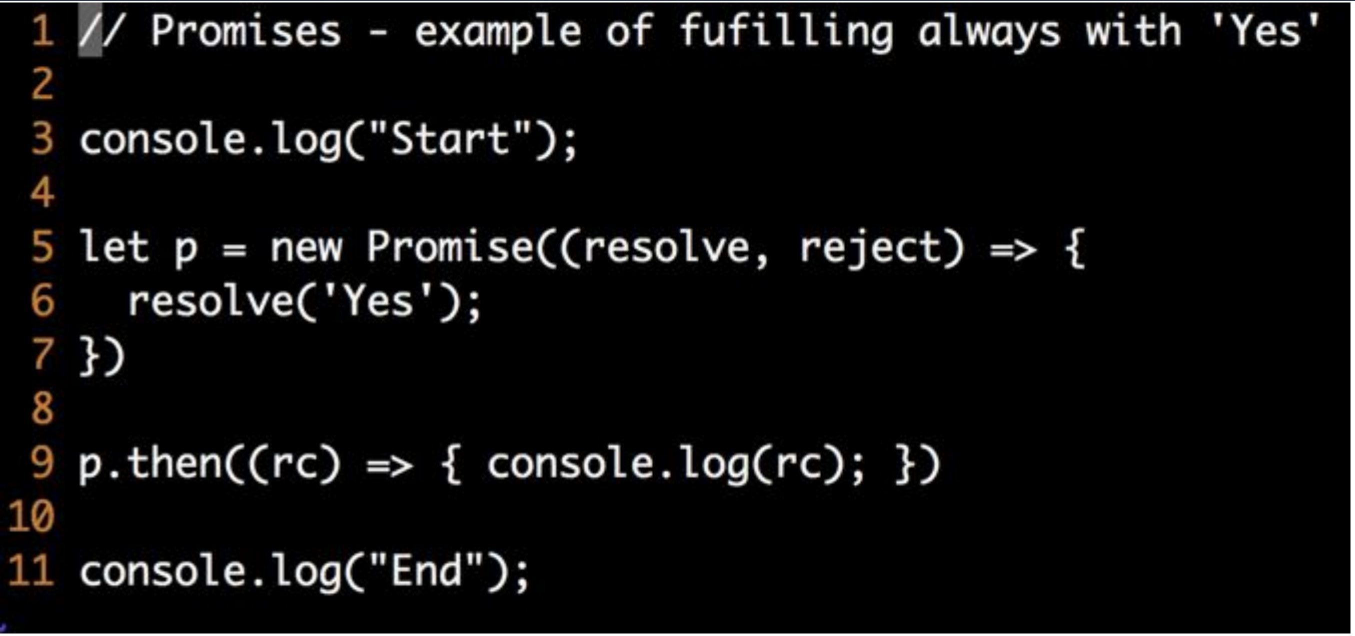
Node.js::Promises::Consume

```
promise.then(function(result) {
        console.log(result); // "My result"
       }, function(err) {
        console.log(err); // Error: Yikes!
});
```

```
// ES6 - Arrow functions
promise.then((result) => {
        console.log(result); // "My result"
       }, (err) => {
        console.log(err); // Error: "Yikes"
       }
);
```

Promises::Exercise 1

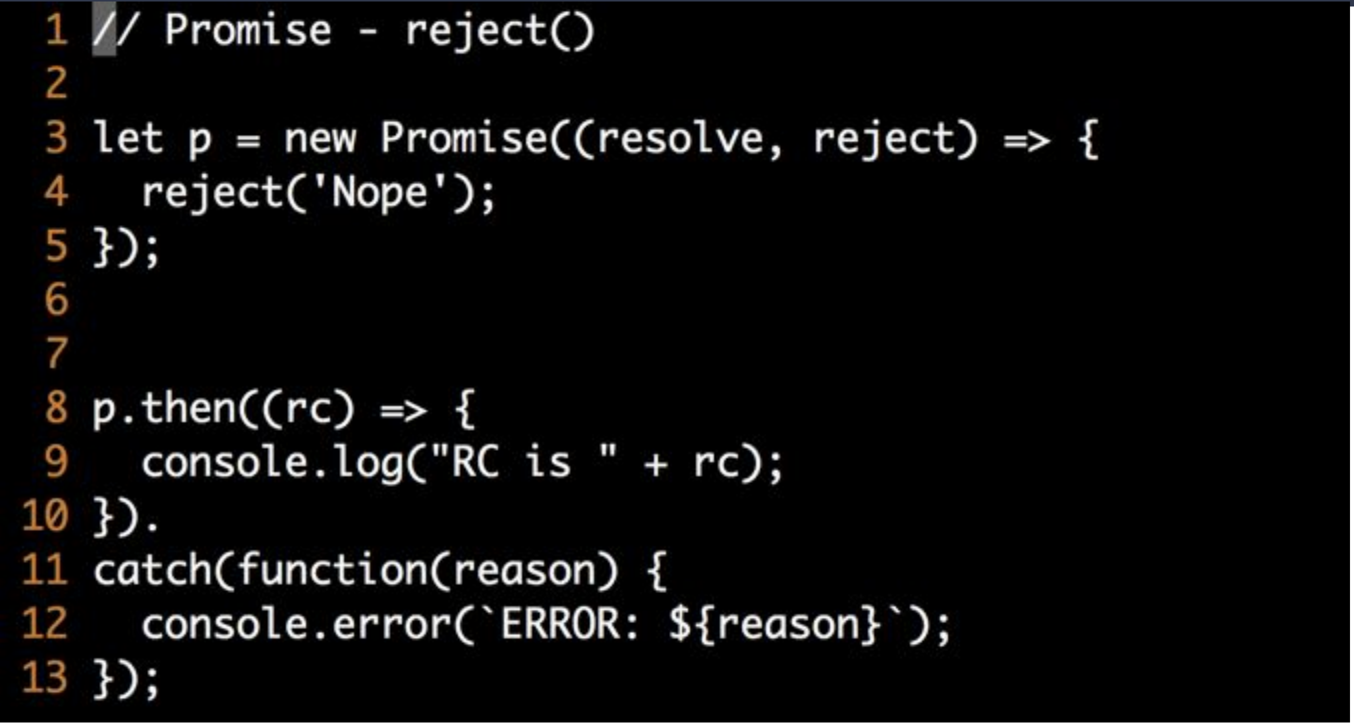
```
npm run promises:ex1
```



```
// Promises - example of fufilling always with 'Yes'

console.log("Start");

let p = new Promise((resolve, reject) => {
  resolve('Yes');
})

p.then((rc) => { console.log(rc); })

console.log("End");
```

Promises::Exercise 2

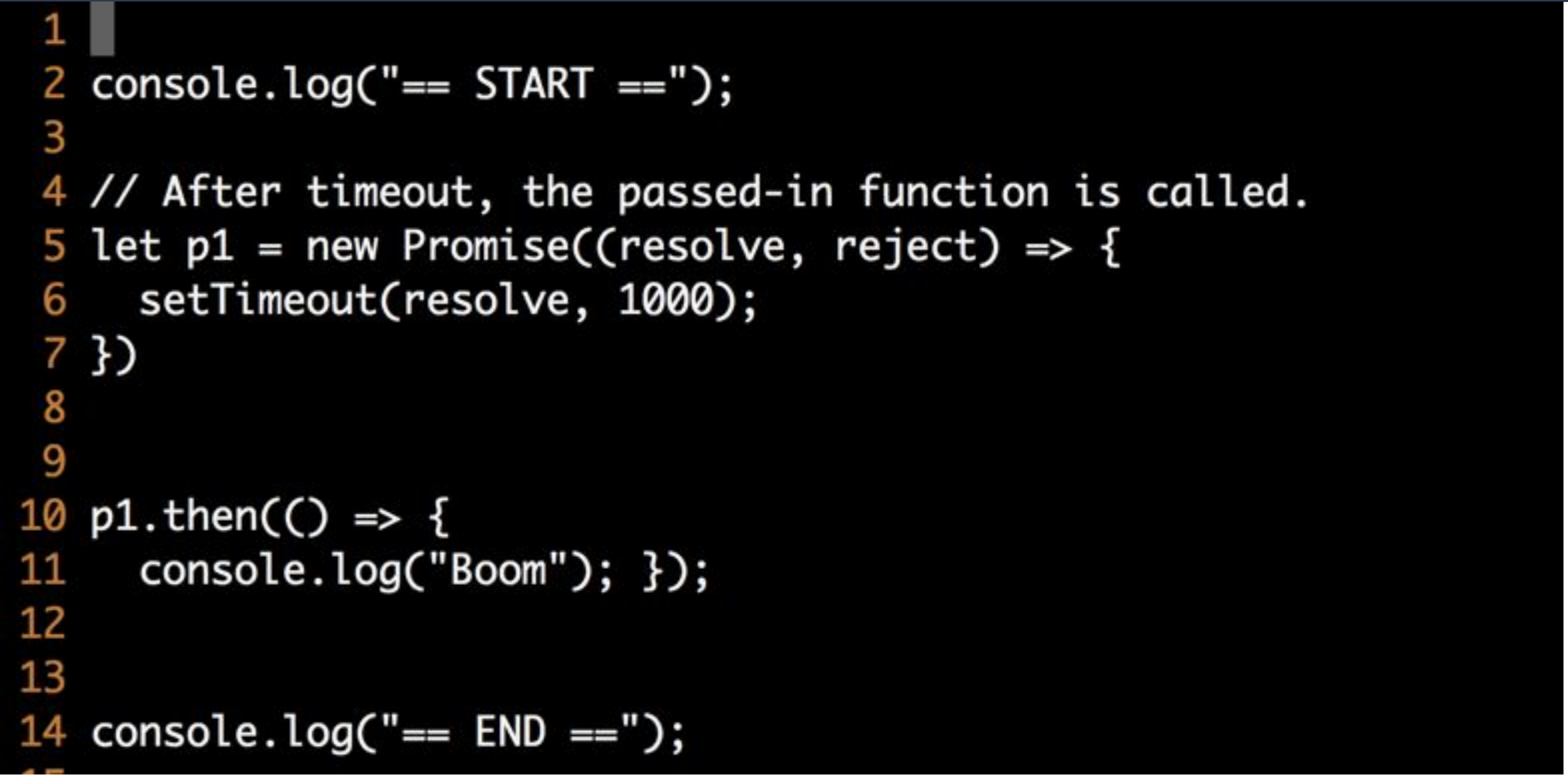
```
npm run promises:ex2
```



```
// Promise - reject()

let p = new Promise((resolve, reject) => {
  reject('Nope');
});


p.then((rc) => {
  console.log("RC is " + rc);
}).
catch(function(reason) {
  console.error(`ERROR: ${reason}`);
});
```

Promises::Exercise 3

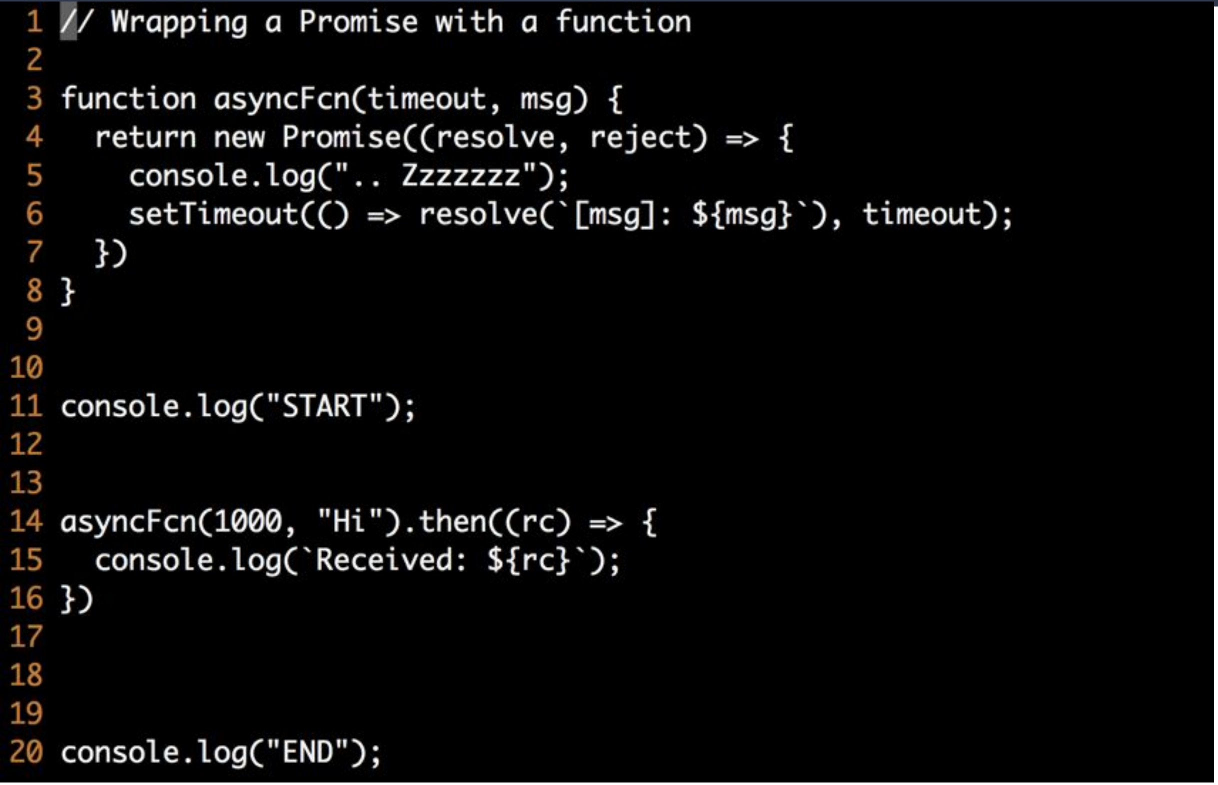
```
npm run promises:ex3
```



```

console.log("== START ==");

// After timeout, the passed-in function is called.
let p1 = new Promise((resolve, reject) => {
  setTimeout(resolve, 1000);
})


p1.then(() => {
  console.log("Boom"); });


console.log("== END ==");
```

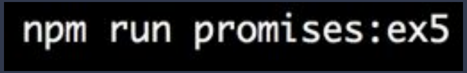
Promises::Exercise 4

```
npm run promises:ex4
```

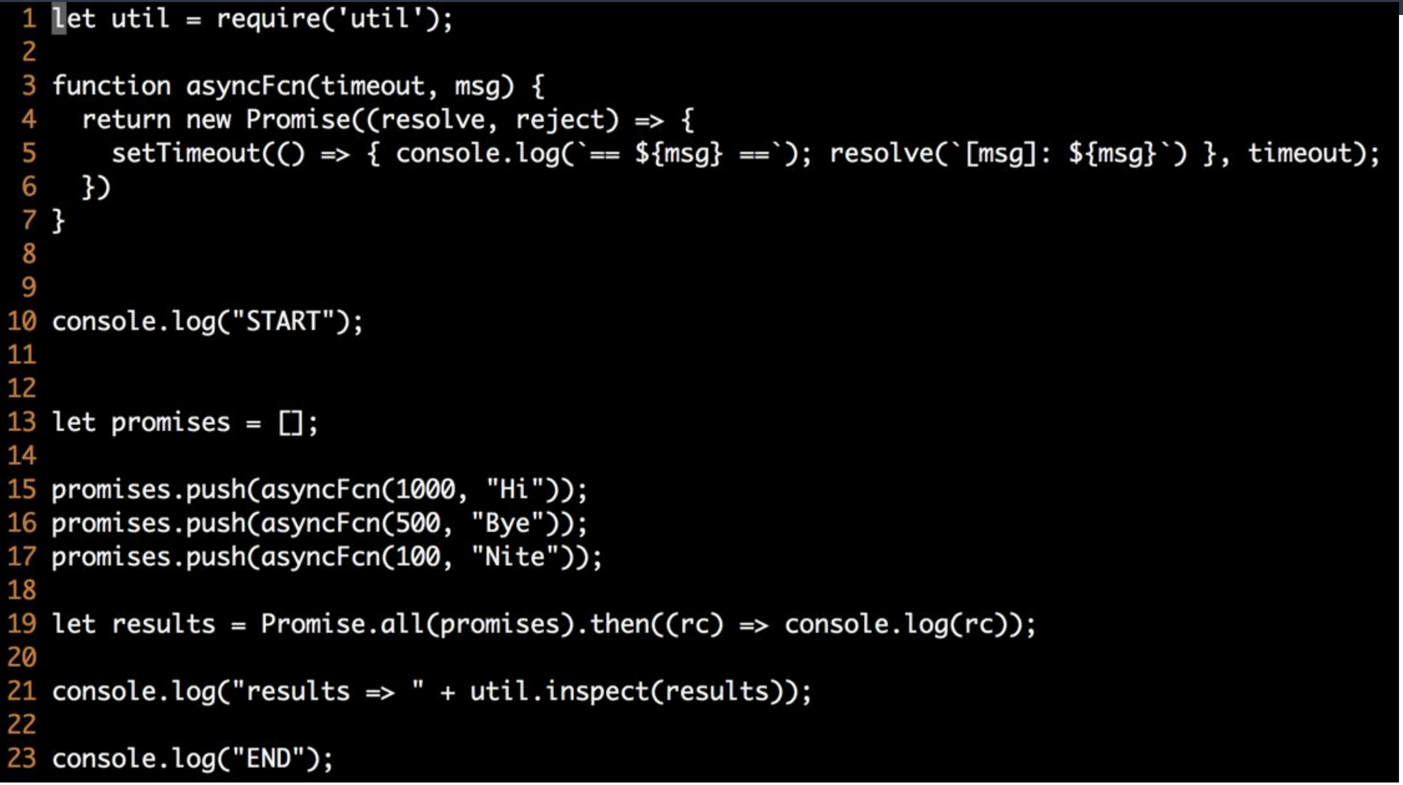


```
// Wrapping a Promise with a function

function asyncFcn(timeout, msg) {
  return new Promise((resolve, reject) => {
    console.log(".. Zzzzzzz");
    setTimeout(() => resolve(`[msg]: ${msg}`), timeout);
  })
}


console.log("START");


asyncFcn(1000, "Hi").then((rc) => {
  console.log(`Received: ${rc}`);
})



console.log("END");
```

Promises::Exercise 5

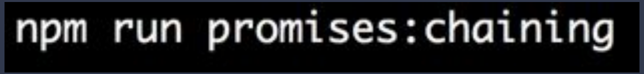


```
npm run promises:ex5
```

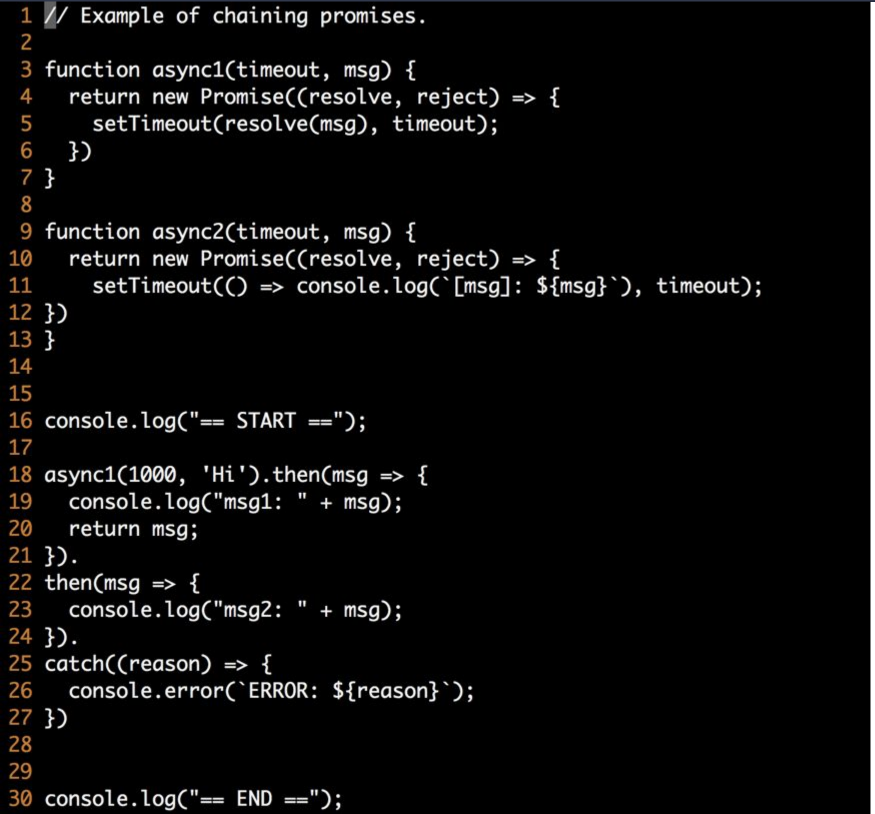


```
let util = require('util');

function asyncFcn(timeout, msg) {
  return new Promise((resolve, reject) => {
    setTimeout(() => { console.log(`== ${msg} ==`); resolve(`[msg]: ${msg}`) }, timeout);
  })
}


console.log("START");


let promises = [];

promises.push(asyncFcn(1000, "Hi"));
promises.push(asyncFcn(500, "Bye"));
promises.push(asyncFcn(100, "Nite"));

let results = Promise.all(promises).then((rc) => console.log(rc));

console.log("results => " + util.inspect(results));

console.log("END");
```

Promises::Exercise – Chaining

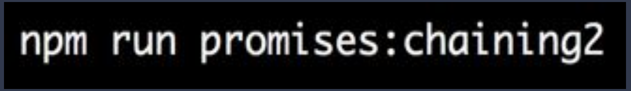


```
npm run promises:chaining
```

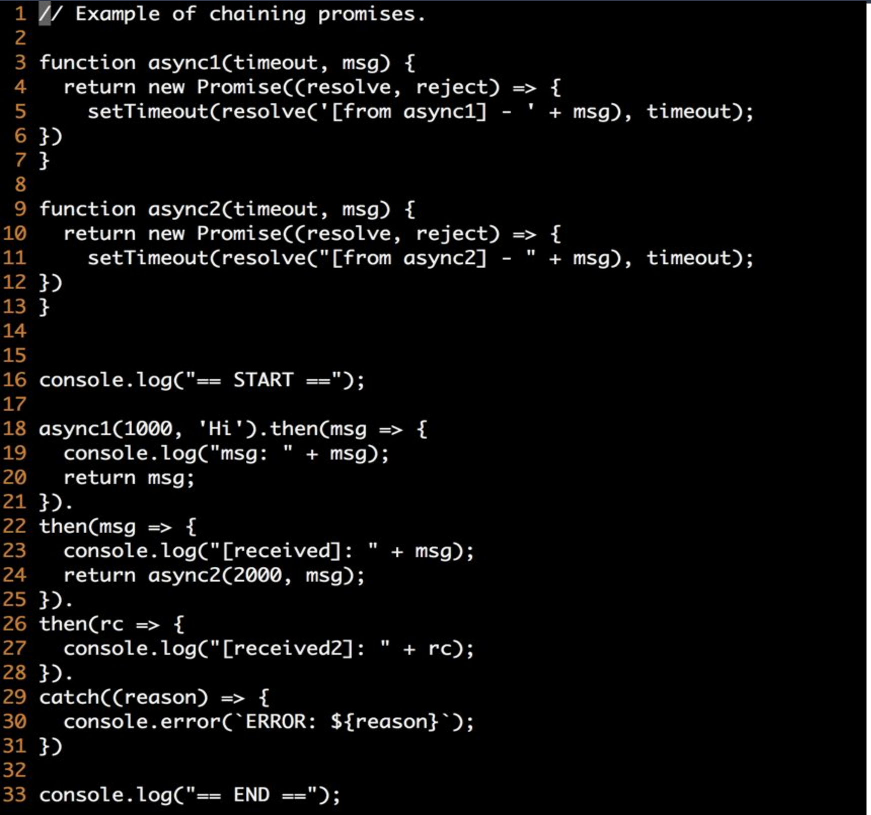


```
// Example of chaining promises.

function async1(timeout, msg) {
  return new Promise((resolve, reject) => {
    setTimeout(resolve(msg), timeout);
  })
}

function async2(timeout, msg) {
  return new Promise((resolve, reject) => {
    setTimeout(() => console.log(`[msg]: ${msg}`), timeout);
})
}


console.log("== START ==");

async1(1000, 'Hi').then(msg => {
  console.log("msg1: " + msg);
  return msg;
}).
then(msg => {
  console.log("msg2: " + msg);
}).
catch((reason) => {
  console.error(`ERROR: ${reason}`);
})


console.log("== END ==");
```

Promises::Exercise – Chaining 2



```
npm run promises:chaining2
```



```
// Example of chaining promises.

function async1(timeout, msg) {
  return new Promise((resolve, reject) => {
    setTimeout(resolve('[from async1] - ' + msg), timeout);
})
}

function async2(timeout, msg) {
  return new Promise((resolve, reject) => {
    setTimeout(resolve("[from async2] - " + msg), timeout);
})
}


console.log("== START ==");

async1(1000, 'Hi').then(msg => {
  console.log("msg: " + msg);
  return msg;
}).
then(msg => {
  console.log("[received]: " + msg);
  return async2(2000, msg);
}).
then(rc => {
  console.log("[received2]: " + rc);
}).
catch((reason) => {
  console.error(`ERROR: ${reason}`);
})

console.log("== END ==");
```

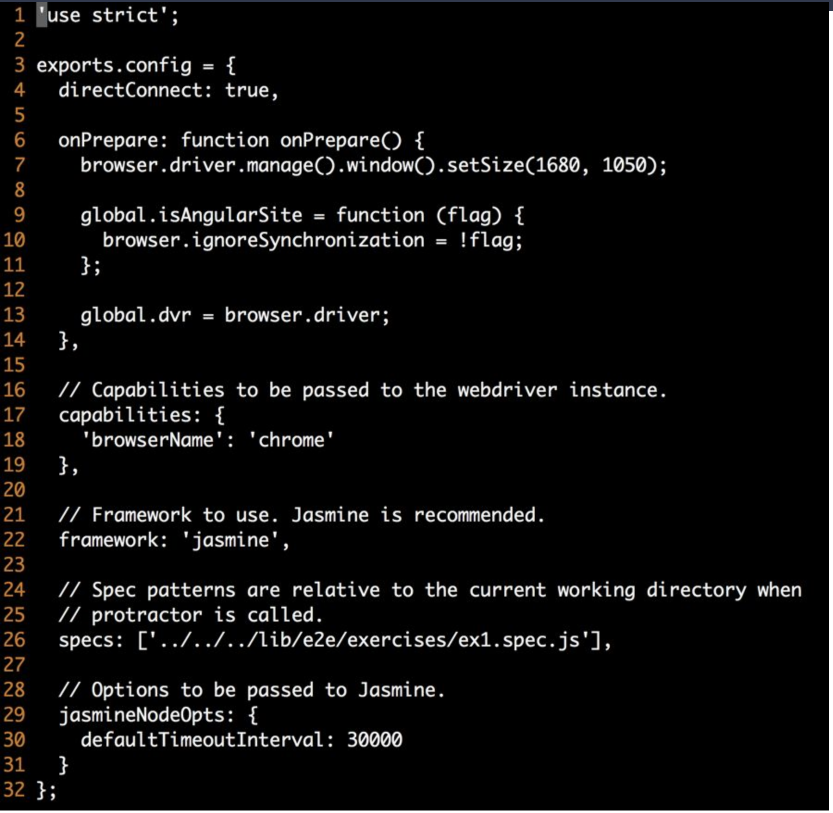
Test Frameworks

Test Framework::Protractor

Protractor is an end-to-end test framework for Angular and AngularJS applications. Protractor runs tests against your application running in a real browser, interacting with it as a user would.

Test Framework::Protractor::Run Tests::Setup

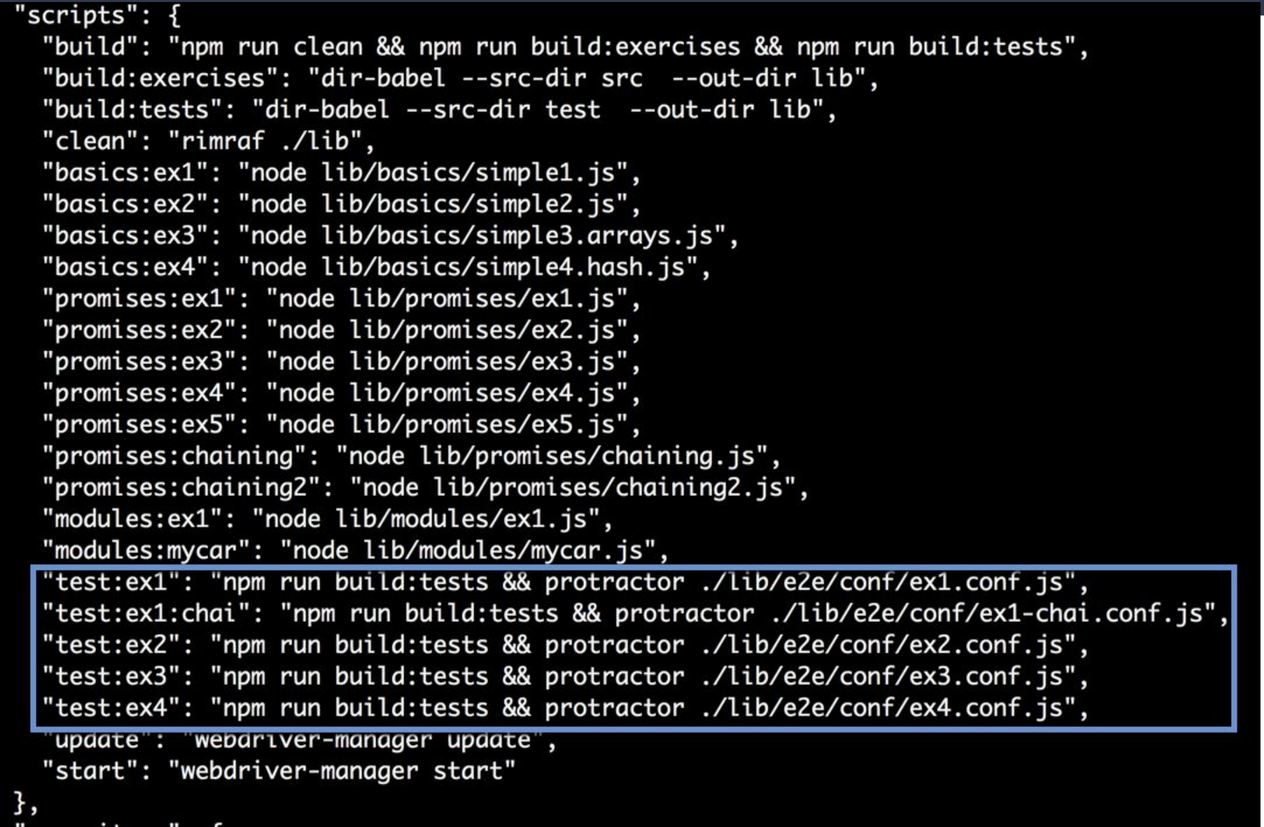
Create a configuration file - used to manage your automated Protractor tests.



```
'use strict';

exports.config = {
  directConnect: true,

  onPrepare: function onPrepare() {
    browser.driver.manage().window().setSize(1680, 1050);

    global.isAngularSite = function (flag) {
      browser.ignoreSynchronization = !flag;
    };

    global.dvr = browser.driver;
  },

  // Capabilities to be passed to the webdriver instance.
  capabilities: {
    'browserName': 'chrome'
  },

  // Framework to use. Jasmine is recommended.
  framework: 'jasmine',

  // Spec patterns are relative to the current working directory when
  // protractor is called.
  specs: ['../../../lib/e2e/exercises/ex1.spec.js'],

  // Options to be passed to Jasmine.
  jasmineNodeOpts: {
    defaultTimeoutInterval: 30000
  }
};
```

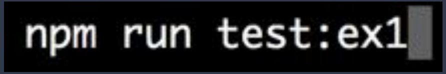
Test Framework::Protractor::Run Tests::Add Script

Add a ‘script’ to package.json

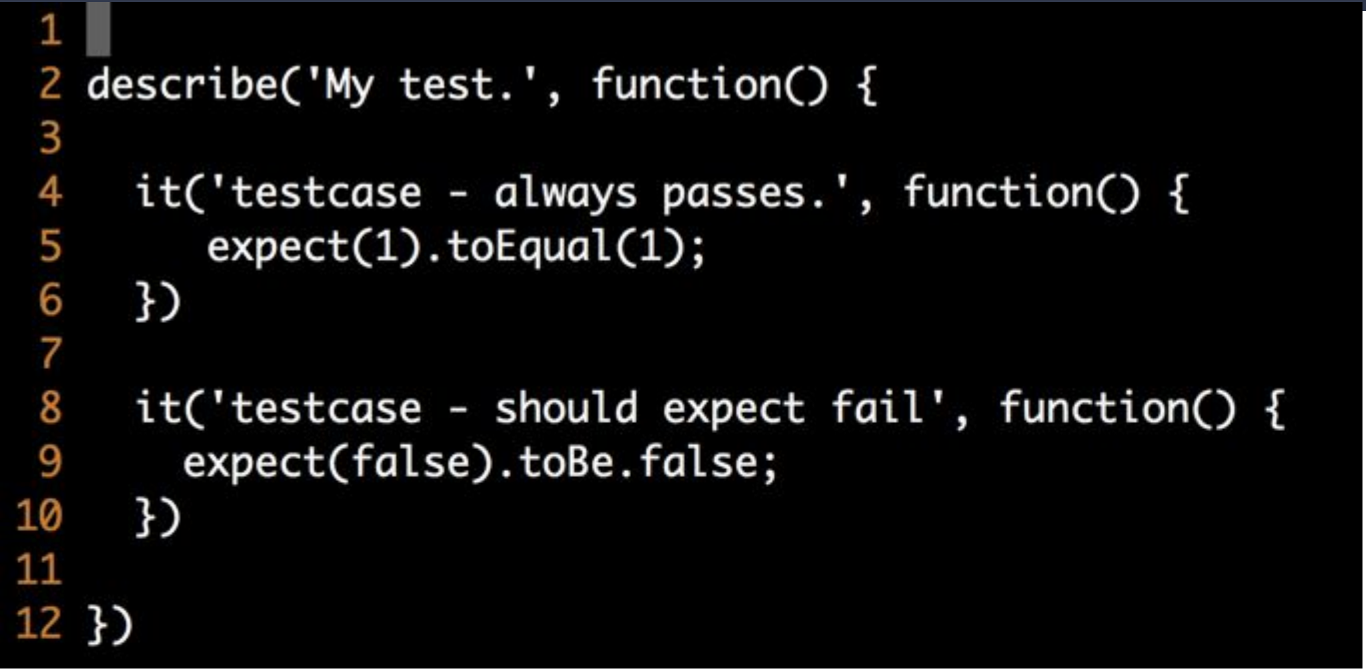


```
"scripts": {
  "build": "npm run clean && npm run build:exercises && npm run build:tests",
  "build:exercises": "dir-babel --src-dir src  --out-dir lib",
  "build:tests": "dir-babel --src-dir test  --out-dir lib",
  "clean": "rimraf ./lib",
  "basics:ex1": "node lib/basics/simple1.js",
  "basics:ex2": "node lib/basics/simple2.js",
  "basics:ex3": "node lib/basics/simple3.arrays.js",
  "basics:ex4": "node lib/basics/simple4.hash.js",
  "promises:ex1": "node lib/promises/ex1.js",
  "promises:ex2": "node lib/promises/ex2.js",
  "promises:ex3": "node lib/promises/ex3.js",
  "promises:ex4": "node lib/promises/ex4.js",
  "promises:ex5": "node lib/promises/ex5.js",
  "promises:chaining": "node lib/promises/chaining.js",
  "promises:chaining2": "node lib/promises/chaining2.js",
  "modules:ex1": "node lib/modules/ex1.js",
  "modules:mycar": "node lib/modules/mycar.js",
  "test:ex1": "npm run build:tests && protractor ./lib/e2e/conf/ex1.conf.js",
  "test:ex1:chai": "npm run build:tests && protractor ./lib/e2e/conf/ex1-chai.conf.js",
  "test:ex2": "npm run build:tests && protractor ./lib/e2e/conf/ex2.conf.js",
  "test:ex3": "npm run build:tests && protractor ./lib/e2e/conf/ex3.conf.js",
  "test:ex4": "npm run build:tests && protractor ./lib/e2e/conf/ex4.conf.js",
  "update": "webdriver-manager update",
  "start": "webdriver-manager start"
},
```

Test Framework::Protractor::Run Tests::Execute

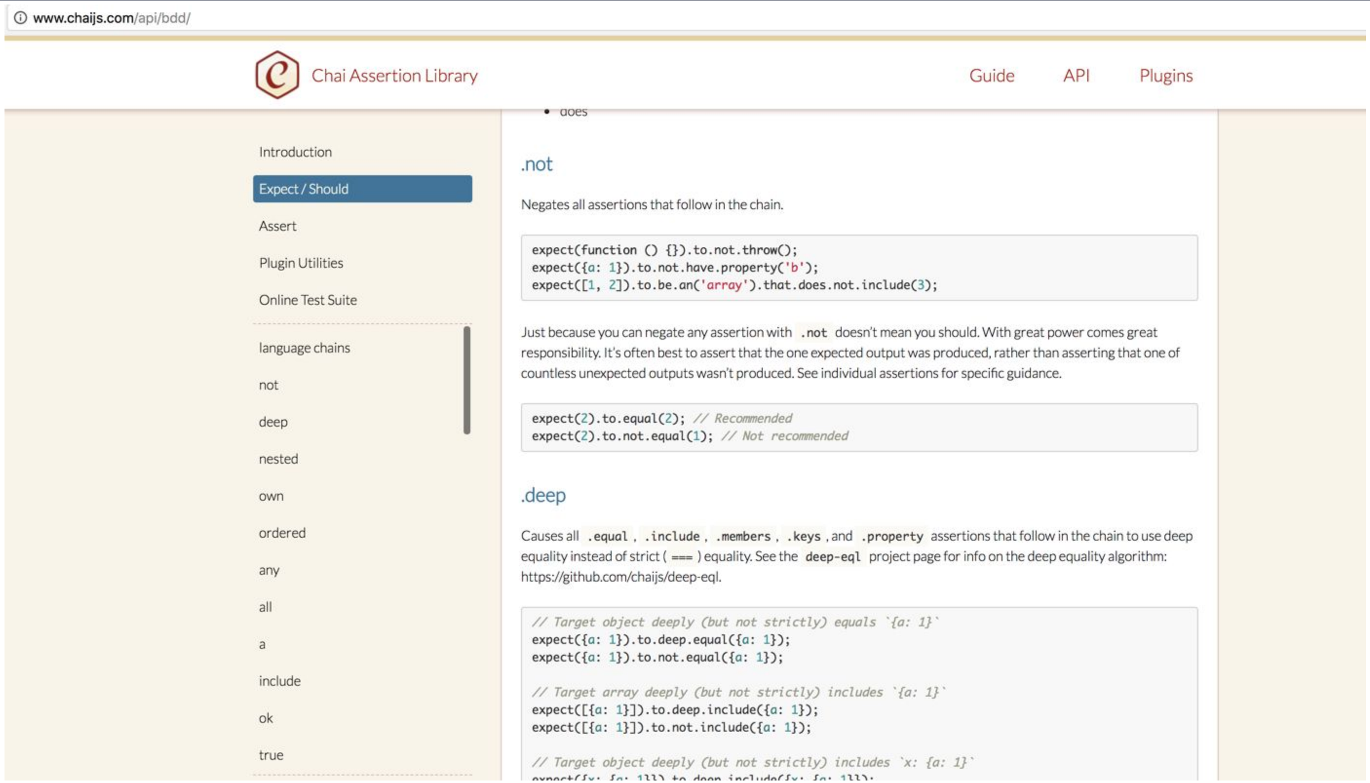


```
npm run test:ex1
```



```

describe('My test.', function() {

  it('testcase - always passes.', function() {
     expect(1).toEqual(1);
  })

  it('testcase - should expect fail', function() {
    expect(false).toBe.false;
  })

})
```

Test Framework::Assertion Library with Chai



www.chaijs.com/api/bdd/

- does

.not

Negates all assertions that follow in the chain.

```
expect(function () {}).to.not.throw();
expect({a: 1}).to.not.have.property('b');
expect([1, 2]).to.be.an('array').that.does.not.include(3);
```

Just because you can negate any assertion with `.not` doesn't mean you should. With great power comes great responsibility. It's often best to assert that the one expected output was produced, rather than asserting that one of countless unexpected outputs wasn't produced. See individual assertions for specific guidance.

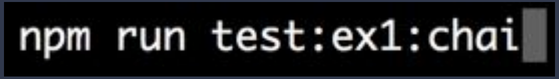
```
expect(2).to.equal(2); // Recommended
expect(2).to.not.equal(1); // Not recommended
```

.deep

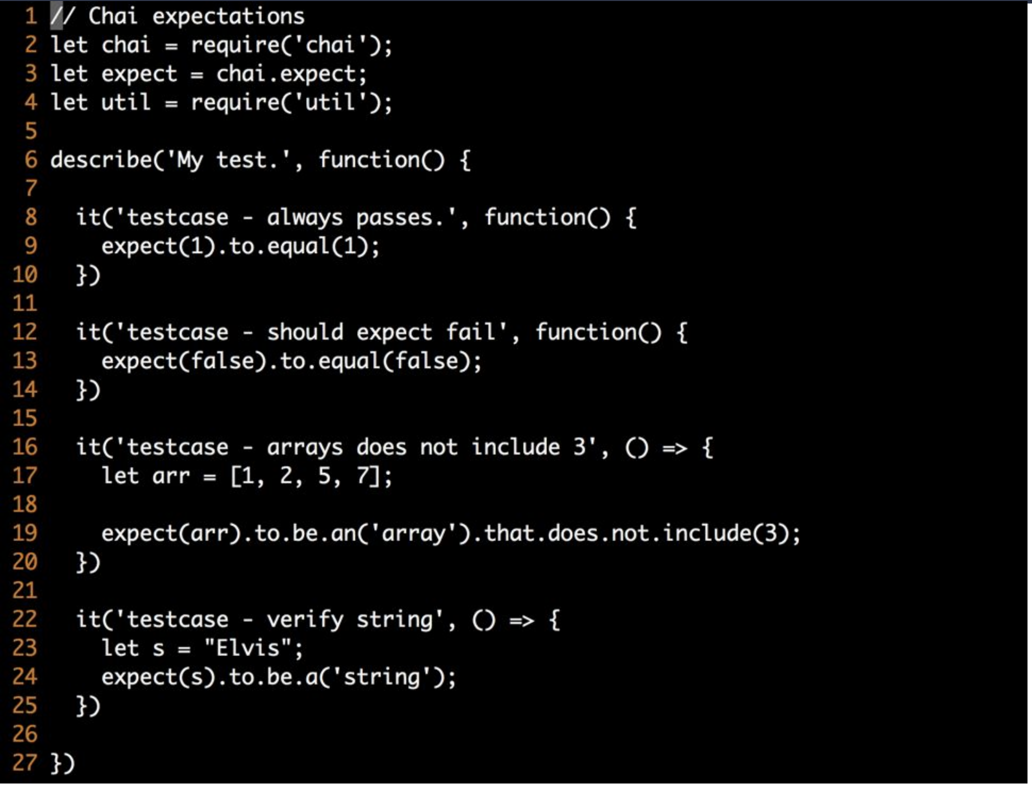
Causes all `.equal`, `.include`, `.members`, `.keys`, and `.property` assertions that follow in the chain to use deep equality instead of strict (`===`) equality. See the `deep-eql` project page for info on the deep equality algorithm: https://github.com/chaijs/deep-eql.

```
// Target object deeply (but not strictly) equals `{a: 1}`
expect({a: 1}).to.deep.equal({a: 1});
expect({a: 1}).to.not.equal({a: 1});

// Target array deeply (but not strictly) includes `{a: 1}`
expect([{a: 1}]).to.deep.include({a: 1});
expect([{a: 1}]).to.not.include({a: 1});

// Target object deeply (but not strictly) includes `x: {a: 1}`
```

Test Framework::Assertion Library with Chai

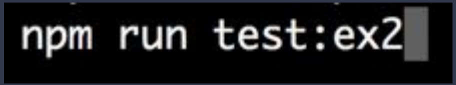


```
npm run test:ex1:chai
```

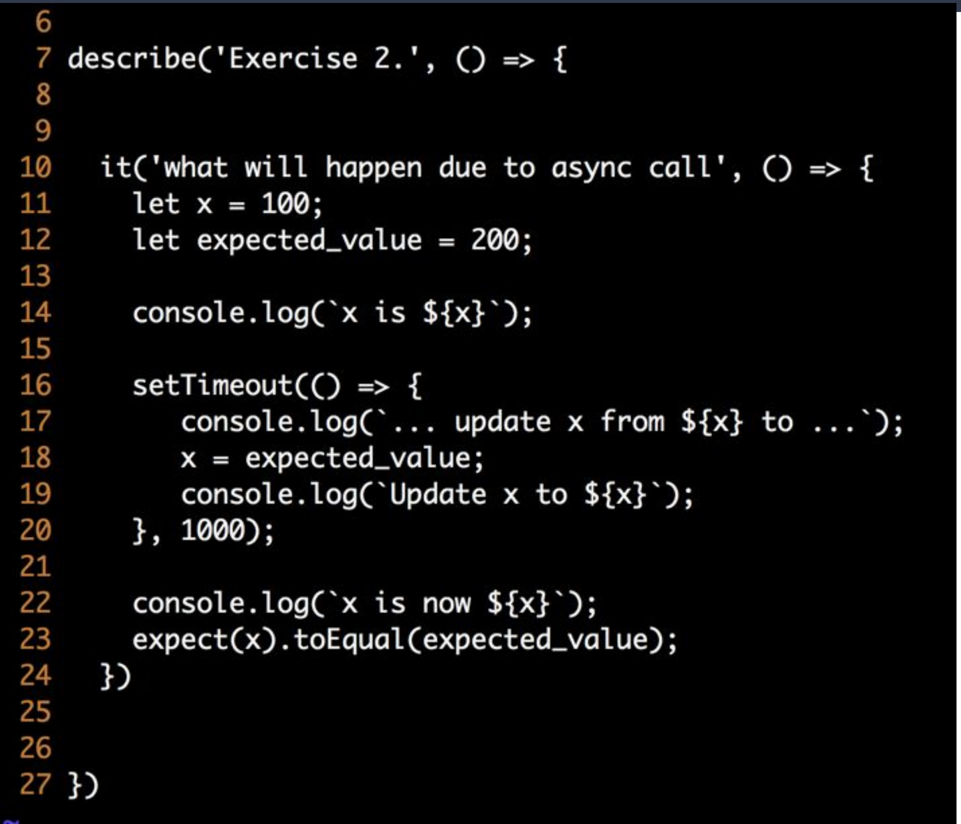


```
// Chai expectations
let chai = require('chai');
let expect = chai.expect;
let util = require('util');

describe('My test.', function() {

  it('testcase - always passes.', function() {
    expect(1).to.equal(1);
  })

  it('testcase - should expect fail', function() {
    expect(false).to.equal(false);
  })

  it('testcase - arrays does not include 3', () => {
    let arr = [1, 2, 5, 7];

    expect(arr).to.be.an('array').that.does.not.include(3);
  })

  it('testcase - verify string', () => {
    let s = "Elvis";
    expect(s).to.be.a('string');
  })

})
```

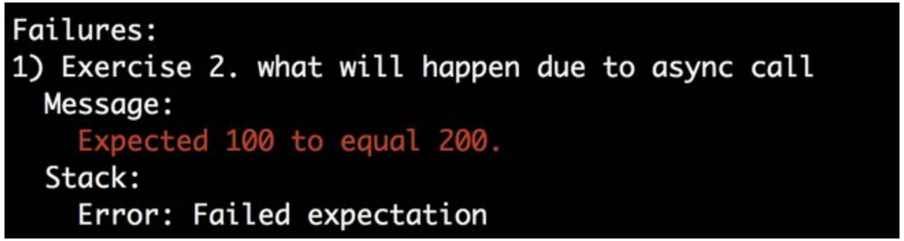
Test Framework::Assertion::Async Issues



```
npm run test:ex2
```

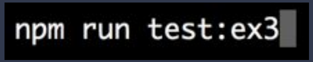


```

describe('Exercise 2.', () => {


  it('what will happen due to async call', () => {
    let x = 100;
    let expected_value = 200;

    console.log(`x is ${x}`);

    setTimeout(() => {
      console.log(`... update x from ${x} to ...`);
      x = expected_value;
      console.log(`Update x to ${x}`);
    }, 1000);

    console.log(`x is now ${x}`);
    expect(x).toEqual(expected_value);
  })


})
```

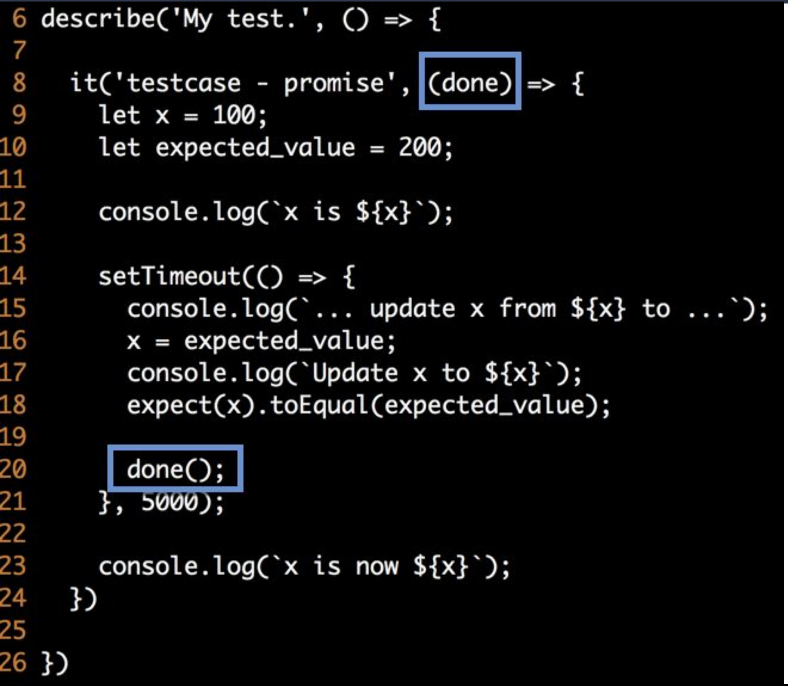


```
Failures:
1) Exercise 2. what will happen due to async call
  Message:
    Expected 100 to equal 200.
  Stack:
    Error: Failed expectation
```

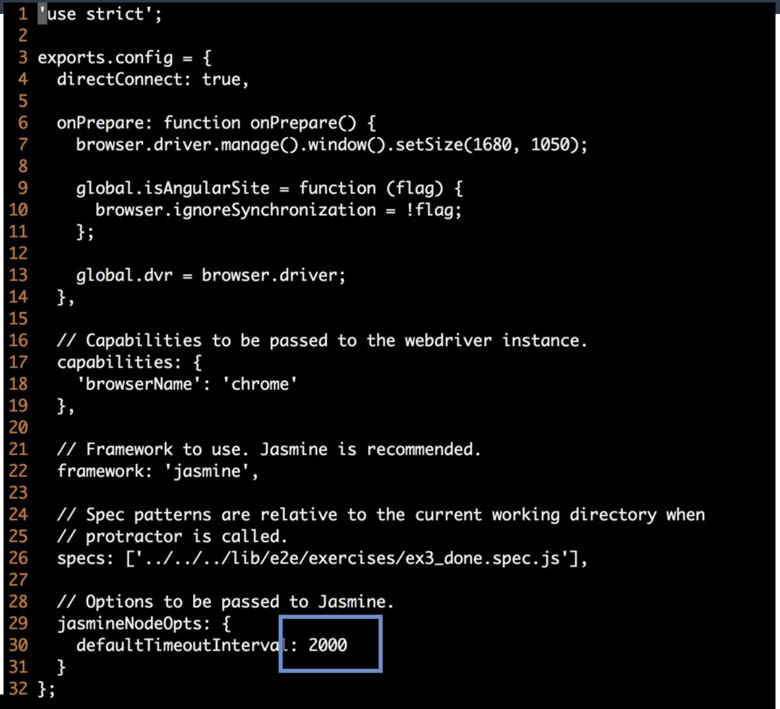
Test Framework::Assertion::Async & Timeouts



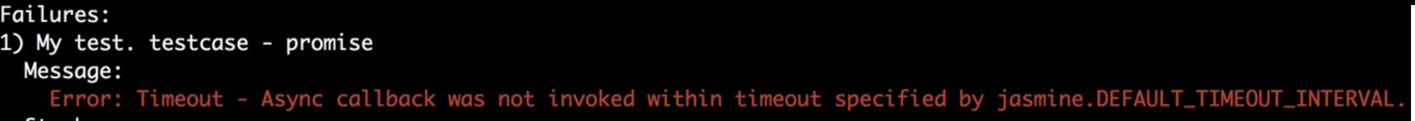
```
npm run test:ex3
```



```
describe('My test.', () => {

  it('testcase - promise', (done) => {
    let x = 100;
    let expected_value = 200;

    console.log(`x is ${x}`);

    setTimeout(() => {
      console.log(`... update x from ${x} to ...`);
      x = expected_value;
      console.log(`Update x to ${x}`);
      expect(x).toEqual(expected_value);

      done();
    }, 5000);

    console.log(`x is now ${x}`);
  })

})
```

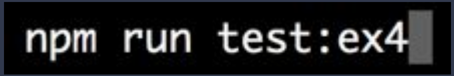


```
'use strict';

exports.config = {
  directConnect: true,

  onPrepare: function onPrepare() {
    browser.driver.manage().window().setSize(1680, 1050);

    global.isAngularSite = function (flag) {
      browser.ignoreSynchronization = !flag;
    };

    global.dvr = browser.driver;
  },

  // Capabilities to be passed to the webdriver instance.
  capabilities: {
    'browserName': 'chrome'
  },

  // Framework to use. Jasmine is recommended.
  framework: 'jasmine',

  // Spec patterns are relative to the current working directory when
  // protractor is called.
  specs: ['../../../lib/e2e/exercises/ex3_done.spec.js'],

  // Options to be passed to Jasmine.
  jasmineNodeOpts: {
    defaultTimeoutInterval: 2000
  }
};
```

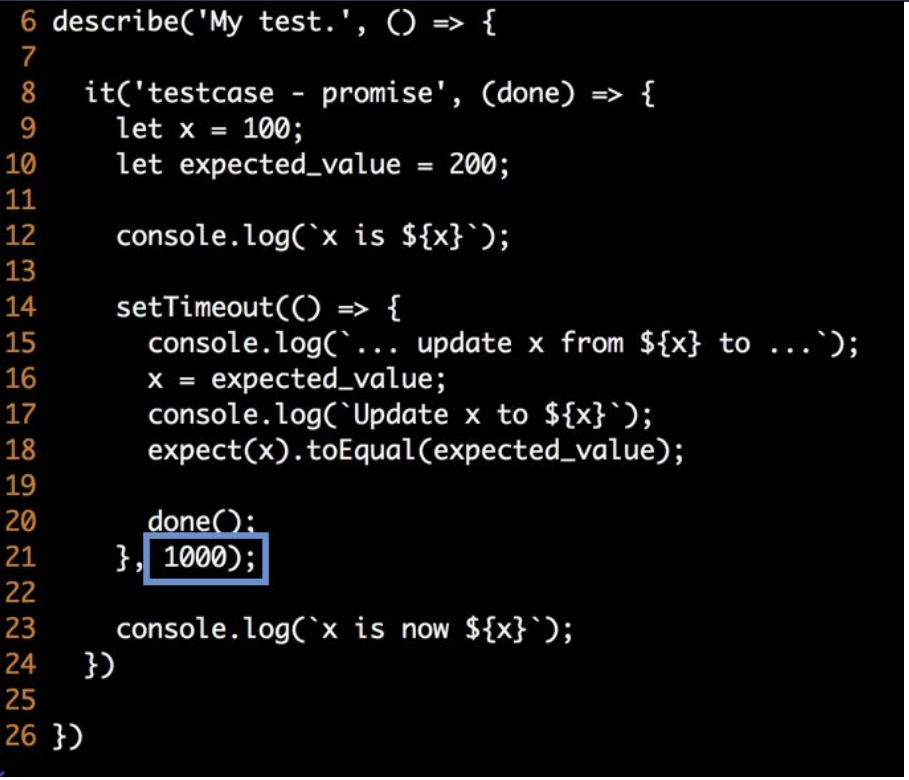


```
Failures:
1) My test. testcase - promise
  Message:
    Error: Timeout - Async callback was not invoked within timeout specified by jasmine.DEFAULT_TIMEOUT_INTERVAL.
```

Test Framework::Assertion::Async & Timeouts

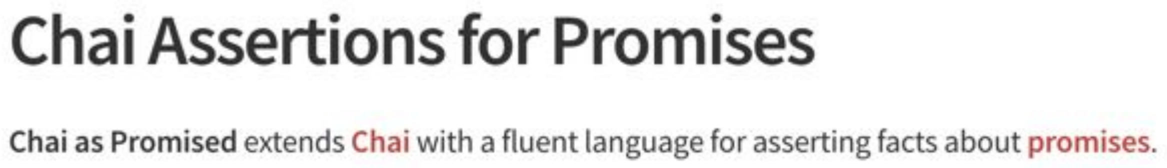


```
npm run test:ex4
```



```
describe('My test.', () => {

  it('testcase - promise', (done) => {
    let x = 100;
    let expected_value = 200;

    console.log(`x is ${x}`);

    setTimeout(() => {
      console.log(`... update x from ${x} to ...`);
      x = expected_value;
      console.log(`Update x to ${x}`);
      expect(x).toEqual(expected_value);

      done();
    }, 1000);

    console.log(`x is now ${x}`);
  })

})
```

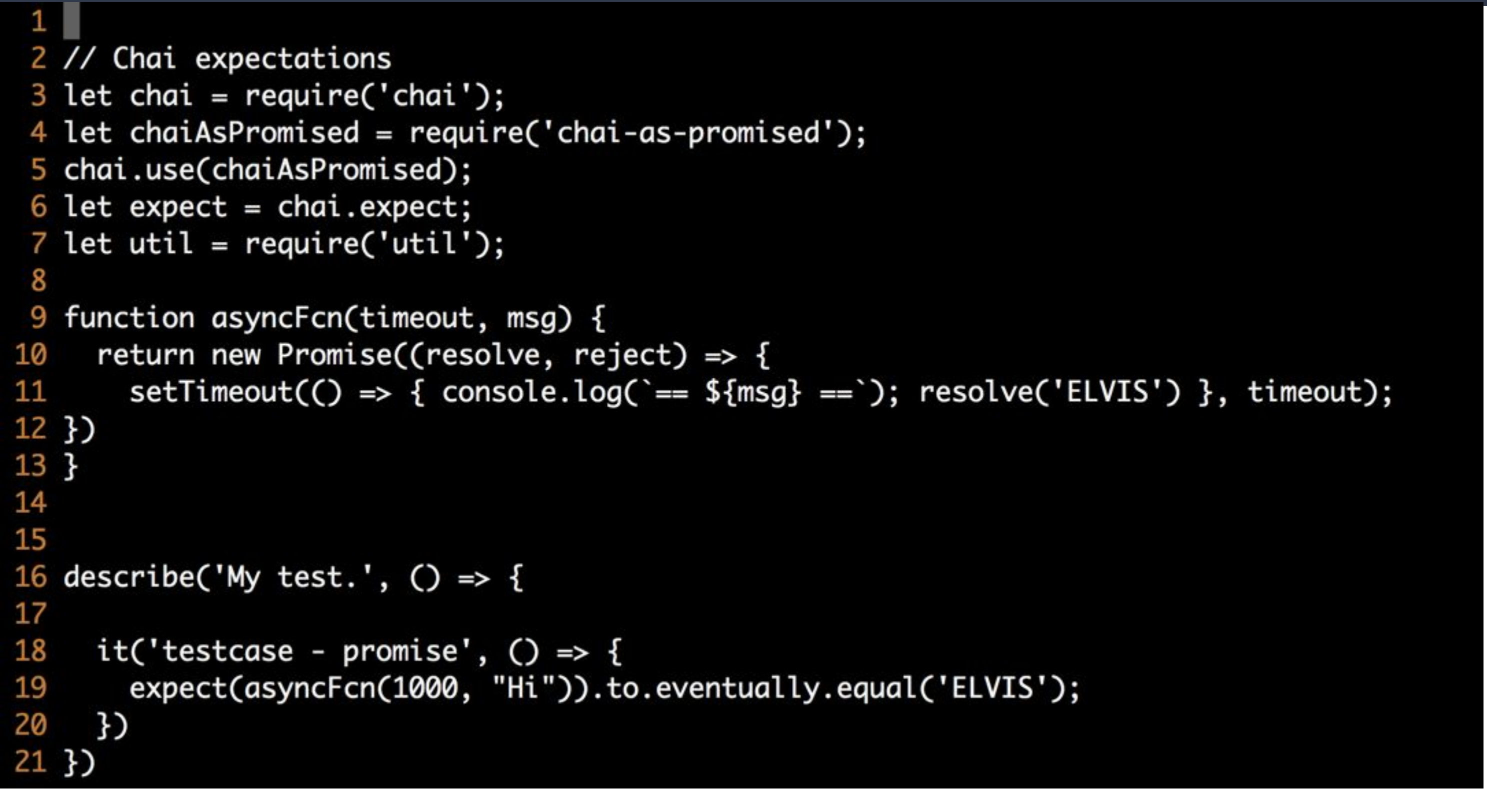
Test Frameworks::Assertion Library with chai-as-promised



Chai Assertions for Promises

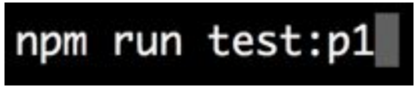
Chai as Promised extends Chai with a fluent language for asserting facts about promises.

Test Frameworks::Assertion Library with chai-as-promised - `npm run test:ex5`



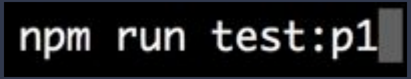
```

// Chai expectations
let chai = require('chai');
let chaiAsPromised = require('chai-as-promised');
chai.use(chaiAsPromised);
let expect = chai.expect;
let util = require('util');

function asyncFcn(timeout, msg) {
  return new Promise((resolve, reject) => {
    setTimeout(() => { console.log(`== ${msg} ==`); resolve('ELVIS') }, timeout);
})
}


describe('My test.', () => {

  it('testcase - promise', () => {
    expect(asyncFcn(1000, "Hi")).to.eventually.equal('ELVIS');
  })
})
```

Protractor::Exercise 1::Open a browser

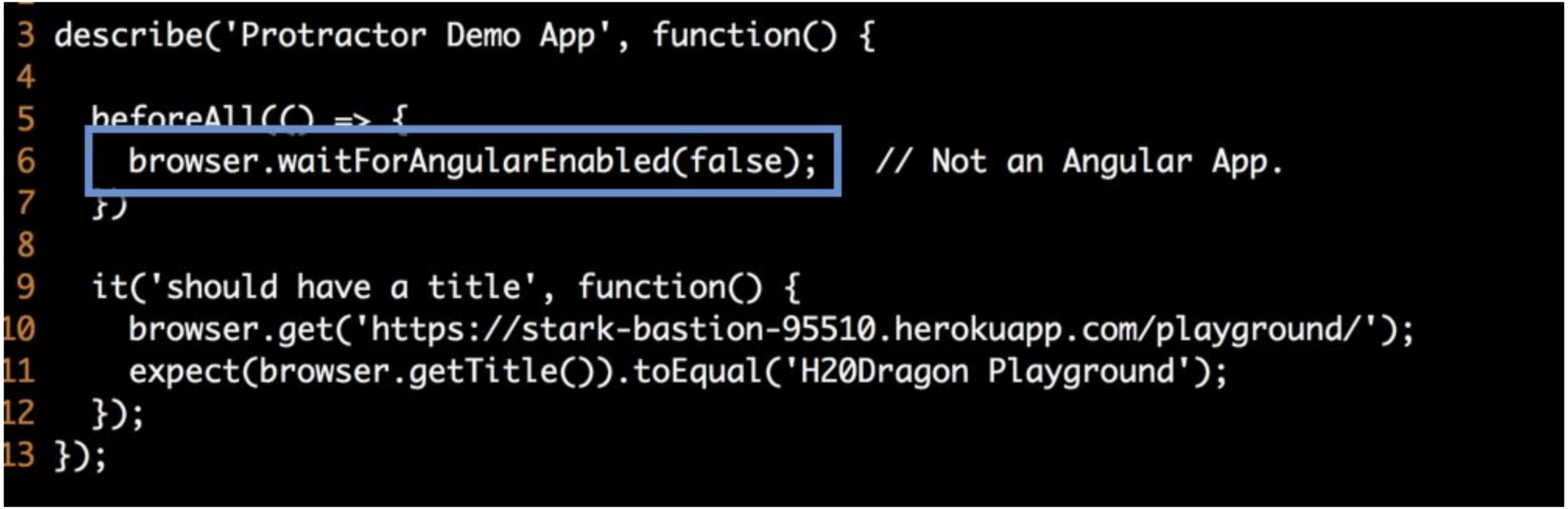


```
npm run test:p1
```

Protractor::Exercise2::Navigate

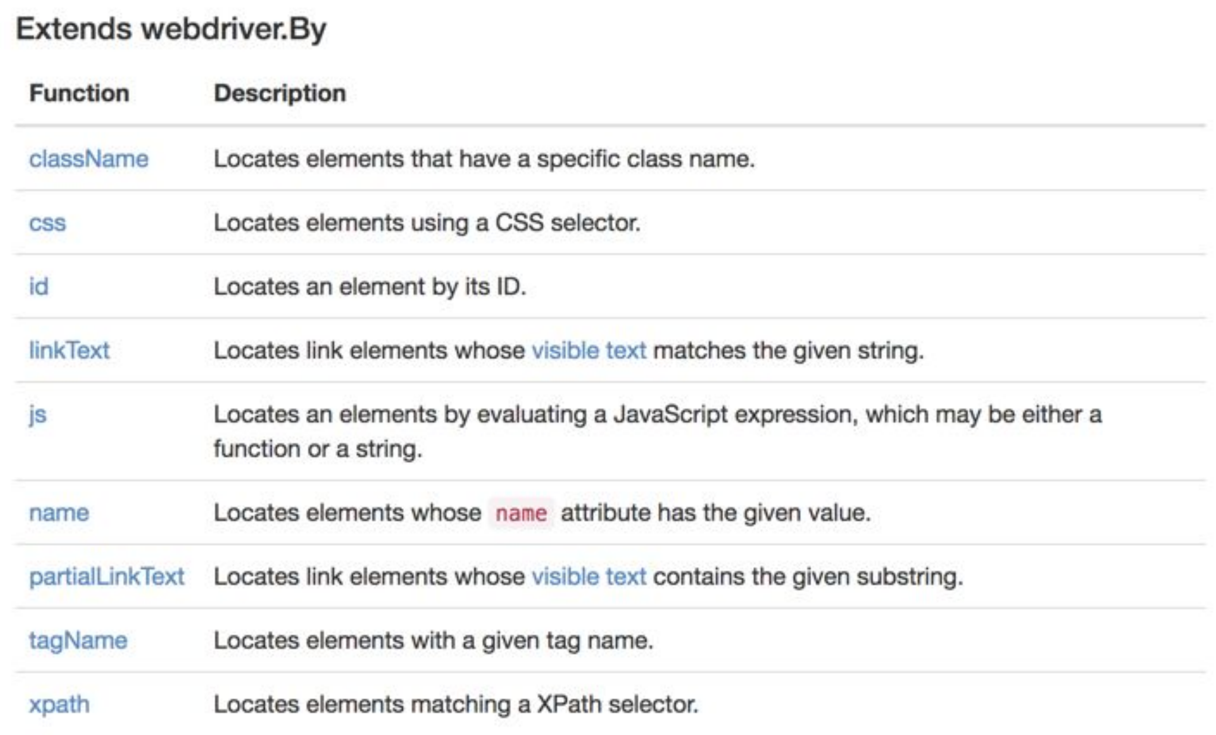


```
npm run test:p1
```



```
describe('Protractor Demo App', function() {

  beforeAll(() => {
    browser.waitForAngularEnabled(false);   // Not an Angular App.
  })

  it('should have a title', function() {
    browser.get('https://stark-bastion-95510.herokuapp.com/playground/');
    expect(browser.getTitle()).toEqual('H20Dragon Playground');
  });
});
```

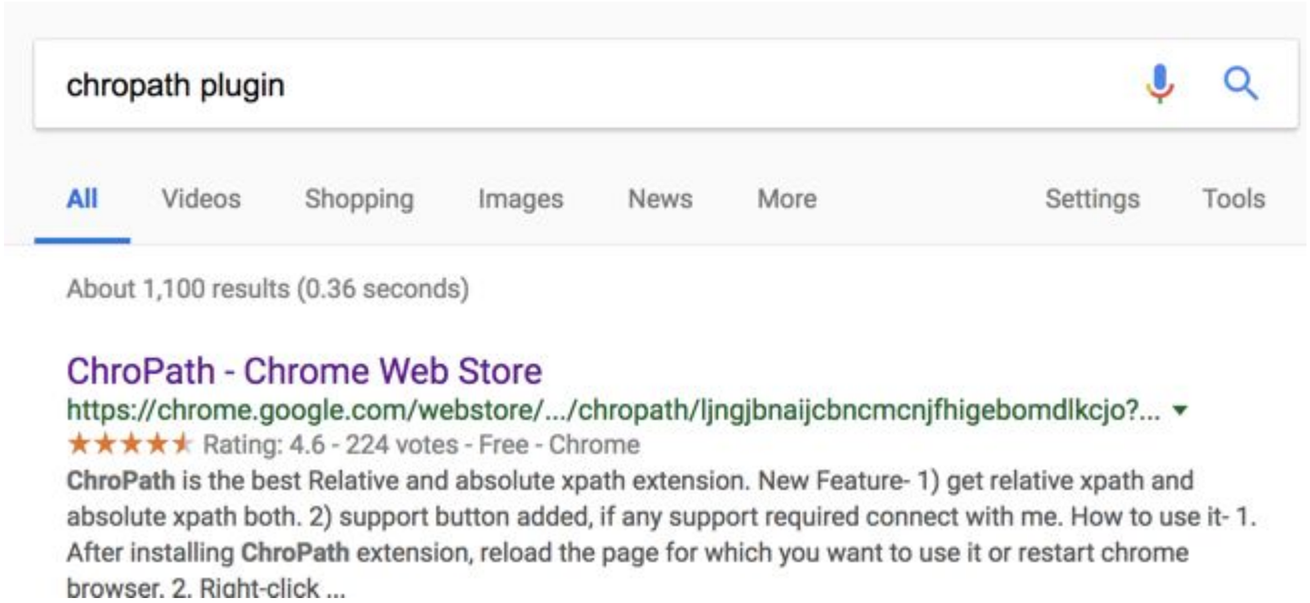
Protractor::Exercise3::Find Elements Selenium Locators



Extends webdriver.By

Function	Description
className	Locates elements that have a specific class name.
css	Locates elements using a CSS selector.
id	Locates an element by its ID.
linkText	Locates link elements whose visible text matches the given string.
js	Locates an elements by evaluating a JavaScript expression, which may be either a function or a string.
name	Locates elements whose `name` attribute has the given value.
partialLinkText	Locates link elements whose visible text contains the given substring.
tagName	Locates elements with a given tag name.
xpath	Locates elements matching a XPath selector.

Selenium Locators::Demo
Use a Chrome Plugin ‘chropath’

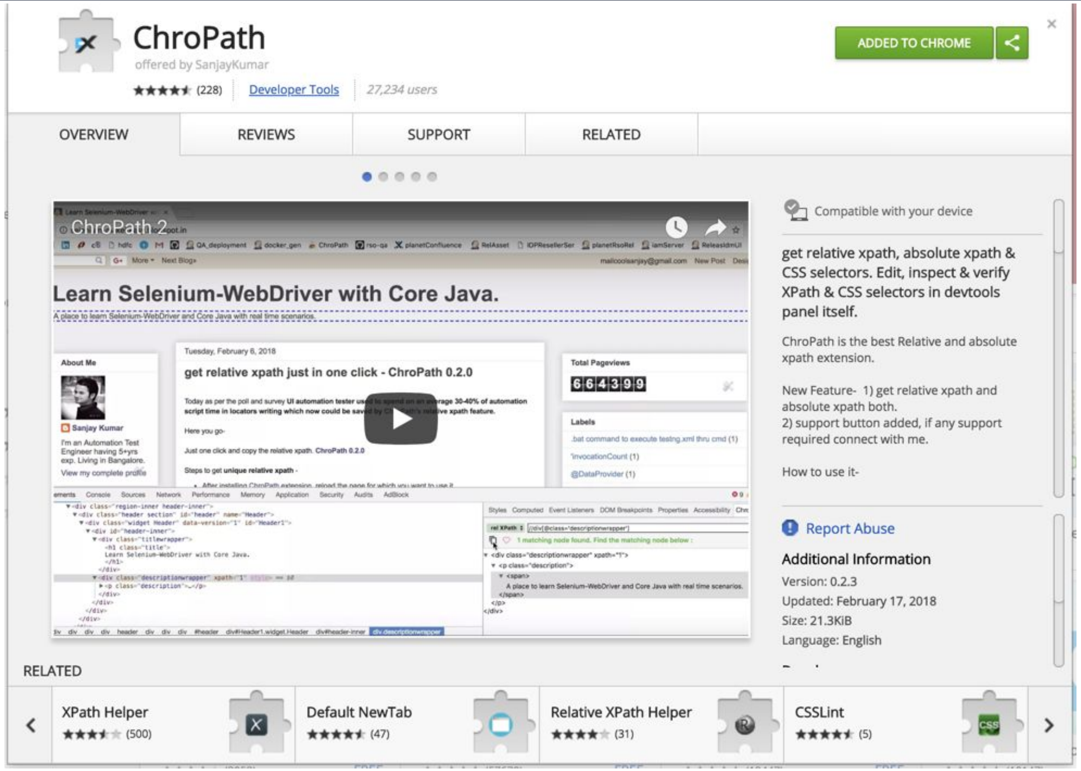


chropath plugin

About 1,100 results (0.36 seconds)

ChroPath - Chrome Web Store
https://chrome.google.com/webstore/.../chropath/ljngjbnaijcbncmcnjfhigebomdlkcjo?...
★★★★★ Rating: 4.6 - 224 votes - Free - Chrome
ChroPath is the best Relative and absolute xpath extension. New Feature- 1) get relative xpath and absolute xpath both. 2) support button added, if any support required connect with me. How to use it- 1. After installing **ChroPath** extension, reload the page for which you want to use it or restart chrome browser. 2. Right-click ...

Selenium Locators::Demo
Use a Chrome Plugin ‘chropath’



ChroPath
offered by SanjayKumar
(228) Developer Tools 27,234 users

ADDED TO CHROME

OVERVIEW REVIEWS SUPPORT RELATED

Compatible with your device

get relative xpath, absolute xpath & CSS selectors. Edit, inspect & verify XPath & CSS selectors in devtools panel itself.

ChroPath is the best Relative and absolute xpath extension.

New Feature- 1) get relative xpath and absolute xpath both.
2) support button added, if any support required connect with me.

How to use it-

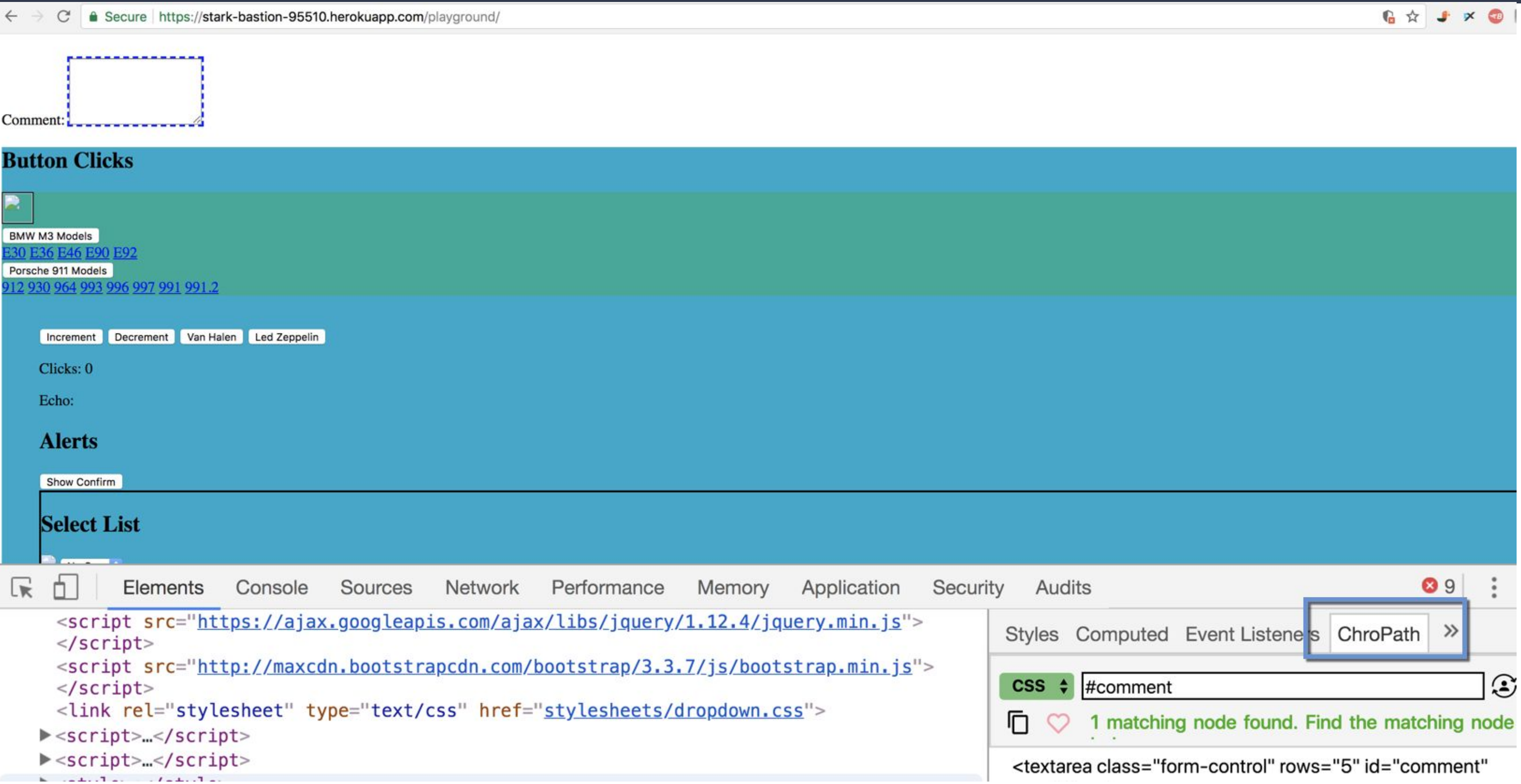
Report Abuse

Additional Information
Version: 0.2.3
Updated: February 17, 2018
Size: 21.3KiB
Language: English

RELATED

XPath Helper (500)
Default NewTab (47)
Relative XPath Helper (31)
CSSLint (5)

Selenium Locators::Demo
Use a Chrome Plugin ‘chropath’



Secure https://stark-bastion-95510.herokuapp.com/playground/

Comment:

Button Clicks

BMW M3 Models

E30 E36 E46 E90 E92

Porsche 911 Models

912 930 964 993 996 997 991 991.2

Increment Decrement Van Halen Led Zeppelin

Clicks: 0

Echo:

Alerts

Show Confirm

Select List

Elements Console Sources Network Performance Memory Application Security Audits

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
</script>
<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
</script>
<link rel="stylesheet" type="text/css" href="stylesheets/dropdown.css">
▶<script>…</script>
▶<script>…</script>
```

Styles Computed Event Listeners ChroPath

CSS #comment

1 matching node found. Find the matching node

```
<textarea class="form-control" rows="5" id="comment"
```

Protractor::Exercise::Find Elements

```
npm run test:p2
```

STP SOFTWARE TEST PROFESSIONALS CONFERENCE SPRING 2018

Peter Kim
LinkedIn: peterkim777
Twitter: peter_kim777

Jennifer Peeling