Automation Testing

Setting Up the Cypress

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
   cout << "Hello, world!" << endl;
   return 0;
}
```

A Day in the Life of an Automation Test Engineer

Mark is working in an organization as an Automation Testing Engineer.

He has been asked to test a basic website using Cypress tool.

To achieve this, he will learn the structure of Cypress, its installation and its framework which will help him to find a solution for the given scenario.

Learning Objectives

By the end of this lesson, you will be able to:

- Describe Cypress
- Comprehend the Cypress framework
- Define the features of the Cypress framework
- Describe the installation of Cypress

What is Cypress?

Defining Cypress

Cypress is a JavaScript online test automation tool that allows teams to develop web test automation scripts.

Cypress supports the Mocha test framework, therefore the primary technology for developing web test automation would be Java Script on top of Mocha.

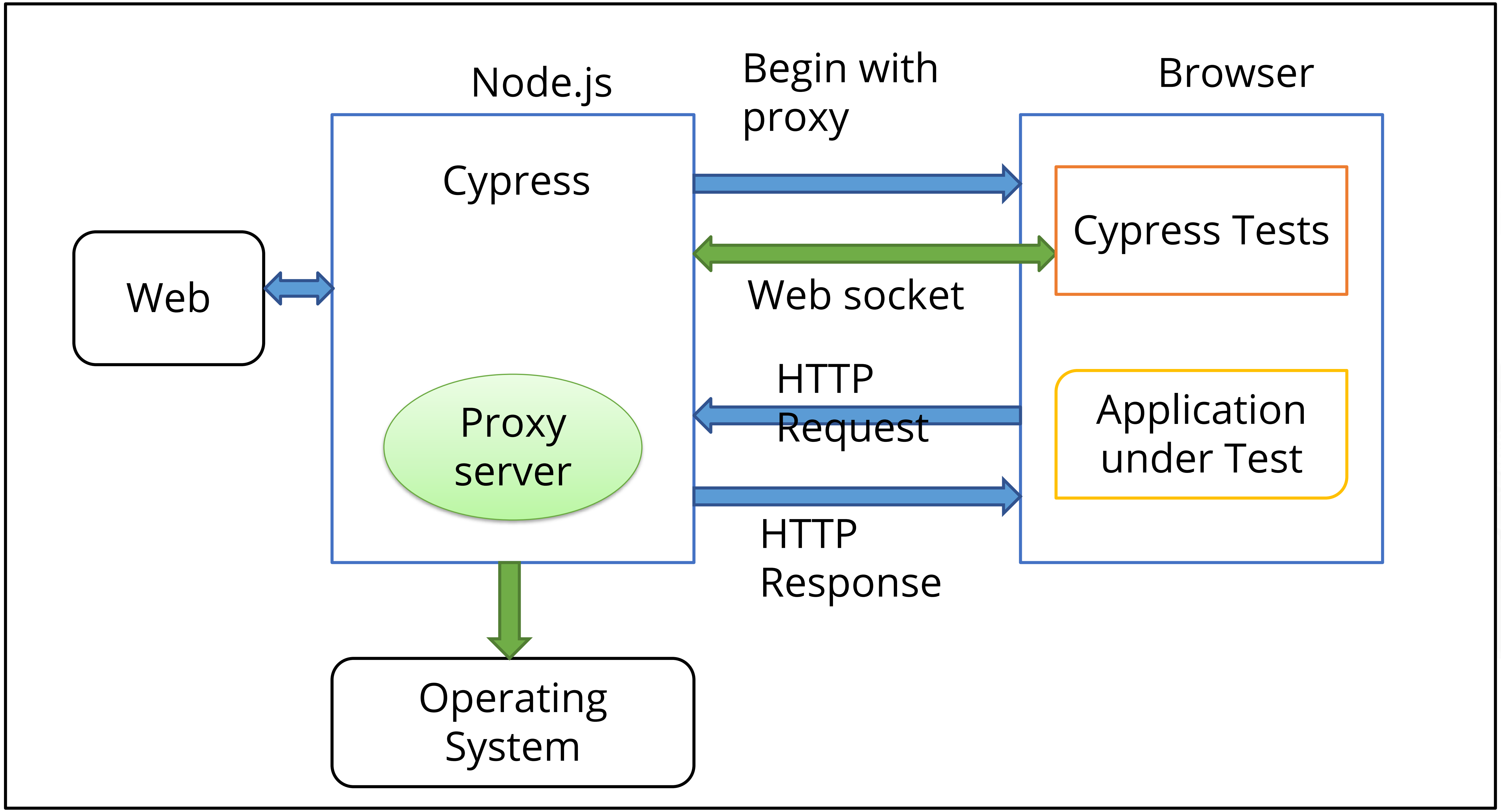
Who Uses Cypress?

Most users are often web developers or QA engineers who utilize current JavaScript frameworks to create web apps.

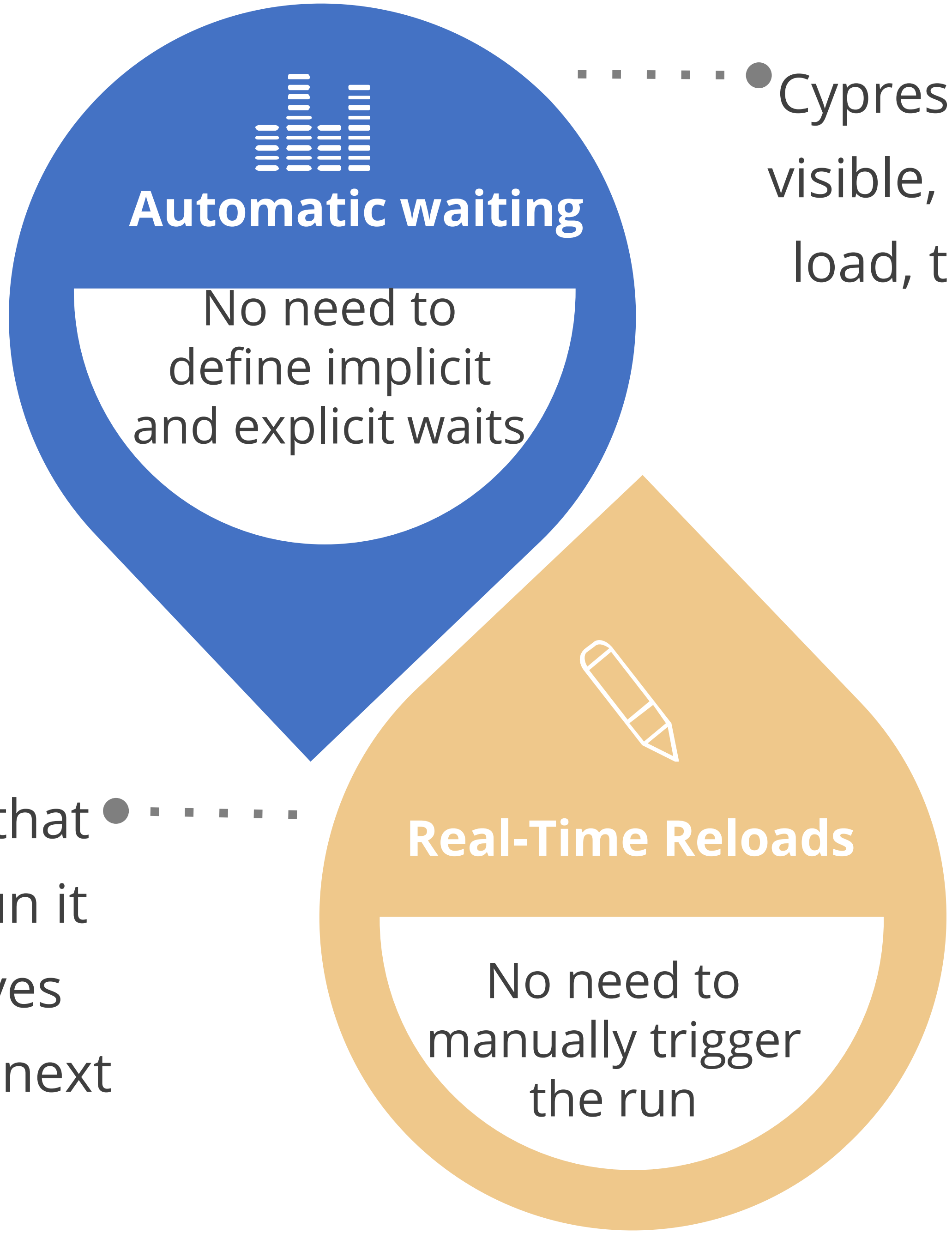
Developers

Testers

Cypress Architecture (Test Automation)



Primary Features for the Cypress Framework



Automatic waiting

No need to define implicit and explicit waits

Cypress waits for the elements to become visible, the animation to finish, the DOM to load, the XHR and AJAX calls to complete, and so on.

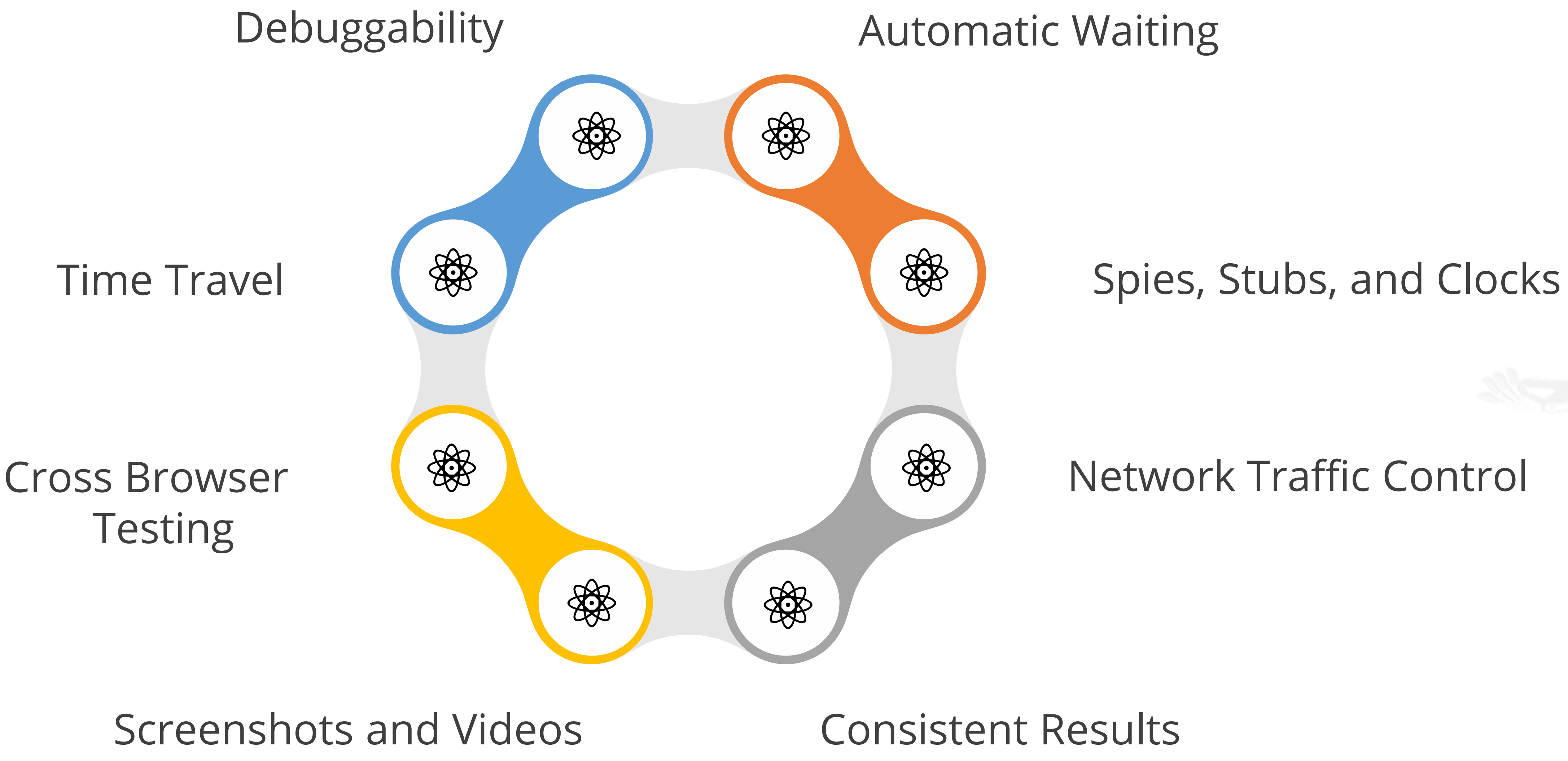
Real-Time Reloads

No need to manually trigger the run

Cypress is smart enough to recognize that after saving a test file, the tester will run it again. As a result, when the tester saves their file, it automatically starts the run next to the browser.

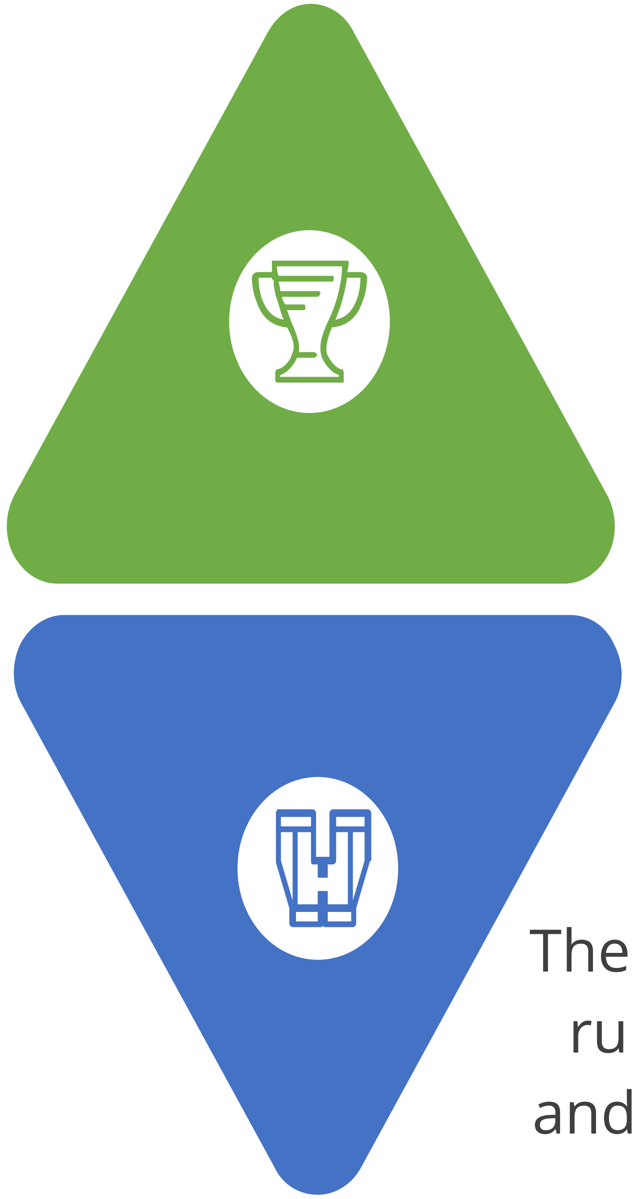
General Features of the Cypress Framework

The general features of the Cypress framework can be described as the following:



Cypress Ecosystem

Cypress assists users in setting up and starting to write tests daily while they create the application locally.

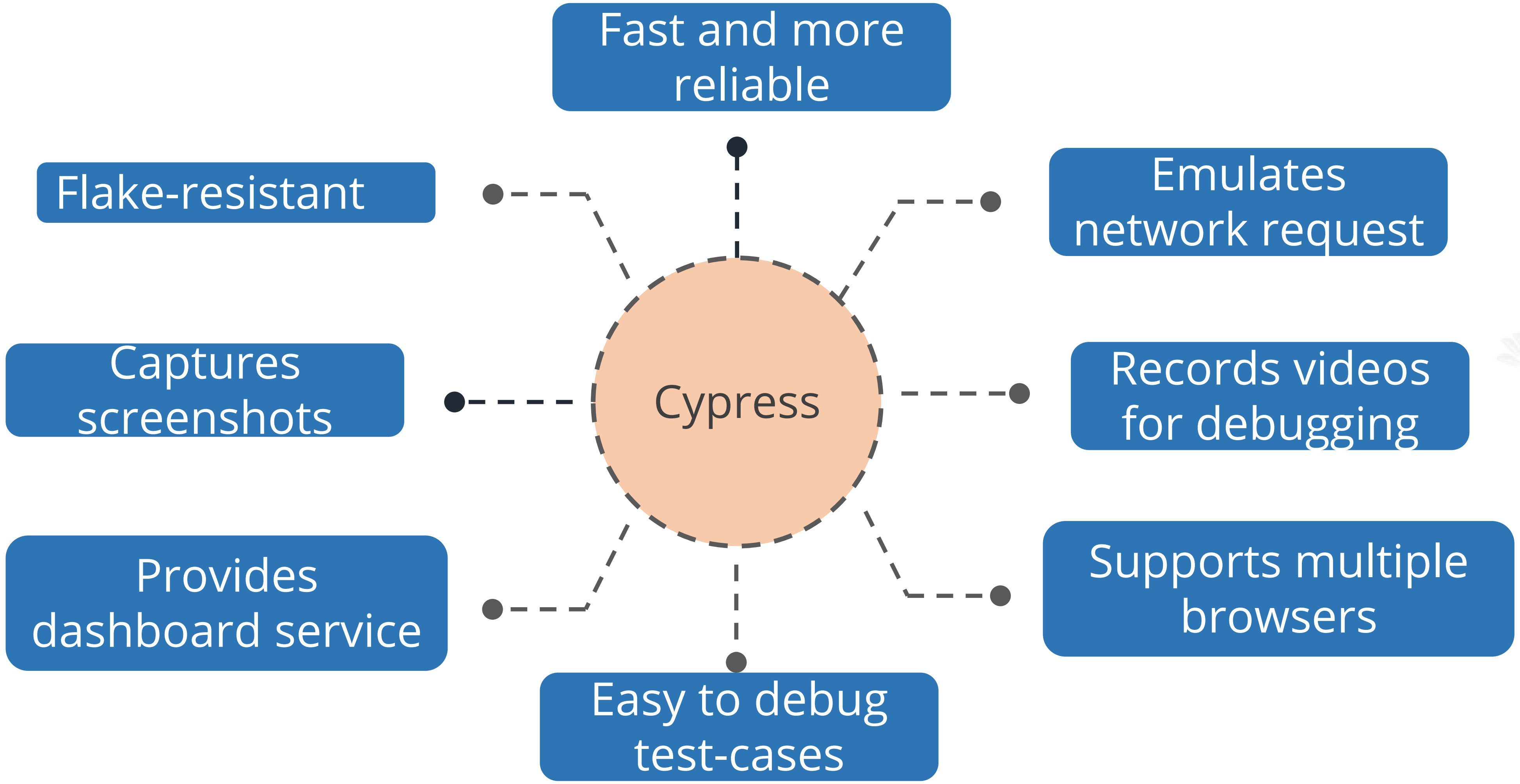


The Dashboard Service can record the test runs after users have built a suite of tests and integrated Cypress with the CI Provider.

Why Use Cypress?

Reasons for Using the Cypress

There are several advantages of using Cypress testing instead of using other web testing frameworks:



Cypress vs Selenium

Cypress

- It runs in the same browser and run loop as the device under test.
- Cypress is simpler to set up than Selenium.
- It supports only Js.

Selenium

- It runs in a process separate from the browser or the device under test.
- Its configuration is rather complicated.
- It supports a variety of programming languages, including Java and Python.

Cypress vs Selenium

- It supports only one browser, Chrome.
- It uses DOM manipulation methods to do tasks.
- It is tough in nature.

Selenium

- It can work with a variety of browsers.
- It does not make use of any DOM modification methods.
- It is not tough by nature.

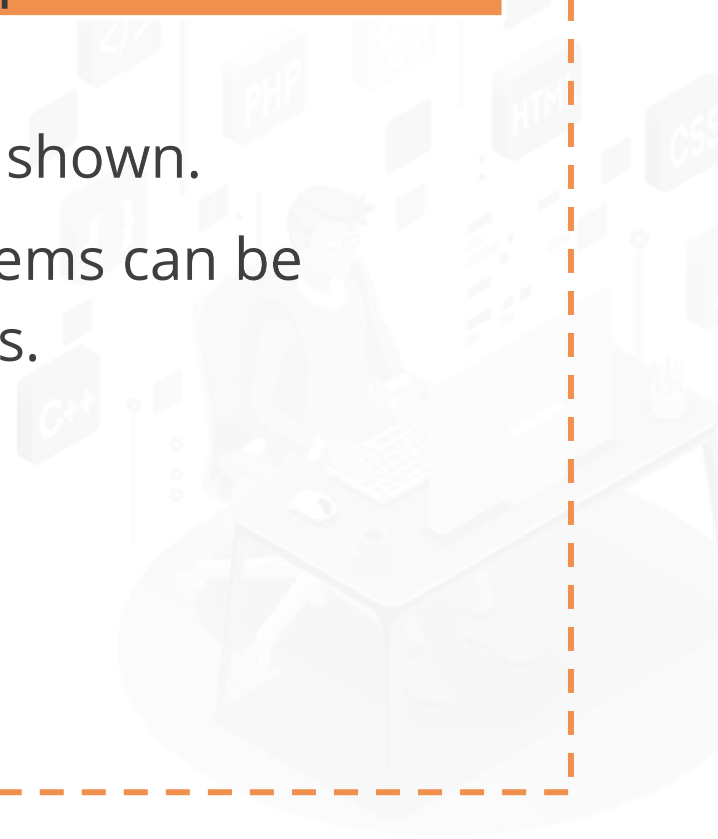
Cypress vs Selenium

Cypress

- There is no support for multiple tabs.
- Operating system support is limited.

Selenium

- Multiple tabs can be shown.
- Many operating systems can be used to process tasks.



Installing the Cypress

Installation Prerequisite

Cypress must be preloaded on the user's computer.

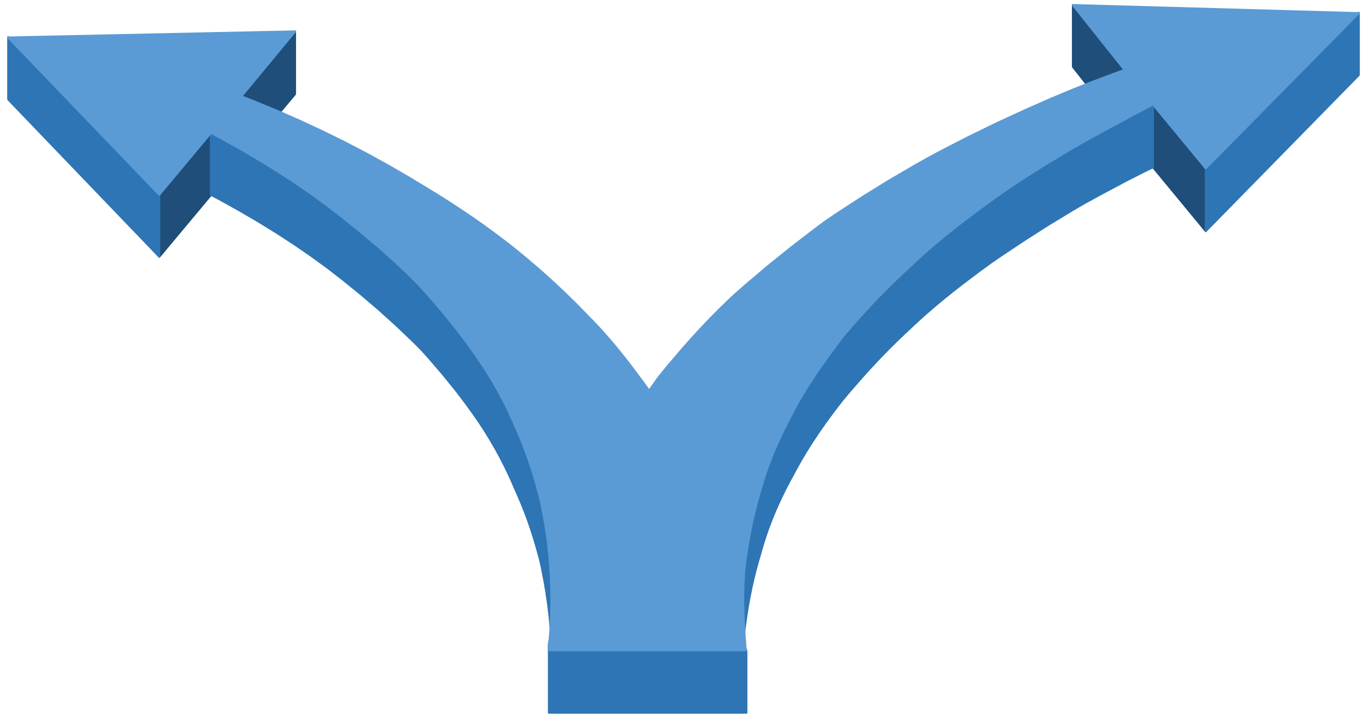
Node.js may be downloaded and installed for a specific operating system from their download page.

Execute the command node –version in the terminal to double-check the Node.js installation.

A code editor, such as Microsoft's Visual Studio Code, is also required.

Setting Up the Cypress

Cypress may be installed in two ways:



Using NPM

Downloading Cypress Directly

Cypress

Cypress Using NPM

Navigate to the project directory and run the following command to install Cypress using the npm (Node package manager):

```
npm init
```

The package.json file is created by the above command, provide only the most basic information, such as the package name, description, keywords, and author name.

Cypress Using NPM

Users can run the below command to install Cypress once the previous steps are completed:

```
npm install cypress --save-dev
```

The command above adds Cypress to the project's local development environment as a dev dependency.

Cypress Using NPM

Users who want to employ the Yarn package management to install Cypress should go to the project directory and run the following command:

```
yarn add cypress --dev
```

Setting up the Cypress by Downloading the Cypress Directly

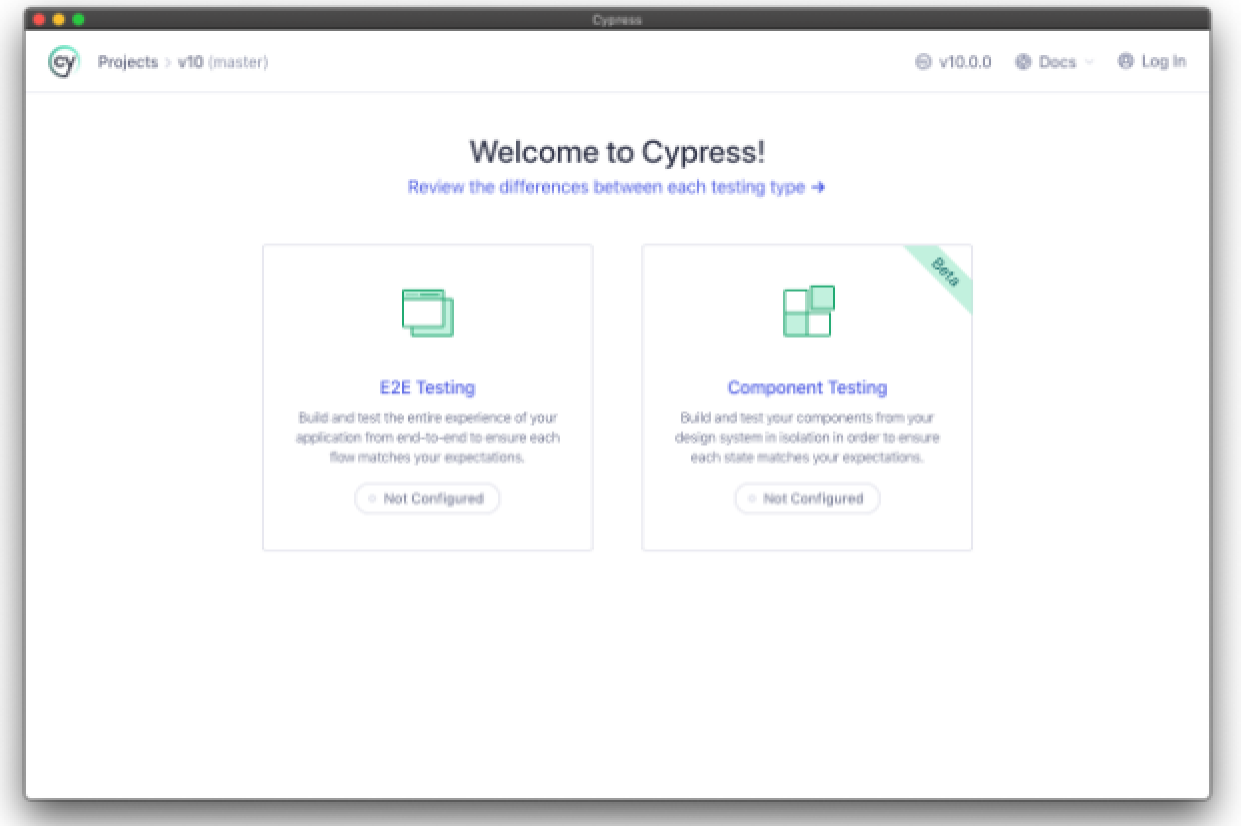
Cypress can be set up directly by following the below steps:

- Users who aren't familiar with npm but want to try Cypress right away can do so by downloading the Cypress zip file from the CDN.
- The direct download link recognizes the user's platform and downloads the current version of Cypress as a zip file.
- After downloading the zip file, unzip it and double-click it to launch the Cypress programme.
- It will start without requiring any prerequisites to be installed.
- The installation of Cypress is now complete.

Using the Cypress Launchpad

The Launchpad

When users open Cypress, the testing experience starts with the Launchpad. Its role is to assist users through the decisions and configuration activities one must accomplish before beginning to write their first test.



Welcome to Cypress!

Review the differences between each testing type →

E2E Testing

Build and test the entire experience of your application from end-to-end to ensure each flow matches your expectations.

Component Testing

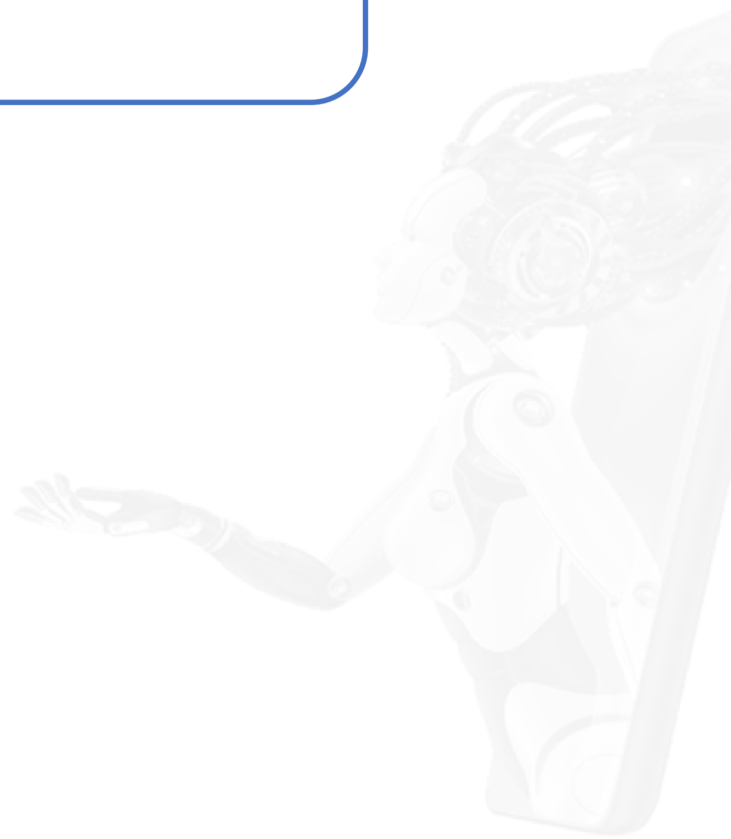
Build and test your components from your design system in isolation in order to ensure each state matches your expectations.

https://docs.cypress.io/guides/getting-started/opening-the-app

Selecting a Testing Type

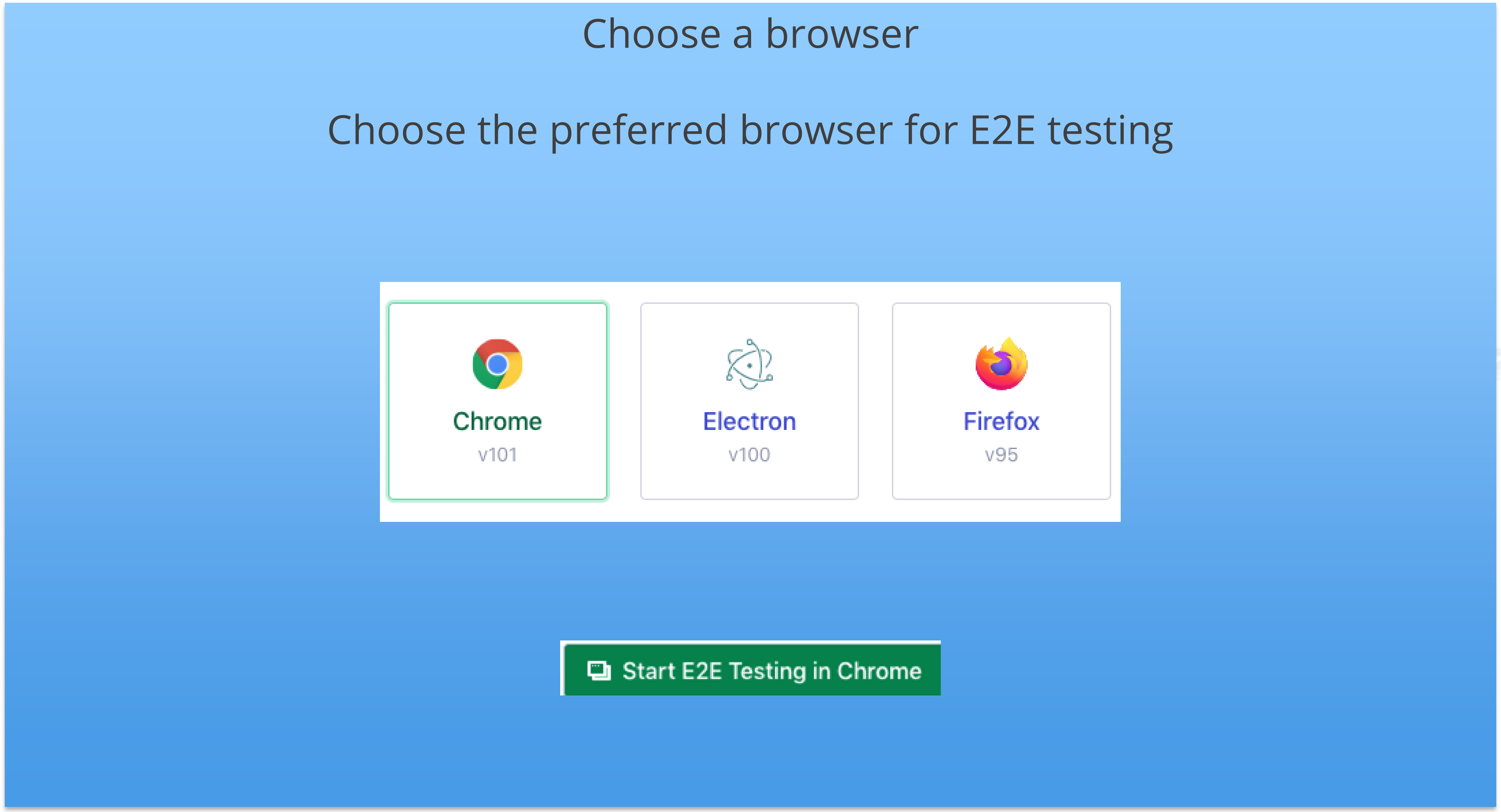
The Launchpad starts with the most important decision: What kind of testing should users conduct? E2E testing, in which they run their entire program and visit various pages to test them? Or Component Testing, in which they mount and test specific components of the program in isolation?

- ✓ Tested
- ✓ Tested
- ✓ Tested



Launching a Browser

Cypress presents a list of compatible browsers found on the system.

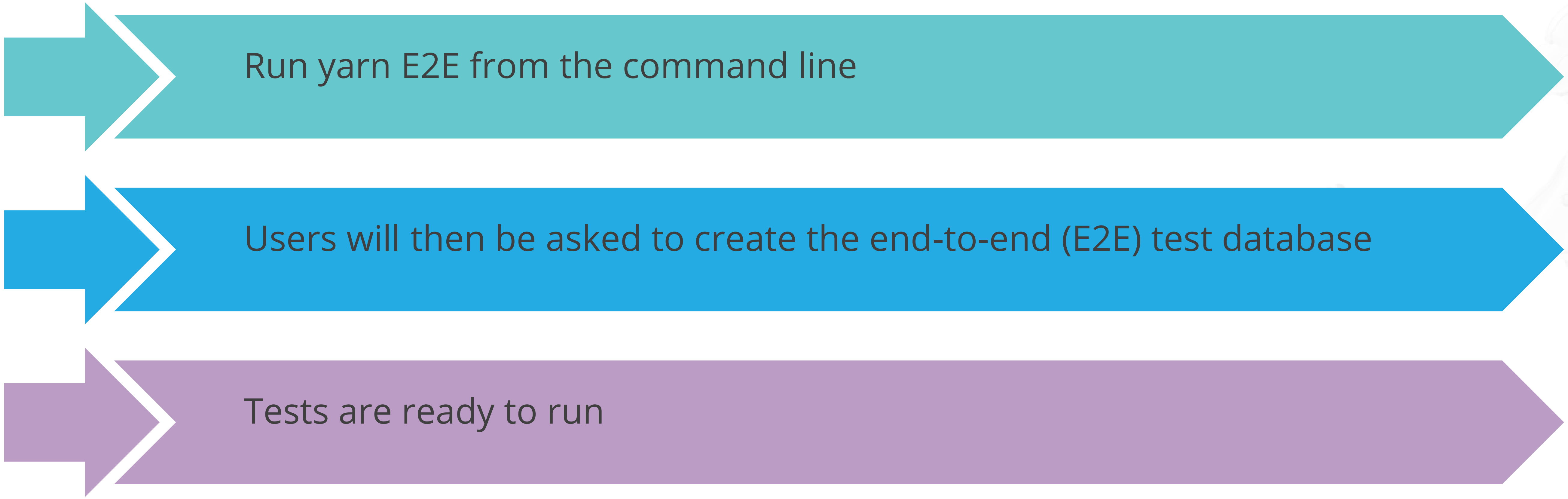


Choose a browser

Choose the preferred browser for E2E testing

Running E2E Tests Locally

There are three ways to run E2E tests locally:



Run yarn E2E from the command line

Users will then be asked to create the end-to-end (E2E) test database

Tests are ready to run

Custom Commands for Test Setup

Some useful custom commands for test setup include:

cy.loginAndVisit(user, url)

cy.loginUser(user)

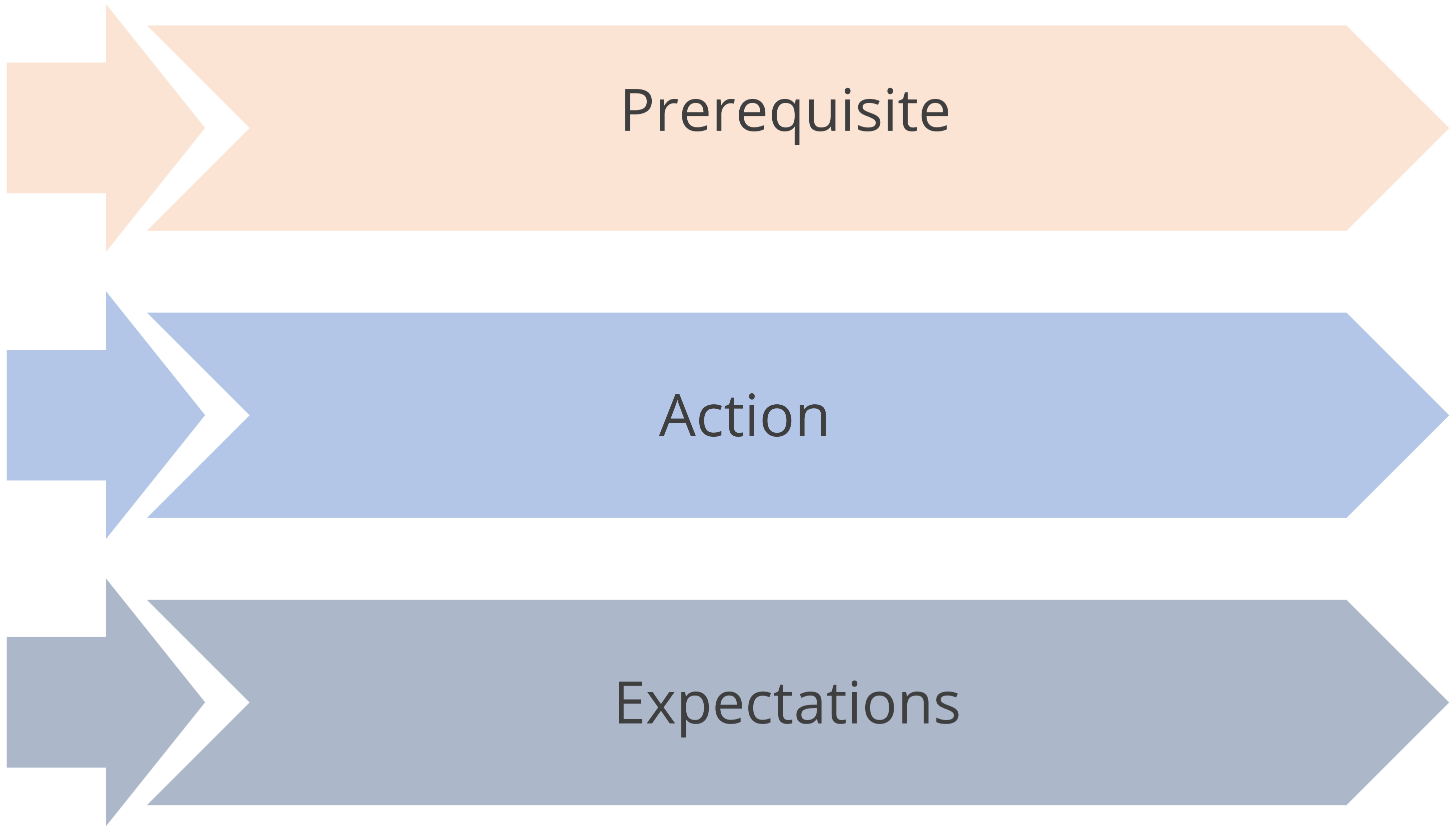
cy.createArticle(articleData)

cy.visitAndWaitForUserSide Effects(url)

cy.signOutUser()

Steps to Write a Cypress Test Case

Each test is typically comprised of three steps:



Prerequisite

Action

Expectations

Writing the First Test Case for Cypress Automation

The cypressdemo folder includes:

- Node_modules folder
- Cypress folder
- Cypress.json file
- Package-lock.json file
- Package.json file

Installing Cypress and Opening Launchpad

Problem Statement:

You have been asked to install Cypress and open Launchpad.

Assisted Practice: Guidelines

Steps to install Cypress and open Launchpad are:

1. Install Cypress and open Launchpad

Key Takeaways

- Cypress is a Mocha-based JavaScript-based end-to-end testing framework that runs on and in the browser, making asynchronous testing simple and convenient.

- Cypress is resistant to flaking, faster, and more trustworthy, and it can simulate XHR or network requests (mocking) - Network Traffic Management.

- Cypress can be installed either by using npm or downloading cypress directly.

- Cypress must be downloaded on the user's PC because it is a Node.js-based application.