Perfecto
by Perforce

Cypress Testing

Practical Considerations for Dev Teams

Contents

Introduction to Cypress

Cypress is a front-end dev friendly end-to-end testing framework for testing your web apps. The framework was built by developers and is for a developer audience. While this framework is relatively new, it's already building traction.

Complementing the leading Selenium WebDriver framework that has various language bindings and is built on a grid architecture, Cypress benefits its users in various other ways from the creation phase through the execution abilities.

In this eBook, you'll learn about:

- The top benefits of using Cypress for automated web app testing.
- The ways Cypress differs from the most commonly used web testing framework, Selenium.
- How to get started with Cypress automation testing.

5 Benefits of Cypress Automation for Web Testing

Selenium has long been the de facto framework for web testing. But Selenium is not your only option for web testing. Interest is growing for the emerging framework, Cypress. Here are the key benefits of using Cypress for automated web app testing.

1. CYPRESS IS MORE UNIVERSAL

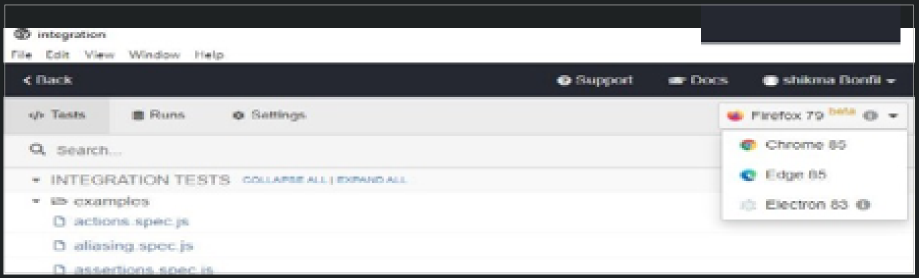
When compared to other automation frameworks, Cypress is more universal. That's because it is written in Javascript and based on Mocha and Chai. It also uses Node.js when running in browsers. So, why does that matter? JavaScript is the main language used by developers to develop websites. Hence, testing is created in the same language.

With Cypress, you can run cross-browser testing. Execute tests with Cypress on Firefox and browsers within the Chrome family, such as Edge and Electron.

2. CYPRESS IS SIMPLE TO SET UP

It's easy to get started with Cypress automation for web testing. If you've worked with Selenium, you know that before you start testing, you need to select all the dependencies and libraries needed for that suite of tests. But you don't have to worry about those things with Cypress. They're already set in place, with no configuration needed.

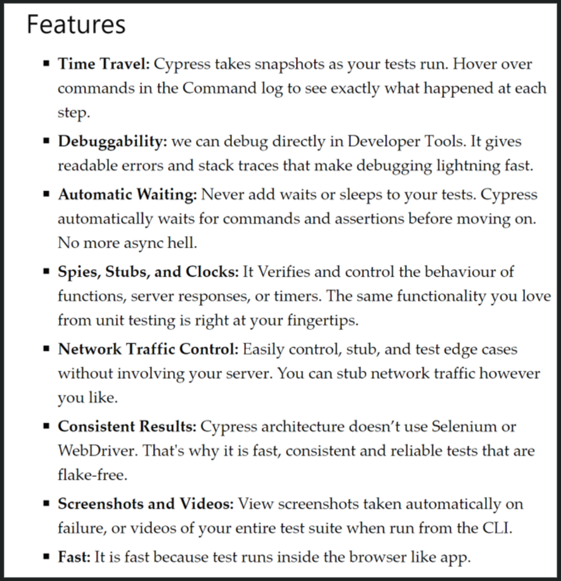
And unlike Selenium, where you have to download a relevant driver and set up a grid to get started testing, Cypress comes bundled with a Chrome browser, so there is no complex environment to set up. In addition, any other browser that is installed on your local machine can be used for testing with Cypress (as the below image shows).



3. CYPRESS HAS DEBUGGING CAPABILITIES

You can debug your web apps from Cypress quickly and easily. When tests fail, you get suggestions for how to fix the defect. From there, you can debug directly from Chrome DevTools. Also, Cypress supports capabilities like Time Travel and real-time reload so developers can examine their website code during test execution and after.

And because Cypress has access to every object, it simplifies and streamlines error analysis. Cypress also can provide screenshots of test failures, which makes finding defects and debugging apps quick and simple.



Features

- **Time Travel:** Cypress takes snapshots as your tests run. Hover over commands in the Command log to see exactly what happened at each step.
- **Debuggability:** we can debug directly in Developer Tools. It gives readable errors and stack traces that make debugging lightning fast.
- **Automatic Waiting:** Never add waits or sleeps to your tests. Cypress automatically waits for commands and assertions before moving on. No more async hell.
- **Spies, Stubs, and Clocks:** It Verifies and control the behaviour of functions, server responses, or timers. The same functionality you love from unit testing is right at your fingertips.
- **Network Traffic Control:** Easily control, stub, and test edge cases without involving your server. You can stub network traffic however you like.
- **Consistent Results:** Cypress architecture doesn't use Selenium or WebDriver. That's why it is fast, consistent and reliable tests that are flake-free.
- **Screenshots and Videos:** View screenshots taken automatically on failure, or videos of your entire test suite when run from the CLI.
- **Fast:** It is fast because test runs inside the browser like app.

4. CYPRESS OFFERS FAST TEST EXECUTION

Cypress is known for its speed of execution — with a response t ime of less than 20 MS. Cypress has automatic waiting built into the framework, which means that you don't need to define implicit and explicit waits. The framework automatically waits for things like DOM loading, animation, elements, and more.

And the framework also automatically runs tests after completion of another. This eliminates downtime and the need to manually trigger the next test.

The other way Cypress accelerates testing is by improving developer productivity. The skillset required to test with Cypress is very similar to that of developers, as opposed to Selenium which is out of their realm.

So, developers can easily create UI tests in Javascript right from their workstation. They can then run these tests locally before committing any code. We are seeing more and more developers taking on this responsibility within their role. This in-cycle testing can vastly speed up the entire process.

5. CYPRESS HAS AN ACTIVE COMMUNITY

Cypress is a free and open source framework. It operates on a freemium model, where you can use the free version or paid version. The paid version includes advanced features, such as a dashboard with artifacts, such as DOM snapshots, which are helpful for debugging, as well as video storage.

Interest is growing in Cypress, and it boasts an active community on GitHub, Gitter, and StackOverflow. Plus, Cypress offers robust documentation.

Comparing Cypress vs. Selenium: Which Framework Is Right for You?

Cypress and Selenium are both automation frameworks for web app testing. Selenium is an established solution, while Cypress is emerging. Cypress supports JavaScript, while Selenium supports many languages. Cypress supports end-to-end testing. Selenium does too, but also offers security and unit testing.

Selenium is without a doubt the de-facto test automation framework for cross-browser testing — and it has been for many years. But now, Cypress is gaining traction. With dozens of variations of frameworks that were built on its protocol, WebDriver, momentum in the market is growing for Cypress.io.

Keep reading for a breakdown of Cypress vs. Selenium and to explore some of the objectives that can help teams decide which option is right for them — or if using both is better.

INTRODUCTION TO SELENIUM AND CYPRESS TESTING

First, let's define what Selenium and Cypress testing are and what they offer practitioners.

What Selenium Automates

Selenium is a test automation tool that enables developers to automate web browser testing. The Selenium WebDriver protocol enables sending commands in various development languages — like Java, Java Script, C#, Python, and others — from the test environment (IDEs) to a selected desktop browser (Chrome, Firefox, Edge, Safari).

Each of the browsers has its own WebDriver that is a dependency for the test to be able to communicate and perform the required actions (click, swipe, assert, etc.).

As a leading solution, Selenium serves as a foundation to common test frameworks like Protractor, WebDriverIO, and others, as well as mobile app testing frameworks like Appium. Looking at the market trends around adoption and downloads, it is obvious that Selenium is a key enabler for browser test automation.

We can also see the growth in adoption of Cypress testing that has passed WebDriverIO in the number of downloads.

What Cypress Testing Automates

Cypress is a JavaScript test automation solution for web applications. It enables teams to create web test automation scripts. This solution aims to enable frontend developers and test automation engineers to write web tests in the de-facto web language that is JavaScript.

Cypress also supports the Mocha test framework so the core technologies in which you would develop your web test automation are Java Script on top of Mocha.

Pros and Cons: Cypress vs. Selenium

While there is no right or wrong with these two frameworks, it is important to distinguish them and understand some of the benefits and in some cases the disadvantages each framework has.

TEST FRAMEWORK	SUPPORTED DEV LANGUAGES	SUPPORTED BROWSERS	SUPPORTED TEST FRAMEWORKS	SETUP & EXECUTION	INTEGRATIONS	BREADTH OF TESTING OPTIONS	MATURITY, DOCUMENTATION, SUPPORT
SELENIUM WEBDRIVER	Java, C#, Java Script, Python, Ruby, Objective-C	Chrome, Safari, Firefox, Edge, IE	Mocha JS, Jest, Other super set on top of Selenium (Protractor, WebDriverIO, etc.	Download relevant driver, set up a grid, network and location impacts execution speed	Plenty of integrations (CI, CD, reporting, cisual testing, cloud vendors)	End-to-end, security, unit	Robust community, multiple bindings, best practices
CYPRESS.IO	JavaScript, TypeScript, Cucumber	Chrome, Electron, Edge, Firefox	Mocha JS	Comes with bundled Chrome browser, can run on any local installed browsers, no complex environment setup	Limited integrations	End-to-end	Good documentation and code sample, growing community

The two solutions provide different capabilities as in the above table. There are other elements, like the simplicity of writing scripts, reporting, and dashboards, but as a whole and in every tool selection, it all depends on the team's objectives, skillset, scope to test, and other product-specific considerations.

Cypress is a great growing tool. It is fast to ramp up with and provides a good execution environment that is baked in. It is fully JavaScript/MochaJS oriented with specific new APIs to ease the scripting.

On the other hand, Selenium is a mature framework covering multiple browsers with different development languages. And it works well within a grid to scale testing. The element of test flakiness is debatable between the two tools.

Some would argue that Cypress testing produces a more robust and reliable test scripts, while Selenium experts can provide good practices to overcome such problems.

WHICH IS BETTER — CYPRESS VS. SELENIUM?

When it comes to debating Cypress vs. Selenium, we recommend that teams start exploring Cypress to see if it can complement their existing Selenium scripts and grow their overall test coverage and stability.

If you already have a good working Selenium suite that is stable and covers enough functionality, there is no real urgency to switch tools. If you are starting a new project, perhaps having a simple POC with Cypress can prove to be a good future solution for you.

How to Get Started With Cypress

To get started with the Cypress tool, you would need to download the NPM package through ***"npm install cypress"*** as well as ***"npm install mocha"*** or download the actual Cypress desktop application from the cypress.io website.

The solution comes with a nice set of JavaScript examples. Once you launch the Cypress tool (./node_modules/bin/cypress open) you can trigger them to run with an embedded Chrome browser that the tools come built with. The fact that the solution already bundles a browser ready to be used makes the ramping up and execution even easier.

Cypress, while built on top of MochaJS, comes with a large set of APIs to leverage browser-specific commands, like setting viewport, clear cookies, basic browsing commands, and more.

An example of a test written in Cypress would look something like this:

The **cy.visit** will simply navigate with the Chrome browser to the URL. **cy.get** will perform the required action on a specific element on the page like click, perform assertions, or type text.

```
context('Waiting', () => {
  beforeEach(() => {
    cy.visit('https://example.cypress.io/commands/waiting')
  })
  // BE CAREFUL of adding unnecessary wait times.
  // https://on.cypress.io/best-practices#Unnecessary-Waiting

  // https://on.cypress.io/wait
  it('cy.wait() - wait for a specific amount of time', () => {
    cy.get('.wait-input1').type('Wait 1000ms after typing')
    cy.wait(1000)
    cy.get('.wait-input2').type('Wait 1000ms after typing')
    cy.wait(1000)
    cy.get('.wait-input3').type('Wait 1000ms after typing')
    cy.wait(1000)
  })

  it('cy.wait() - wait for a specific route', () => {
    cy.server()

    // Listen to GET to comments/1
    cy.route('GET', 'comments/*').as('getComment')

    // we have code that gets a comment when
    // the button is clicked in scripts.js
    cy.get('.network-btn').click()

    // wait for GET comments/1
    cy.wait('@getComment').its('status').should('eq', 200)
  })

})
```

Practical Considerations for Teams Using Cypress

Cypress is a versatile framework that excels in many areas of E2E testing. As teams start using Cypress, they can run into a number of challenges. These tend to emerge as teams try to scale their Cypress testing efforts. Here are the most notable challenges teams face.

1. SCALABILITY

Whereas Selenium has Selenium Grid for scaling, it’s much more complicated with Cypress — think Kubernetes, docker, etc. And you’ll need a DevOps Engineer to orchestrate it all. Overall, it can be expensive and difficult to scale Cypress yourself.

Perfecto makes Cypress easier to scale. Because Perfecto is a cloud-based solution, the infrastructure provides virtually unlimited scaling, and you can run your tests faster with parallel execution in the cloud.

For example, running your tests sequentially on Cypress alone would take you 35 minutes to complete a regression. With Perfecto’s parallel execution, those same tests would take 5 minutes.

2. BROWSER MAINTENANCE

If you're using only Cypress, you're also going to need to spend time maintaining browsers. If you don't have the resources to make sure all browsers are up to date and available, then you may need to utilize a commercial solution alongside Cypress.

Utilizing a cloud platform like Perfecto takes the pressure of platform maintenance off your team.

For example, with Perfecto everything you need is always up to date with the latest versions or releases, and it’s always available 24/7 in the testing cloud. Plus, you can get same-day support for new browser versions, so there is never any downtime for new releases.

Additionally, extended test coverage in the cloud enables you to include more browser/OS permutations in your tests.

3. ENTERPRISE-GRADE SECURITY

If security is of utmost importance to you, like for those in the financial sector, then Cypress alone may not be the right fit. Cypress is an open source framework. Many enterprises run into roadblocks when using open source solutions. One reason f or that is a lack of security.

Open source testing frameworks, such as Cypress, lack the security standards that many enterprises seek. But by using Cypress with Perfecto, you'll get the added benefit of enterprise-grade security in your testing.

The cloud provides secure testing access for teams all over the world, and those working remotely too. In addition to unshakeable security, Perfecto also provides teams with access to internal environments.

4. REPORTING & ANALYTICS

The role of a developer covers many things, but reporting is not one of them. Yet reporting and analytics are a critical function of any successful test automation strategy.

Cypress does offer a dashboard function in the paid version of the framework. However, reporting only reflects Cypress tests. Teams that use a more than one framework in their toolstack will have to find a way to aggregate reporting or compile reporting data from multiple sources.

Instead of sifting through console output from Cypress, you can get actionable insights from Perfecto's built-in test reporting and analytics. It provides big picture and CI pipeline insights that can speed up defect resolution and give you single pane of glass view into Cypress and everything else you're running.

Developers can take this fast feedback and quickly fix any defects, thanks to ML-powered test failure analysis. Faster feedback translates into less downtime, and more development.

Using Cypress With Perfecto

As a leader in the testing space, Perfecto is proud to already offer full support for the Cypress test automation framework. Keep reading to learn how to get started with Cypress testing — with Cypress executing web tests on browsers in the Perfecto cloud.

HOW TO GET STARTED WITH CYPRESS TESTING

To get started with Perfecto and Cypress, follow these steps:

1. Install the Perfecto-Cypress SDK through: **"npm I perfecto-cypress-sdk"**
2. After installing, run the init command to generate your initial perfecto-config.json file:perfecto-cypress init --tests.path= ./tests/"

In this step you can also define the cloud name, the Cypress relevant project ID, and more.

3. To get support for the Perfecto reporting SDK, install the Cypress reporting package by running this command: **npm install perfecto-cypress-reporter** and then import it inside the support file cypress/support/index.js through this command: **import 'perfecto-cypress-reporter'; // or require('perfecto-cypress-reporter');**
4. In addition to the abovementioned **init** command, the perfecto-cypress-sdk supports various commands:

- **Run**
- **Pack**
- **Upload**

```
$ npx perfecto-cypress --help
bin.js <comand>

Comands:
      bin.js init       init Perfecto and Cypress configuration files
      bin.js pack       Zip tests files according to configurations
      bin.js run        Run Cypress tests on Perfecto cloud
      bin.js upload           Upload tests zip archive to Perfecto cloud repository

Options:
      --version   Show version number                        [boolean]
      --help            Show help                            [boolean]
```

To learn each, simply run the command on the left.

5. With the new SDK, you can either run your Cypress test specs from the command-line by pointing to a test.spec.js file, or through an **artifact key** that points to a packed test suite in the Perfecto testing cloud.

Below are the supported command options and their shortcut aliases, including the known reporting SDK commands that should be passed as well.

```
$ npx perfecto-cypress --help
bin.js run

Run Cypress tests on Perfecto cloud

Credentials options (will override config file)
       --credentials.cloud, --cloud            Cloud name                  [string]
       --credentials.securityToken, -- token   Offline token               [string]

Test options (will override config file)
       --tests.artifactKey, --ta        Repository artifact key            [string]
       --tests.path, --tp               Root path for test to pack         [string]
       --tests.ignore, --ti             ignore files list                  [array]
       --tests.specsExt, -- ts          specs file extension               [string] [default: "*.spec.js"]

Reporting options (will override config file)
https://developers.perfetomobile.com/display/PD/Download+the+Reporting+SDK
       -- reporting.jobName, --rjn      reporting job name                 [string]
       --reporting, jobNumber, --rjNum  reporting job number               [number]
       --reporting, branch, --rb        reporting branch                   [string]
       --reporting, projectName, --rpn  reporting project name             [string]
       --reporting, projectVersion, --rpv      reporting project version   [string]
       --reporting, author, --ra        reporting author                   [string]
       --reportingm tags, --rt          reporting tags, ex:tagl tag2       [array]
       --reporting.customeFields, --rcf reporting custom fields,           [array]
                                        ex. key1,value1 key2,value2
Options:
       --version     Show version number                                   [boolean]
       --help        Show help                                             [boolean]
       --env, -e     environment variables to attach to cypres run command
       --config, -c  Path to config file, see documemntation: [default: "./perfecto-config.json"]
```

6. When uploading a Cypress test suite to the Perfecto cloud, utilize the below command options. It is recommended to **pass the securityToken as a parameter** rather than storing it in the perfecto-config.json file.

```
$ npx perfecto-cypress --help
bin.js upload

Upload tests zip archive to Perfecto cloud repository

Credentials options (will override config file)
      --credentials.cloud, --cloud        Cloud name                                [string]
      --credentials.securityToken, --token      Offline token                             [string]

Options:
      --version         Show version number                                         [boolean]
      --help                  Show help                                             [boolean]
      --archivePath, -p Path to tests zip file [default: "./perfecto-cypress.zip"]
      --temporary, -t   Upload tests archive as temp artifact                       [boolean] [default: true]
      --folderType, -f  Set the location of tests archive in the repository
                        [choices: "PRIVATE", "PUBLIC", "GROUP"]                     [default: "PRIVATE"]
      --config, -c            Path to config file, see documentation:               [default: "./perfecto-config.json"]
```

7. Perfecto's Cypress SDK is fully compliant with Cypress. It supports Chrome, Firefox, and Edge browsers across the various permutations. Configuring the platform capabilities is done within the perfecto-config.json file (see below example).

```
"capabilities":   [
      {
         "deviceType": "Web",
         "platformName": "Windows",
         "platformVersion": "10",
         "browserName": "Chrome",
         "browserVersion": "latest",
         "resolution": "1024x768",
         "location": "US East",
      }
```

CYPRESS TESTING WITH PERFECTO: HOW THEY WORK TOGETHER

Open source only takes you so far. Perfecto is an end-to-end cloud testing platform that works with tools like Cypress, Jenkins, Selenium, and more, and syncs them all together so you can execute at scale.

With Perfecto and Cypress, you can:

- Test faster — run Cypress tests in parallel and at scale, rather than sequentially with just Cypress alone.
- Be more secure — run Cypress tests from the cloud and secured repository from command-line of CI (e.g. Jenkins).
- Spend more time testing — rather than maintaining browsers, Perfecto does that for you.
- Fix defects faster — utilize the Perfecto reporting SDK within Cypress test execution for enhanced quality visibility, tagging, and app debugging purposes.

Summary

Open source solutions can only take you so far. Pair Cypress with Perfecto for even faster web testing at scale. Access the browser permutations you need to test against in the cloud. And move faster with parallel testing and bursting.

Perfecto's Cypress-SDK package was built to provide rich and advanced capabilities for web application front-end developers. The SDK supports uploading a suite of Cypress tests to the Perfecto cloud-based repository for command line and continuous integration (CI) execution.

In addition, the SDK allows users to execute Cypress tests at scale and in parallel, integrated with the Perfecto reporting platform. It gives users the ability to slice and dice Cypress tests vs. Selenium and others.

A snippet of the Perfecto Cypress SDK capabilities under the RUN command are to the right:

```
npx perfecto-cypress run

Run Cypress tests on Perfecto cloud
      Credentials options (will override config file)
      --credentials,cloud, --cloud Cloud Name                [string]
      --credentials.securityToken, --token Offline token     [string]

      Test options (wil override config file)
      --tests.artifactKey, --ta Repository artifact key      [string]
      --test,path, --tp Root path for test to pack           [string]
      --test.ignore, --ti ignore files list                  [array]
      --tests.specsExt, --ts specs files extention           [string]
                                              [default: "*.spec.js"]
```

See Cypress and Perfecto in action in a demo.

WATCH DEMO

About Perfecto

Perfecto by Perforce enables exceptional digital experiences and helps you strengthen every interaction with a quality-first approach for web and mobile apps through a cloud-based test platform. The cloud is comprised of real devices, emulators, and simulators, along with real end-user conditions, giving you the truest test environment available. Our customers, including 50 percent of the Fortune 500 companies across banking, insurance, retail, telecommunications, and media rely on Perfecto to deliver optimal mobile app functionality and end-user experiences, ensuring their brand's reputation, establishing loyal customers, and continually attracting new users. For more information about Perfecto, visit **perfecto.io**.

TRY PERFECTO

Perfecto