Unit- V
Software Test Automation

- Dr. Shivani Budhkar

UI Automation tools

- To test UI and APIs independently, we can use several tools and frameworks, like Selenium, Cypress etc.

Cypress

- Cypress is a free, open-source, locally installed Test Runner and Dashboard Service for recording your tests.
- Cypress is a great tool with a growing feature-set. It makes setting up, writing, running, and debugging tests easy for QA automation engineers. It also has a quicker learning cycle with a good, baked-in execution environment.
- It is fully JavaScript/MochaJS-oriented with specific new APIs to make scripting easier. It also provides a flexible test execution plan that can implement significant and unexpected changes.
- It is a frontend and backend test automation tool built for the next generation of modern web applications.
- It is useful for developers as well as QA engineers to test real-life applications developed in React.js, Angular.js, Node.js, Vue.js, and other front-end technologies.

How does Cypress Work Functionally?

- Cypress is executed in the same run loop as your application.
- Behind Cypress is a Node.js server process.
- Most testing tools operate by running outside of the browser and executing remote commands across the network.
- Cypress does the opposite, while at the same time working outside of the browser for tasks that require higher privileges.
- Cypress takes snapshots of your application and enables you to time travel back to the state it was in when commands ran.

Why Cypress?

- Cypress is a JavaScript test automation solution for web applications.
- This all-in-one testing framework provides a chai assertion library with mocking and stubbing all without Selenium.
- Moreover, it supports the Mocha test framework, which can be used to develop web test automations.

Key Features of Cypress:

- **Mocking** - By mocking the server response, it has the ability to test edge cases.
- **Time Travel** - It takes snapshots as your tests run, allowing users to go back and forth in time during test scenarios.
- **Flake Resistant** – It automatically waits for commands and assertions before moving on.
- **Spies, Stubs, and Clocks** - It can verify and control the behaviour of functions, server responses, or timers.
- **Real Time Reloads** - It automatically reloads whenever you make changes to your tests.
- **Consistent Results** - It gives consistent and reliable tests that aren’t flaky.
- **Network Traffic Control** - Easily control, stub, and test edge cases without involving your server.
- **Automatic Waiting** - It automatically waits for commands and assertions without ever adding waits or sleeps to your tests. No more async hell.
- **Screenshots and Videos** - View screenshots taken automatically on failure, or videos of your entire test suite when it has run smoothly.
- **Debuggability** - Readable error messages help you to debug quickly.

Key Features of Cypress:

Cypress supports the following browsers:

- Google Chrome
- Firefox (beta)
- Chromium
- Edge
- Electron

Cypress Reporting

- Reporter options can be specified in the cypress.json configuration file or via CLI options.
- Cypress supports the following reporting capabilities:
- Mocha Built-in Reporting - As Cypress is built on top of Mocha, it has the default Mochawesome reporting
- JUnit and TeamCity - These 3rd party Mocha reporters are built into Cypress.

Here are some other usages of Cypress :

- Continuous integration and continuous deployment with Jenkins
- Behavior-driven development (BDD) using Cucumber
- Automating applications with XHR
- Test retries and retry ability
- Custom commands
- Environment variables
- Plugins
- Visual testing
- Slack integration
- Model-based testing
- GraphQL API Testing

Limitations with Cypress

- Cypress is a great tool with a great community supporting it. Although it is still young, it is being continuously developed and is quickly catching up with the other full-stack automation tools on the market.
- Here are the current limitations of using Cypress (version 5.2.0):
- It can’t use two browsers at the same time.
- It doesn't provide support for multi-tabs.
- It only supports the JavaScript language for creating test cases.
- It doesn't currently provide support for browsers like Safari and IE.
- It has limited support for iFrames.

Cypress tool -Case study 1

- Leading logistics company DHL relies on DHL Parcel software to serve 150,000 packages daily around the world. DHL Parcel provides standard domestic and international parcel pick-up, delivery and return solutions for business customers and consumers around the world.
- **By using Cypress in combination with Jenkins and Docker, the DHL Parcel team achieved:**
    - 65% faster run time
    - Increased test execution and coverage
    - Significant improvements in their front-end testing culture
- Test runs have been reduced by 65%, and feature coverage has improved dramatically. For DHL Parcel’s software team, unacceptably slow build times are a thing of the past—and team members are a lot happier when writing tests

Cypress tool -Case study 2

- GoDaddy’s GoCentral website builder powers the world’s largest cloud platform dedicated to small, independent businesses. The Growth Team’s mission is providing a world-class user experience for on-boarding and upgrading customers, working with more than a dozen other internal teams who manage many different parts of the GoDaddy platform.
- **By choosing Cypress for their test automation, the GoDaddy team achieved:**
  - 70% increase in productivity
  - 75% decrease in test maintenance
  - 2,500 tests run daily
  - 100% new feature test coverage

TestCafe

- Testcafe is a product of DevExpress which is built on top of Node.js to automate End to End web testing which overcomes all the major challenges faced using Selenium.
- Testcafe is a free and open-source Node.js tool to automate end to end scenarios.
- It runs on Windows, macOS, and Linux.
- Testcafe uses a proxy that performs URL rewriting and injects the testcript into the browser since these proxies manage all cookies/storage for the tests so we get a clean and isolated test environment

Why use Testcafe?

- Supports Multiple Browser for Parallel Execution.
- One does not need to care about waiting for an element on the page to load.
- Supports Page Object Model, Data-Driven, and BDD.
- Compatible with Continuous Integration Systems like Jenkins, Teamcity, Travis, Github Actions, etc.
- Development Language
- Testcafe Supports JavaScript as well as Typescript for writing Automation test cases.Testcafe automatically compiles Typescript before running the tests so that you do not need to compile the typescript code explicitly.
- Browser Support:- Testcafe supports all the browsers like Chrome,FireFox,Internet Explorer and Safari. Moreover it supports Mobile Browsers as well, also the browser support is for both UI and Headless.

Advantages of TestCafe Framework

- **Easy setup:** TestCafe is easy to set up. Just execute a few commands and installation is done.
- **No third-party dependency:** TestCafe doesn't depend on any third-party libraries like WebDriver, external jars, etc.
- **Easy to write tests:** TestCafe command chaining techniques make developers and testers more productive. 30 lines of code in other frameworks can be just written in 15 to 20 lines using the TestCafe framework.
- **Cross Browser Testing:** TestCafe supports all major browsers Edge, Safari, Chrome, Firefox, and IE.
- **Automated Waiting:** TestCafe waits automatically for the elements to appear. There's no need to insert External Waits.

Limitation of TestCafe Framework

- **Programming Language:** TestCafe supports only Javascript/Typescript programming language.
- **Assertion Libraries:** TestCafe has its assertion libraries, this feature can be good or bad based on the requirement. While hooking into other assertion libraries is very difficult, making it a limitation. On the other hand, it eliminates dependency and makes the framework more stable, becoming an advantage.
- **Selector Support:** By default, TestCafe supports only CSS-based selectors. If you want to use XPath you need to work around it.

TestCafe – Case study

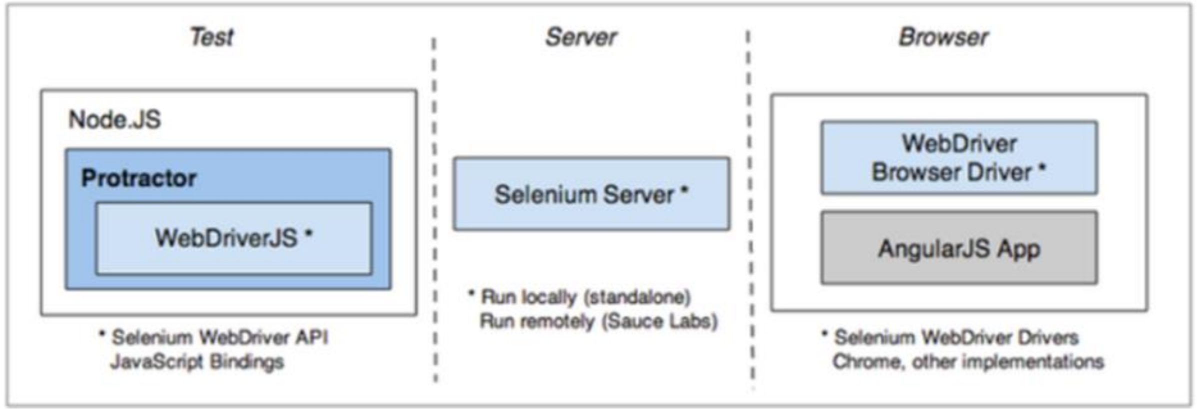
- Neos is an open-source web content management system that allows editors to master content intuitively.
- Neos CMS Increased Reliability and Confidence with E2E testing and TestCafe
- Building Confidence through TestCafe- The biggest benefit of TestCafe is its ability to build confidence within your team and in your product. In this case, if a TestCafe test passes, it means that at least half of our UI works correctly. If you compare the regressions and bugs which we had before and after we wrote our TestCafe test scripts, it is a whole lot different. Since we are open-source, we have a lot of pull requests coming in from external contributors. It is really stressful if you need to manually check everything. Now, if it is something simple, You can just merge it without opening it locally. With TestCafe, You would get confidence that not just a single module works in isolation but the whole system together would work after merge a certain PR. That is the number one thing.
- “70% of hundred our bugs could have been caught had we used TestCafe from the start”.
- Results –
- Increased reliability and confidence through TestCafe tests
- Caught 70% of regressions in advance
- Saved significant time by ending the need to check changes manually

Protractor

- Protractor is an open-source automation testing framework that is written using NodeJS.
- It offers combined end to end testing for web applications that are built using AngularJS.
- It supports both Angular and Non-Angular applications.
- But because it can be used to test advanced HTML attributes, Protractor is widely preferred for testing AngularJS.
- It leverages the power of various technologies such as NodeJS, Selenium Webdriver, Jasmine, Mocha, Cucumber, etc. to offer a strong automation test suite that is capable of performing Cross Browser Testing for web applications.

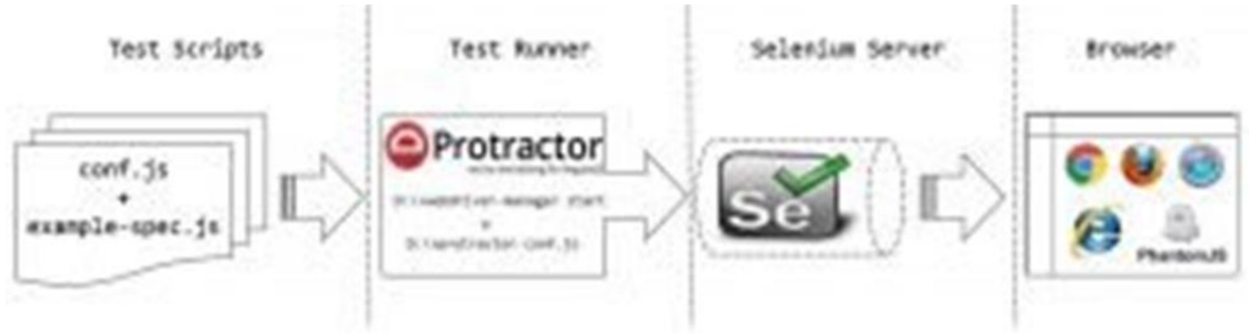
Protractor Architecture: How does it work?

- Protractor is a wrapper around Selenium Webdriver that provides an automation test framework, which simulates user interaction with an Angular web application for a range of browsers and mobile devices.
- It provides all features of Selenium WebDriver along with Angular specific features for seamless end to end testing. Protractor uses JSON Wire protocol similar to that of Selenium WebDriver to allow user interaction with the browser


Test
Server
Browser
Node.JS
Protractor
WebDriverJS *
* Selenium WebDriver API JavaScript Bindings
Selenium Server *
* Run locally (standalone) Run remotely (Sauce Labs)
WebDriver Browser Driver *
AngularJS App
* Selenium WebDriver Drivers Chrome, other implementations

Protractor Architecture: How does it work?

- The whole process comprises three elements:
  - Test Script
  - Server
  - Browser
- The Test Script interacts with the browser with the help of the Selenium server, where the commands from the test script are forwarded to one or more browsers (in case of parallel execution using Selenium Grid).
- Jasmine helps to create the test.
- Protractor helps to run the test.
- Selenium Server helps to manage browsers.
- Selenium WebDriver helps to invoke browsers APIs.



Why use Protractor for Automation Testing?

- JavaScript is used in almost all web applications. As the applications grow, JavaScript also increases in size and complexity. In such case, it becomes a difficult task for Testers to test the web application for various scenarios.
- Sometimes it is difficult to capture the web elements in AngularJS applications using JUnit or Selenium WebDriver.
- Protractor is a NodeJS program which is written in JavaScript and runs with Node to identify the web elements in AngularJS applications, and it also uses WebDriver to control the browser with user actions.
- AngularJS applications are Web Applications which uses extended HTML's syntax to express web application components. It is mainly used for dynamic web applications. These applications use less and flexible code compared with normal Web Applications.
- **Identifies Web Elements with advanced HTML attributes**
- For various Angular-based web applications, the web elements using the advanced HTML attributes like ng-controller, ng-repeater, ng-model, etc. cannot be easily tested, hindering the overall functional testing. These HTML attributes cannot be gauged by Selenium as they are not present in the Selenium Locators. Protractor helps in identifying and testing web elements using these attributes. This is why Protractor is used as a wrapper over Selenium WebDriver for automated end to end Testing of Angular-based web applications

Why use Protractor for Automation Testing?

- **Performs End to End Functional Testing**
- While other testing frameworks offer unit testing for Angular web apps, Protractor allows the tester to perform automated functional testing on Angular web apps using Selenium WebDriver. It allows testing of all layers of the application, which ensures high-quality software that is functionally robust.
- **Performs Cross Browser Testing**
- Protractor simulates user interactions by automating browsers such as Chrome, Firefox, Edge, IE, Safari, Opera, and Yandex. It does so with the help of the Browser Drivers for Selenium WebDriver like ChromeDriver, GeckoDriver, SafariDriver, etc. This allows wider coverage for Cross Browser Testing.
- **Supports Real Devices on Cloud for a wider coverage**
- When performing tests on a remote server, Protractor can be used to test Cross Browser Compatibility for a wide range of devices using a Real Device Cloud. BrowserStack's real device cloud provides access to a fleet of 2000+ desktop browsers and real mobile devices like iPhone, iPad, Galaxy, OnePlus, Pixel, Xiaomi, and Redmi, to name a few.

Why use Protractor for Automation Testing?

- **Offers Flexibility by supporting different languages for API bindings**
- Protractor is compatible with WebDriver API bindings written in different languages such as JavaScript, Java, Python, Ruby, etc., thereby offering flexibility.
- **Supports Asynchronous Test Execution**
- Protractor supports asynchronous execution, using Callbacks, Promises, and Async/Await to improve performance and make it faster. Thus, comprehensive End to End Testing can be performed on the Angular web apps in a short span of time.
- **Allows Automatic Waiting**
- Protractor offers testers the feature of Automatic Waiting, where they don’t have to add **wait(s)** and **sleep(s)** in the code. It can automatically execute the next step in the test, as soon as the webpage completes the ongoing pending tasks. Thus, waiting for the test and webpage to sync is not required when using Protractor.

Advantages of Protractor:

- Protractor runs on top of the selenium webdriver. Hence all the capabilities of webdriver are supported in protractor.
- It has extra locators compared to selenium webdriver those are model,repeater, binding etc.
- It has default waits which waits for angular which is not present in selenium webdriver.
- Easy to write and manage page objects.
- If your application is angular based then it is better to go with protractor.
- It supports behavior driven frameworks jasmine, mocha, cucumber etc.
- Image comparison is very easy in protractor and it works great.
- Running automation script in multiple machine is achieved in easy way in protractor.
- Easy installation and setup.
- Easily readable jasmine framework.
- Support Data-Driven test.
- Include all advantages of Selenium WebDriver.
- Auto-Synchronization.
- Support parallel testing through multiple browsers.
- Excellent speed.
- If you are not looking for above advantages then it is better to go with selenium. Selenium works fine with angular applications as well. Providing xpath should be defined for extra locators of angular application and waits should be defined. Selenium is in the market from long time so it has many user base and helping communities. Selenium has wide support of browsers compared to protractor.

Disadvantages of Protractor:

- It supports only javascript.
- It runs very well in chrome browser. It don’t have much support on other browsers.
- Robot class support is not there in Protractor
- Basically, a UI driven testing tool, provide only front-end testing.
- No detailed test report, other than test log.
- Even though easily understandable framework, need more time to script.
- More sensitive and Zero Tolerance.

Protractor- Case Study

- ScienceSoft’s software consulting and development company - Automated Testing for Insurance Estimation Software of a Worldwide Insurance Company
- Business gains by using Protractor-
- 20% decrease in test design costs; ~95% test coverage
- To effectively test against large datasets, the team employed the pairwise approach to test design. The technology allows saving up to 20% of test design costs if done early in the project while increasing test coverage by 20%-30%.
- ScienceSoft’s test automation team successfully handled the challenge of complex insurance calculations and managed to achieve 90-95% test coverage. After two years of development, pre-release end user acceptance testing revealed only 2 bugs.