

## BETTER CONSOLE REPORT

1. Install jasmine-spec-reporter through the command:
  - a. **npm install jasmine-spec-reporter --save-dev**
2. In the Protractor configuration file, import the package and configure the customizable options:

```
let SpecReporter = require('jasmine-spec-reporter').SpecReporter;
exports.config = {
  framework: 'jasmine',

  directConnect: 'true',
  specs: ['zzzElnar.js'],

  onPrepare: function() {
    browser.manage().window().maximize();
    jasmine.getEnv().addReporter(new SpecReporter({
      displayFailuresSummary: true,
      displayFailedSpec: true,
      displaySuiteNumber: true,
      displaySpecDuration: true
    }));
  }
};
```

## ALERTS, POPUPS, MULTIPLE WINDOWS and IFRAMES

Alert is a small message box which displays on-screen notification to give the user some kind of information or ask for permission to perform certain kind of operation. It may be also used for warning purpose. We cannot identify alerts using inspect tools

- We cannot just locate alerts.
- It is not a Window.
- It will not allow to perform any other operation on WebPage unless it is taken care of.
- We will need to switch to alert window via: `browser.switchTo().alert()`

### Alert Methods:

```
browser.switchTo().alert().getText();
browser.switchTo().alert().accept();
browser.switchTo().alert().dismiss();
browser.switchTo().alert().sendKeys();
```

the-internet.herokuapp.com says  
I am a JS Alert

OK

<button onclick="jsAlert()">Click for JS Alert</button> == \$0

</li>

<script></script>

body div #content div.example ul li button

ript Alerts

Computed Event Listeners DOM Breakpoints >>

.style {

:hover, button:focus, .button:hover,

!focus {

Click for JS Alert

examples of different JavaScript alert which can be troublesome for automation

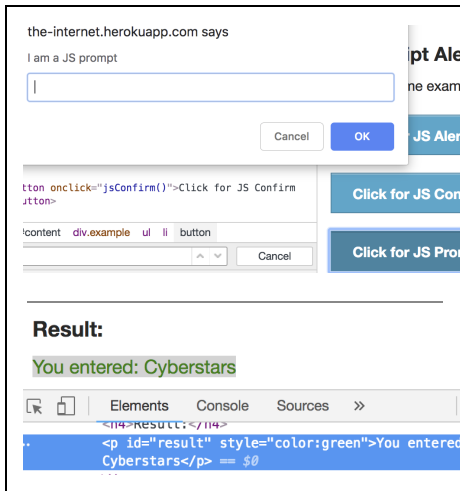
### Result:

You successfully clicked an alert

Elements Console Sources Netw

<h4>Result:</h4>  
<p id="result" style="color:green">  
You successfully clicked an alert</p>

```
describe('Testing Alerts', () => {  
  beforeAll(() => {  
    browser.waitForAngularEnabled(false);  
    browser.get('http://the-internet.herokuapp.com/');  
    element(by.linkText('JavaScript Alerts')).click();  
  })  
  
  it('should get text of alerts', () => {  
    //which can be troublesome for automation  
    element(by.buttonText('Click for JS Alert')).click();  
    browser.sleep(2000);  
    var myAlert = browser.switchTo().alert();  
    expect(myAlert.getText()).toEqual('I am a JS Alert');  
  });  
  
  it('should click OK', () => {  
    var myAlert = browser.switchTo().alert();  
    myAlert.accept(); //click OK  
    expect($('#result').getText()).toEqual('You successfully clicked an alert');  
  })  
  
  it('should dismiss the alert', () => {  
    element(by.buttonText('Click for JS Confirm')).click();  
    browser.sleep(2000);  
    var myAlert = browser.switchTo().alert();  
    expect(myAlert.getText()).toEqual('I am a JS Confirm');  
    browser.sleep(2000);  
    myAlert.dismiss(); //click cancel  
    expect($('#result').getText()).toEqual('You clicked: Cancel');  
  });  
});
```



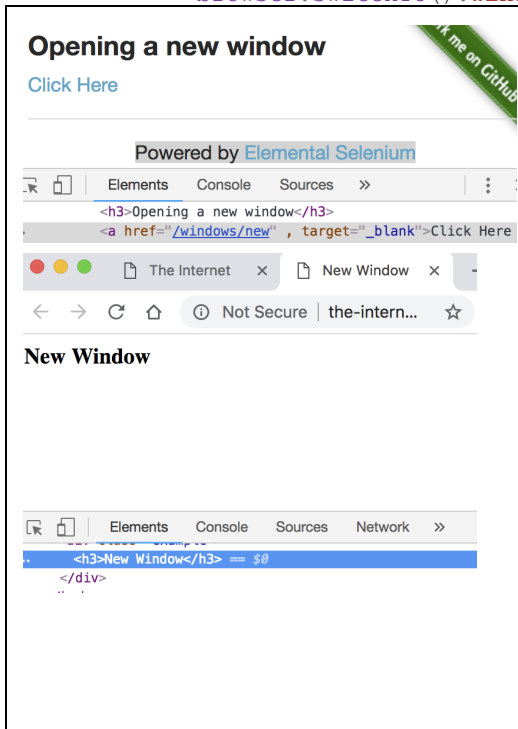
```

it('should send some text to the alert', () => {
    element(by.buttonText('Click for JS Prompt')).click();
    browser.sleep(2000);
    var myAlert = browser.switchTo().alert();
    expect(myAlert.getText()).toEqual('I am a JS prompt');
    browser.sleep(2000);
    var text = 'Cyberstars';
    myAlert.sendKeys(text);
    myAlert.accept();
    expect($('#result').getText()).toEqual('You entered: ' + text);
});

```

## MULTIPLE WINDOWS & POPUPS

- There are some cases where you will have more than window/tab open
- When application opens up a new pop-up window or tab, Selenium/protractor will not automatically pass control to that page. Just because it is shown on the screen doesn't mean protractor has control on it by default.
- We have to pass control to the new tab/window perform our desired actions and then pass control back.
- Tabs and windows are treated as same in protractor. They both have unique GU IDs.
- To get the current active) window/tab GU ID:
  - `browser.getWindowHandle()`;
- To get all windows/tabs GU IDs:
  - `browser.getWindowHandles()`;
  - Once you get the the window handles, it returns you art array as seen. we can loop through this array and do other operations as well.
- To switch control over to the target window/tab GU ID:
  - `browser.switchTo().window(guid)`;



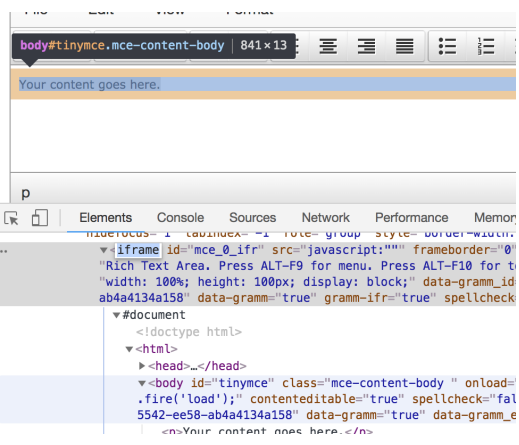

```

describe('Testing Alerts', () => {
    var browserHandles = [];
    beforeAll(() => {
        browser.waitForAngularEnabled(false);
        browser.get('http://the-internet.herokuapp.com/');
        element(by.linkText('Multiple Windows')).click();
        element(by.linkText('Click Here')).click();
    });
    it('should switch to new window', () => {
        browser.getAllWindowHandles().then(function(handles) {
            browserHandles = handles;
            browser.switchTo().window(browserHandles[1]).then(function() {
                browser.sleep(3000);
                expect(browser.getCurrentUrl()).toContain('windows/new')
            });
        });
    });
    it('should go back to previous window', () => {
        browser.close().then(function() {
            browser.switchTo().window(browserHandles[0]);
            browser.sleep(2000);
            expect(element(by.linkText('Click Here')).isDisplayed()).toBe(true);
        });
    });
});

```

## FRAME & iFRAME

- iFrame is nothing but another webelement in html page, which displays another part of webpage.
- You **won't be able to interact with it via the DOM**. We have to **switch into frame** to see the elements present in the frame. Protractor throws **NoSuchElementException** unless we switch to the iframe
- We can handle iframes present in the webpage using **browser.switchTo()** command in protractor.
- There are couple of ways to switch to frame/iframe.
  - Using locator:  
**browser.switchTo().frame(browser.driver.findElement(by.tagName('iframe')));**
  - If using element array finder, then make sure to use **.getWebElement()** to convert.  
**browser.switchTo().frame(element(by.tagName('iframe')).getWebElement());**
  - Using Index: (not preferred as order can change)  
// switch to 1st frame  
**browser.switchTo().frame(1);**
  - If nested iFrames, then to go back to outer iframe, use:  
**browser.switchToParentFrame();**
  - To switch back to default page:  
**browser.switchTo().defaultContent();**

	<pre>describe('Practising Iframe', ()=&gt;{    beforeAll(()=&gt;{     browser.waitForAngularEnabled(false);     browser.get('https://the-internet.herokuapp.com/iframe');   });    it('should switch to iFrame', ()=&gt;{     browser.sleep(3000);     browser.switchTo()       .frame(browser.driver.findElement(by.tagName('iframe')));     \$('#tinymce').click();     \$('#tinymce').clear();     \$('#tinymce').sendKeys('Cyberstars')     browser.sleep(6000);   }); });</pre>
	<pre>describe('Protractor Frames Steps',function() {   beforeAll(()=&gt;{     browser.waitForAngularEnabled(false);     browser.get("http://qaclickacademy.com/practice.php");   })   it('should open frame',function() {     browser.switchTo().frame("courses-iframe");     expect(\$\$('.tools .container-fluid span').get(0).getText()).toEqual('(+) 323-744-6780')     expect(\$\$('.tools .container-fluid span').get(1).getText()).toEqual('info@qaclickacademy.com')     browser.switchTo().defaultContent();     element(by.css('.logoClass')).click();     browser.sleep(3000);   }); });</pre>

## WAITS

- Why do we need waiting?
  - Because world is not perfect.
  - You will come across scenarios where an element you are trying to interact is not loaded to dom, or not clickable at that moment etc.
- Most basic version of wait is: **browser.sleep(5000);**
  - Avoid using this in your test scripts, as it leads to **unnecessary longer test execution time**.
  - Forces browser to wait for the given amount of time, even if desired element shows up early.
  - Can be used while **troubleshooting** your script.
- When testing Angular pages, Protractor automatically waits for angular to be ready and then it executes the next step in the control flow. So, you don't need to wait.
- However, there are cases that you will need to use waits. Sometimes the tests outrun browser!
- You will face situations that we will need to use waits both on:
  - For some elements on **Angular pages**. (Not as common)
  - For some elements on **Non-angular pages**. (Very common)

### 1. IMPLICIT WAIT

- Implicitly wait in protractor, sets maximum time that we are going to wait for the element to be available in the Website.
- If we have given implicit wait of 30,000ms then, Implicit wait tries to find the element in first go, if element is not present, implicit wait tries to find the element after every 500ms, and it goes on till the time reaches the 30,000 milliseconds limit we set.
- Add following to the on Prepare option in conf.js file.
  - `browser.manage().timeouts().implicitlyWait(30000);`
- If **element is found** before implicitly wait time, **protractor moves to next commands** in the program **without waiting** to complete the implicitly wait time, this wait is also called dynamic wait.
- Implicitly wait is one of the way to request selenium not throw any exception until provided time.
  - **NO\_SUCH\_ELEMENT Error**
- Default wait time of the protractor is **500ms**. This gets overridden when added.
- Implicit wait is **set for the entire duration of your webdriver** and is set at the start of your program. If Mostly gets put in the **conf.js file**.
- **Issues:**
  - You are hardcoding same maximum wait time for all elements. If Element is taking longer than the set timeout, it will throw error.
  - One size fits all —> approach doesn't work.

CONFIG FILE (we add)

```
onPrepare: function() {  
  browser.manage().timeouts().implicitlyWait(30000);  
  ...  
}
```

```
describe('Testing Implicit Wait', () => {  
  beforeAll(() => {  
    browser.waitForAngularEnabled(false);  
    browser.get('http://www.target.com/');  
  })  
  it('should wait for a fixed amount of time when  
  element not found', () => {  
    element(by.linkText('Categories2')).click();  
  });  
});
```

Testing Implicit Wait

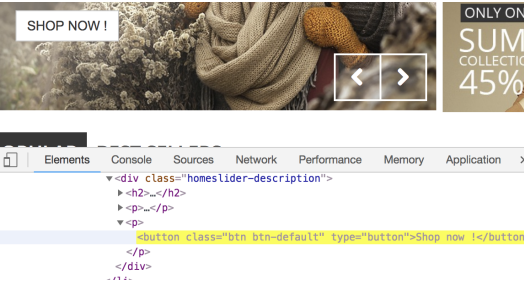
x should wait for a fixed amount of time when element not found

- Error: Timeout - Async callback was not invoked within timeout specified by jasmine.DEFAULT\_TIMEOUT\_INTERVAL.
- Failed: No element found using locator: By(link text, Categories2)

## 2. EXPLICIT WAY (We prefer)

- What if we want to make sure element is in certain condition (displayed, checked, clickable etc) before protractor starts interacting with the element. This is when Explicit wait comes into picture.
- The explicit wait tells the protractor to **wait for certain conditions** or the **maximum time limit** before throwing an exception=error.
- Explicit Wait is a dynamic wait, which means it **will wait only if the condition is not met**.
- Explicit wait's **scope** is only the element it is sued for. (unlike whole script like in implicit wait)
- For Angular Page and Non-Angular waits, we use Expected Conditions.
- **Expected Conditions class** have several methods.

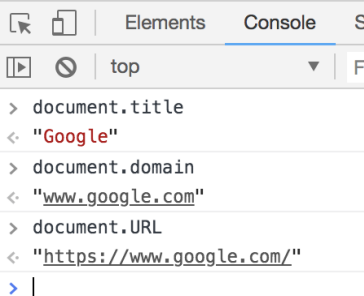
- `var EC = protractor.ExpectedConditions;`
  - `browser.wait(EC.presenceOf(element(by.css('#GatewayTools'))),12000);`
  - `browser.wait(EC.visibilityOf(element(by.css('#GatewayTools'))),12000);`
  - `browser.wait(EC.invisibilityOf(element(by.css('#GatewayTools'))),12000);`
  - `browser.wait(EC.elementToBeClickable(element(by.css('#GatewayTools'))),12000);`
  - `browser.wait(EC.textToBePresentInElement($('#abc'),'foo'),12000);`
  - `browser.wait(EC.alertIsPresent(),12000);`
  - `browser.wait(EC.elementToBeSelected($('#abc'),'foo'),12000);`



```
describe('Working with Explicit Waits', () => {
  var EC = protractor.ExpectedConditions;
  it('should wait for an element to be visible', () => {
    browser.waitForAngularEnabled(false);
    browser.get('http://automationpractice.com/index.php');
    browser.wait(EC.visibilityOf($('#homeslider-description .btn')),
      get(2),14000).then(function() {
        $('#homeslider-description .btn').get(2).click();
      });
  });
});
```

## JAVASCRIPT EXECUTOR

- JavaScriptExecutor is an Interface that helps to execute JavaScript through Selenium Webdriver.
  - (Irrespective of the Selenium language binding (Java, C#, Python etc.) you are using.)
- We should go for laytaLtor only when we are not able to perform a particular task with our protractor:
  - Like sometimes, we may not be able click an element
- We have different javascript executor script method in protractor :
  - `browser.executeScript("javascript command", "arguments");`
  - `browser.executeScriptWithDescription("javascript command", "description message", "arguments");`
- Methods to open a page, get title, url and domain of the page.
  - `browser.executeScript("window.location='http://automationpractice.com.index.php';");`
  - `browser.executeScript("return document.title");`
  - `browser.executeScript("return document.URL");`
  - `browser.executeScript("return document.domain");`



```
describe('Practising with JS Executor',function() {
  beforeEach(()=>{
    browser.waitForAngularEnabled(false);
    browser.executeScript("window.location='https://www.google.com/';");
  });
  it('should get title/url/domain of the page ',function(){
    browser.executeScript("return document.title").then(text=>{
      console.log(text);
    });
    browser.executeScript("return document.URL").then(url=>{
      console.log('url is: ' +url)
    });
    browser.executeScript("return document.domain").then(domain=>{
      console.log('domain is: ' +domain)
    });
  });
});
```

## JS EXECUTOR METHODS

- Methods to click:
  - `browser.executeScript("document.getElementsByName('submit_search')[0].click()");`
  - `browser.executeScript('arguments[0].click();', element);`
- sendKeys:
  - `browser.executeScript("document.getElementById('search_query_top').value='leather jacket'");`
- Scroll down:
  - `browser.executeScript("window.scrollTo(0,600)");`
  - `browser.executeScript("document.getElementById('defaults').scrollIntoView(true)");`
- Why is it not recommended as first choice ?
  - The **Protractor works similar way to an user**, protractor will perform the operation only if a user(physical person) can perform the operation on element.
  - We can rely on protractor, if protractor says that it cannot interact with an element, then is **physical person also cannot interact with the element**.
  - So, It is **better to fail the test case when protractor methods fails** instead of trying with javascript Executor. That is why manual check here is important when writing test case.



```
describe('Practising with JS Executor',function() {
  beforeEach(()=>{
    browser.waitForAngularEnabled(false);
    browser.executeScript("window.location='https://www.google.com/'");
  });
  it('should type',()=>{
    browser.executeScript("document.getElementsByName('q')[0].value='kindle'");
    browser.sleep(2000);
    browser.actions().sendKeys(protractor.Key.ENTER).perform();
    browser.sleep(2000);
    browser.executeScript("window.scrollTo(0,600)");
    browser.sleep(3000);
  });
});
```

## FILE UPLOAD

- File Upload:
    - `element(By.id('file-upload')).sendKeys('C:/Users/emirzayev/Desktop/CTEK/README.md');`
  - There is no need to simulate the clicking of the "Browse" button. WebDriver automatically enters the file path onto the file-selection text box of the `<input type="file">` element
- ```
function uploadFile (fileName) {
  // Append folder location of your current directory to your file to get full path.
  var absolutePath = path.resolve(__dirname, fileName);
  // Send file location for upload
  $$('input[type="file"]').get(0).sendKeys(absolutePath); };
```
- File Upload (When element not visible)
    - If you receive error that element is not visible.
    - You need to make the input visible using javascript executor. (You can add this to the above method.)

```
// Unhide file input
browser.executeScript("arguments[0].style.visibility='visible'; ", fileElem.getWebElement());
$$('input[type="file"]').get(0).sendKeys(absolutePath);
Return; };
```

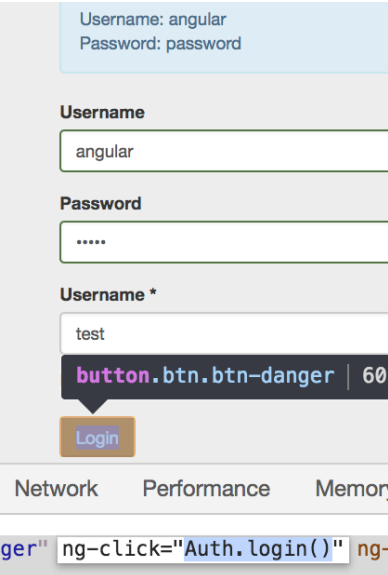


```
describe('Practising with JS Executor',function() {
  //when you see `require` --> means `importing`
  var path = require('path');
  it('should upload file',()=>{
    browser.get('http://nervgh.github.io/pages/angular-file-upload/examples/simple');

    //__dirname = /Users/mac/Desktop/CtekProject/DEMO/
    $$('input[type="file"]').get(0).sendKeys(__dirname + '/Docs/test.txt');
    $('btn-success').click();
  });
});
```

## BY.ADDLOCATORS

- Add a locator to this instance of ProtractorBy. You can use **any attribute in the DOM** to create Custom Locators.
- This locator **can then be used** with `element(by.locatorCustomName(args))`.
- Practice: <http://www.way2automation.com/angularjs-protractor/registration/#/login>
- Below method is created to locate any element with different ng-click values.

|                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <pre>describe('Testing Adding Locators', ()=&gt;{   by.addLocator('ngClick', function(toState, parentelement) {     //Below line tells to look for the element in the parent element if not in     then the entire document.     var using = parentelement    document ;     var prefixes = ['ng-click'];     for (var p = 0; p &lt; prefixes.length; ++p) {       var selector = '*' + prefixes[p] + '=' + toState + '"';       var inputs = using.querySelectorAll(selector);       if (inputs.length) {         return inputs;       }     }   });   it('should locate element using custom locator', ()=&gt;{     browser.get('http://www.way2automation.com/angularjs-protractor/registration/#/login');     browser.sleep(3000);     \$('#username').sendKeys('angular');     \$('#password').sendKeys('password');     \$('[ng-model*=\'model\']').sendKeys('Eagle');     //This is our custom locator that we created.     element(by.ngClick('Auth.login()')).click();     browser.sleep(3000);     expect(element(by.linkText('Logout')).isDisplayed()).toBe(true);   }); });</pre> |
|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

- We can now call this method with our own parameters: `element(by.ngClick('Auth.login()')).click();`  
//very common attribute on Angular.

## TAKING BASIC SCREENSHOT

- Screenshots are major part of our framework. They help with reporting, and viewing the results of your execution.
- Protractor provides the ability to take a screenshot with `browser.takeScreenshot()` function and save it.

|                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <pre>describe('Taking Basic Screenshot', ()=&gt;{   var fs = require('fs');   it('should take screenshot', ()=&gt;{     browser.waitForAngularEnabled(false);     browser.get('https://www.staples.com/');     // for taking screenshot     browser.takeScreenshot().then(function (png) {       // create stream for writing the image       var stream=fs.createWriteStream('CybertekScreenshot'+Date.now()+'.png');       // write the stream to local file       stream.write(new Buffer(png, 'base64'));       // close the stream       stream.end();     });   }); });</pre> |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## PROTRACTOR - BEAUTIFUL - REPORTER

- Another reporter available via npm.
- This reporters is visually better.
- `npm install protractor-beautiful-reporter --save-dev` (sudo ... for mac)
- Also you can define if you want capture screenshots only from failed test cases:

`takeScreenShotsOnlyForFailedSpecs: true`

to the config.js

```
let SpecReporter = require('jasmine-spec-reporter').SpecReporter;
var HtmlReporter = require('protractor-beautiful-reporter');

exports.config = {
  framework: 'jasmine',

  directConnect: 'true',
  specs: ['screenshot.js'],

  onPrepare: function() {
    browser.manage().window().maximize();
    jasmine.getEnv().addReporter(new SpecReporter({
      displayFailuresSummary: true,
      displayFailedSpec: true,
      displaySuiteNumber: true,
      displaySpecDuration: true
    }));
    // Add a screenshot reporter and store screenshots
    jasmine.getEnv().addReporter(new HtmlReporter({
      baseDirectory: 'report/screenshots'
    }));
    jasmine.getJasmine2Reporter();
  }
};
```

// Other options we will use:

```
// Add a screenshot reporter and store screenshots:
jasmine.getEnv().addReporter(new HtmlReporter({
  baseDirectory: 'report/screenshots',
  preserveDirectory: false,
  screenshotsSubfolder: 'images',
  jsonsSubfolder: 'jsons',
  docName: 'Report.html'
})).getJasmine2Reporter();
```

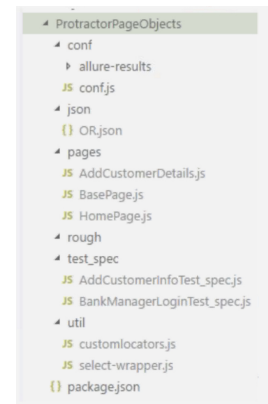
## PAGE OBJECT MODEL

- The chief problem with script maintenance is that if 10 different scripts are using the same page element, with any change in that element, you need to change all 10 scripts. This is time consuming and error prone.
- A better approach to script maintenance is to create a separate file which would find web elements, fill them or verify them.
- This class can be reused in all the scripts using that element. In future, if there is a change in the web element, we need to make the change in just 1 file and not 10 different scripts.
- This approach is called **Page Object Model(POM)**. It helps make the code more
  - **Readable**
  - **Maintainable**
  - **Reusable**
- Page Object Model is a design pattern to create **Object Repository for web UI elements**.
- Under this model, for each web page in the application, there should be corresponding page file. (e.g: loginpage)
- We also store our reusable functions in this page-object files.
- The tests then use the functions of this page object class whenever they need to interact with that page of the UI.



## FRAMEWORK STRUCTURE

- In POM Framework we organize our js files in folders
- We separate configuration files and utilities from test cases and object files
- Simple format of the Page Object model is:  
var HomePage = function(){  
  
} module.exports = new HomePage();
- The idea is to **move all the logic** required to interact with the page from the test **to the page objects**. Our suite now is more focused on the behavior of the page, than on how to select this or that element.



## PAGE OBJECT FILE & SPEC FILE

|                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>require('../util/customlocators.js'); var HomePage = function(){   this.loginAsCustomer = function(){     element(by.partialButtonText("Customer")).click();   };   this.loginAsBankManager = function(){     element(by.ngClick("manager()")).click();     return require('./AddCustomerDetails.js');   }; }; module.exports = new HomePage();</pre> | <pre>var base = require('../pages/BasePage.js'); var home_page = require('../pages/HomePage.js'); describe("BankManager Login Test",function(){   it("Login as Bank Manager",function(){     base.navigateToURL(OR.testsiteurl);     var customer = home_page.loginAsBankManager();     var title = base.getPageTitle();     expect(title).toBe("Protractor practice website -Banking App");     browser.sleep(3000);   }); });</pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

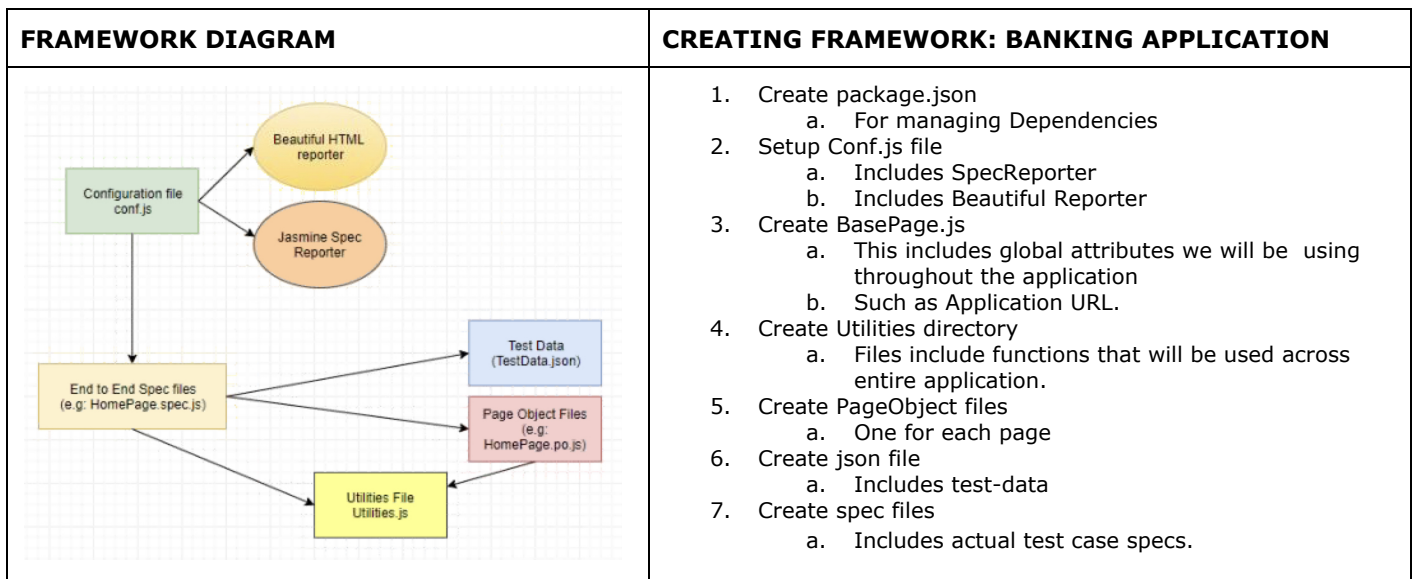
## USING IT FROM SPEC FILE

```
var toDoPage = require('../pages/toDoPage.js');
describe('Protractor Test',function(){
  it('should navigate to the AngularJS homepage',function(){
    toDoPage.go();
  });
  it('should let you add a new task',function(){
    toDoPage.addItem('New Task Item');
  });
});
```

## POM DESIGN PATTERN

|                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>this.loginAsBankManager = function(){   element(by.ngClick("manager()")).click();   //By putting below line, we are confirming that this   method returns next page's object.   //After login we are directed to the next page which is   AddCustomerDetails page.   //If this method ended up in same page, then we would   write:   // return this   return require('./AddCustomerDetails.js'); };</pre> | <pre>it("Login as Bank Manager",function(){   base.navigateToURL(OR.testsiteurl);   var customer = home_page.loginAsBankManager();   // Here we are instantiating an object from that method.   customer.gotoAddCustomer().addCustomerInfo(OR.locators.addcu   stomerdetailspage.fName,OR.locators.addcustomerdetailspage.l   Name,OR.locators.addcustomerdetailspage.pCode);   var title = base.getPageTitle();   expect(title).toBe("Protractor practice website - Banking App");   browser.sleep(3000); });</pre> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## E2E BANKING APPLICATION



## PACKAGE.JSON

- Package.json is a **build tool** available for npm packages, it can **install the required packages** to run a particular system.
  - Basically, instead of installing all of the components of our framework separately, we put them in this file and install them in one shot.
  - It will install npm packages **only if** the mentioned packages are not present in system
- It is used to install and maintain project dependencies.
  - Such as: protractor version, jasmine-reporter version etc.
- Basically whenever you try to install a npm module, it will look for package.json file to make entry of the installation, but there is no such so we got the warning.
- To create package.json file: in the project folder we can run: **npm init**
  - This will prompt us to input values.
- After we have our package.json file, we run "npm install" to install all the dependencies listed in there.
  - **npm install**.
- This becomes especially useful, when we are installing our test framework on remote systems, Erg: on jenkins server etc.

TERMINAL:

macs-MBP:CyberFramework mac\$ **sudo npm init**  
Password:  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.

See ``npm help json`` for definitive documentation on these fields and exactly what they do.

Use ``npm install <pkg>`` afterwards to install a package and save it as a dependency in the package.json file.

Press ^C at any time to quit.  
package name: (cyberframework) **cyberframework**  
version: (1.0.0)  
description: **This is for CyberFramework.**  
entry point: (index.js)  
test command: **protractor conf.js**  
git repository:  
keywords: **protractor**  
author: **CyberStar**  
license: (ISC)

About to write to /Users/mac/Desktop/CyberFramework/package.json:

```
{
  "name": "cyberframework",
  "version": "1.0.0",
  "description": "This is for CyberFramework.",
  "main": "index.js",
  "scripts": {
    "test": "protractor conf.js"
  },
  "keywords": [
    "protractor"
  ],
  "author": "CyberStar",
  "license": "ISC"
}
```

```
Is this OK? (yes) yes  
macs-MBP:CyberFramework mac$ sudo npm install protractor --save-dev  
macs-MBP:CyberFramework mac$ sudo npm install jasmine-spec-reporter  
--save-dev  
macs-MBP:CyberFramework mac$ sudo npm install  
protractor-beautiful-reporter --save-dev
```

## Conf.js

```
let SpecReporter = require('jasmine-spec-reporter').SpecReporter;
var HtmlReporter = require('protractor-beautiful-reporter');
exports.config = {
  directConnect : true,
  capabilities: {
    browserName: 'chrome'
  },
  //specs: ['../Tests/BankManagerSimple.spec.js'],
  suites : {
    smoke: ['../Tests/BankManagerSimple.spec.js', '../Tests/demo.spec.js'],
    regression: ['../Tests/*.spec.js']/* means go and run everything
  },

  onPrepare: function () {
    browser.driver.manage().window().maximize();
    jasmine.getEnv().addReporter(new SpecReporter({
      displayFailuresSummary: true,
      displayFailedSpec: true,
      displaySuiteNumber: true,
      displaySpecDuration: true,
      showstack: false
    }));
    // Add a screenshot reporter and store screenshots to `./tmp/screenshots`:
    jasmine.getEnv().addReporter(new HtmlReporter({
      baseDirectory: 'report/screenshots',
      preserveDirectory: false,
      screenshotsSubfolder: 'images',
      jsonsSubfolder: 'jsons',
      docName: 'CyberBank-Report.html'
    }));
  },
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  }
};
```

The image shows a VS Code interface with the Explorer sidebar on the left and a code editor on the right. The Explorer sidebar shows a project structure for 'CYBERBANK' with folders like Configuration, Pages, Report, TestData, Tests, and Utilities. The code editor shows the file 'BankManagerSimple.spec.js' with a list of dependencies on the left. Arrows point from the dependencies to the corresponding lines in the code. The dependencies are: 'Base.js' (line 2), 'CustomLocators.js' (line 1), 'AddCustomerPage.page.js' (line 3), 'Customers.page.js' (line 4), 'Home.page.js' (line 5), 'BankManager.page.js' (line 6), 'BankData.json' (line 7), 'AddCustomer.spec.js' (line 17), 'demo.spec.js' (line 18), and 'CustomLocators.js' (line 21). The code in the editor is as follows:

```
1 require('../Utilities/CustomLocators.js');
2 var HomePage = require('../Pages/Home.page.js');
3 var BankManagerPage = require('../Pages/BankManager.page.js');
4 var Base = require('../Utilities/Base.js');
5 var AddCustomerPage = require('../Pages/AddCustomerPage.page.js');
6 var Customers = require('../Pages/Customers.page.js');
7 var BankData = require('../TestData/BankData.json');
8
9 describe('Bank Manager', () => {
10
11   describe('Manager Login', () => {
12
13     beforeEach(function(){
14       Base.navigateToHome();
15     });
16
17     it('should have correct page title', () => {
18       expect(browser.getTitle()).toEqual("Protractor practice web");
19     });
20
21     it('should display home button', () => {
22       expect(HomePage.homeButton.isDisplayed()).toBe(true);
23       expect(HomePage.homeButton.getText()).toEqual('Home');
24     });
25   });
26 });
```

| UTILITIES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | PAGES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>CustomLocators.js</b></p> <pre>//we create this file for ng-click locator var customlocators = function() {     by.addLocator('ngClick', function(toState, parentelement) {         var using = parentelement    document;         var prefixes = ['ng-click'];         for (var p = 0; p &lt; prefixes.length; ++p) {             var selector = '[' + prefixes[p] + '=' + toState + ']';             var inputs = using.querySelectorAll(selector);             if (inputs.length) {                 return inputs;             }         }     }); } module.exports = new customlocators();</pre> <hr/> <p><b>Base.js</b></p> <pre>var Base = function() {     this.homeUrl =     'http://www.way2automation.com/angularjs-protractor/banking/#/login';     this.navigateToHome = function() {         browser.get(this.homeUrl);     }; } module.exports = new Base();</pre> | <p><b>Home.page.js</b></p> <pre>//to import the ng-Click; require('../Utilities/CustomLocators.js'); //to manage the changes of the locators; we create this; var HomePage = function() {     this.homeButton = \$('button.home');     this.pageHeader = \$('mainHeading');     this.managerLoginButton = element(by.ngClick('manager()')); }; module.exports = new HomePage();</pre> <hr/> <p><b>BankManager.page.js</b></p> <pre>require('../Utilities/CustomLocators.js'); var BankManagerPage = function() {     this.addCustomerButton = element(by.ngClick('addCust()'));     this.openAccountButton = element(by.ngClick('openAccount()'));     this.customersButton = element(by.ngClick('showCust()')); }; module.exports = new BankManagerPage();</pre> <hr/> <p><b>AddCustomerPage.page.js</b></p> <pre>require("../Utilities/CustomLocators.js"); var BankManagerPage = require('./BankManager.page');  var AddCustomerPage = function() {      this.formLabels = \$\$('form-group&gt;label');     this.firstNameInputBox = element(by.model('fName'));     this.lastNameInputBox = element(by.model('lName'));     this.postalCodeInputBox = element(by.model('postCd'));     this.formRequiredFields = element.all(by.css('input:required'));     this.formAddCustomerButton = \$('btn-default');     this.customerForm = element(by.name('myForm'));      this.goToAddCustomer = function() {         BankManagerPage.addCustomerButton.click();     }; } module.exports = new AddCustomerPage();</pre> <hr/> <p><b>Customers.page.js</b></p> <pre>require("../Utilities/CustomLocators.js");  var Customers = function() {     this.table =     element(by.xpath("//table[contains(@class, 'table-bordered')]"));      this.getLastRowValue = (function(columnNumber) {         return     this.table.element(by.xpath("//tbody/tr[last()]/td[" + columnNumber + "]"));     }); }; module.exports = new Customers();</pre> |

## BankManagerSimple.spec.js

```
require('../Utilities/CustomLocators.js');
var HomePage = require('../Pages/Home.page.js');
var BankManagerPage = require('../Pages/BankManager.page.js');
var Base = require('../Utilities/Base.js');
var AddCustomerPage = require('../Pages/AddCustomerPage.page.js');
var Customers = require('../Pages/Customers.page.js');
var BankData = require('../TestData/BankData.json')

describe('Bank Manager', () => {
  describe('Manager Login', () => {
    beforeEach(function() {
      Base.navigateToHome();
    });
    it('should have correct page title', () => {
      expect(browser.getTitle()).toEqual("Protractor practice website - Banking App");
    });
    it('should display home button', () => {
      expect(HomePage.homeButton.isDisplayed()).toBe(true);
      expect(HomePage.homeButton.getText()).toEqual('Home');
    });
    it('should display page header', () => {
      expect(HomePage.pageHeader.isDisplayed()).toBe(true);
      //expect(HomePage.pageHeader.getText()).toEqual('XYZ Bank');
      expect(HomePage.pageHeader.getText()).toEqual(BankData.appData.bankName);
    });
    it('should display login option for Bank Manager', () => {
      expect(HomePage.managerLoginButton.isDisplayed()).toBe(true);
      expect(HomePage.managerLoginButton.getText()).toEqual('Bank Manager Login');
    });
    it('should stay at the homepage when Home Button is clicked', () => {
      HomePage.homeButton.click();
      expect(browser.getTitle()).toEqual('Protractor practice website - Banking App');
      expect(HomePage.managerLoginButton.getText()).toEqual('Bank Manager Login');
    });
    it('should login as Bank Manager', function() {
      HomePage.managerLoginButton.click();
      expect(BankManagerPage.addCustomerButton.isDisplayed()).toBe(true);
    });
    it('should display options for manager', () => {
      HomePage.managerLoginButton.click();
      expect(BankManagerPage.addCustomerButton.isDisplayed()).toBe(true);
      expect(BankManagerPage.openAccountButton.isDisplayed()).toBe(true);
      expect(BankManagerPage.openAccountButton.getText()).toEqual('Open Account');
      expect(BankManagerPage.customersButton.isDisplayed()).toBe(true);
    });
    it('should navigate back to home page from Manager Login Page', () => {
      HomePage.managerLoginButton.click();
      HomePage.homeButton.click();
      expect(HomePage.managerLoginButton.getText()).toEqual('Bank Manager Login');
    });
  });
});
```

## AddCustomer.spec.js

```
require('../Utilities/CustomLocators.js');
var HomePage = require('../Pages/Home.page.js');
var BankManagerPage = require('../Pages/BankManager.page.js');
var Base = require('../Utilities/Base.js');
var AddCustomerPage = require('../Pages/AddCustomerPage.page.js');
var Customers = require('../Pages/Customers.page.js');
var BankData = require('../TestData/BankData.json')

describe('Add Customer', function() {
  describe('Adding a Customer', () => {
    beforeAll(function() {
      Base.navigateToHome();
      HomePage.managerLoginButton.click();
      AddCustomerPage.goToAddCustomer();
    });
    it('should display form for Adding Customer', () => {
      expect(AddCustomerPage.customerForm.isDisplayed()).toBe(true);
      expect(AddCustomerPage.formLabels.count()).toEqual(3);
    });
    it('should list all the labels', () => {
      expect(AddCustomerPage.formLabels.get(0).getText()).toEqual('First Name :');
      expect(AddCustomerPage.formLabels.get(1).getText()).toEqual('Last Name :');
      expect(AddCustomerPage.formLabels.get(2).getText()).toEqual('Post Code :');
    });
    it('should require first name', () => {
      expect(AddCustomerPage.formRequiredFields.get(0).getAttribute('required')).toEqual('true');
    });
    it('should require last name', () => {
      expect(AddCustomerPage.formRequiredFields.get(1).getAttribute('required')).toEqual('true');
    });
    it('should require post code', () => {
      expect(AddCustomerPage.formRequiredFields.get(2).getAttribute('required')).toEqual('true');
    });
    it('should add customer', () => {
      //AddCustomerPage.firstNameInputBox.sendKeys('Jeff');
      for (var i = 0; i < BankData.customers.length; i++) {
        AddCustomerPage.goToAddCustomer();
        AddCustomerPage.firstNameInputBox.sendKeys(BankData.customers[i].fName);
        AddCustomerPage.lastNameInputBox.sendKeys(BankData.customers[i].lName);
        AddCustomerPage.postalCodeInputBox.sendKeys(BankData.customers[i].pCode);
        AddCustomerPage.formAddCustomerButton.click();
        expect(browser.switchTo().alert().getText()).toContain('added successfully');
        browser.switchTo().alert().accept();
        BankManagerPage.customersButton.click();
        expect(Customers.getLastRowValue(1).getText()).toEqual(BankData.customers[i].fName);
        expect(Customers.getLastRowValue(2).getText()).toEqual(BankData.customers[i].lName);
        expect(Customers.getLastRowValue(3).getText()).toEqual(BankData.customers[i].pCode);
      }
    });
  });
});
```

## READING DATA and LOCATORS THROUGH JSON FILE

- We can use json file to store and retrieve :
  - Locators
  - test data
- To read data from this file we put in our spec file:  
`var objects= require('Objects.json`');`  
`browser.get(Objects.testsiteurl)`  
`element(by.xpath(Objects.locators.loginpage.username)).sendKeys(Objects.userdetails.username1);`

### BankData.json

```
{
  "appData": {
    "bankName": "XYZ Bank",
    "bankManagerLoginButtonText": "Bank Manager Login",
    "addCustomerButtonText": "Add Customer",
    "OpenAccountButtonText": "Open Account",
    "customersButtonText": "Customers"
  },
  "customers": [
    {
      "fName": "Mark",
      "lName": "Zuckerberg",
      "pCode": "21005",
      "accountNumber": ""
    },
    {
      "fName": "Jack",
      "lName": "Ma",
      "pCode": "334455",
      "accountNumber": ""
    },
    {
      "fName": "Jeff",
      "lName": "Bezos",
      "pCode": "221155",
      "accountNumber": ""
    }
  ],
  "locators": {
    "homePage": {
      "bankManagerLogin": "manager()"
    }
  }
}
```



## EXECUTING TEST CASES

There are several ways of executing test cases:

1. Executing just one test case: specs: ['functional/loginTest.spec.js']
2. Executing all spec files in the same folder: [functional/\*.spec.js]
3. Executing test suites:
  - a. Suite is a group of test cases or specs. Protractor provides a feasibility to run group of test cases or specs by dividing them as suites.

```
suites: {  
  smoke: ['./smoke/*.spec.js'],  
  regression: ['./regression/*.spec.js'],  
  functional: ['./functional/*.spec.js'],  
  all: ['./*/*.spec.js'],  
  selected: ['./functional/addcustomer.spec.js','./regression/openaccount.spec.js'],  
}
```

To run the suites, we use command: **protractor conf.js --suite smoke, selected**

We can also run different specs by passing spec parameter in command: **protractor conf.js --spec loginTest.spec.js**