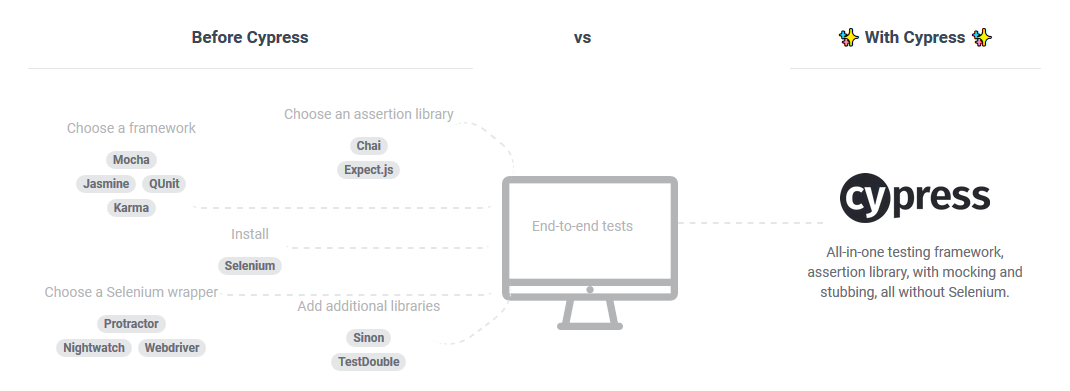
**Cypress vs WebdriverIO Report**

**1. Some facts about Cypress**

Cypress is a next generation front end testing tool built for the modern web. [1]



Cypress ecosystem consists of a Test Runner and a Dashboard Service for recording the tests. [1]

* Cypress does not use Selenium.
* Cypress focuses on doing end-to-end testing
* Cypress works on any front-end framework or website.
* Cypress tests are only written in JavaScript.
* Cypress is all in one
* Cypress doesn’t support cross browser testing, only support Chrome
* Cypress doesn’t support native mobile apps, but they do intend to support mobile web browsers in the future

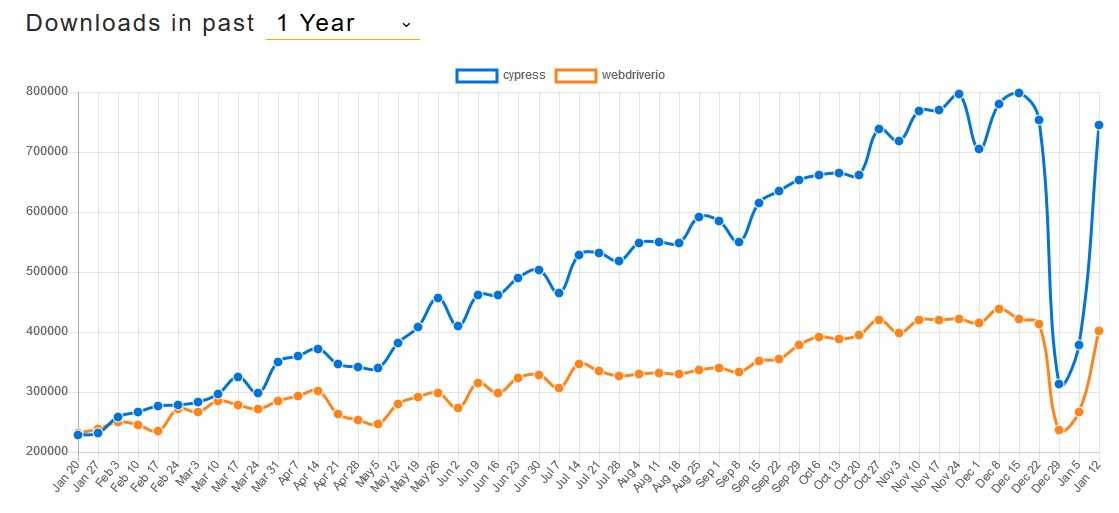
**2. Cypress vs WebdriverIO**

Below is a comparison between Cypress and WebdriverIO, it shows the pros and cons of each in 4 tasks:

* Setting up tests
* Writing tests
* Running tests
* Debugging

|  |  |  |
| --- | --- | --- |
|  | **Cypress** | **WebdriverIO** |
| **1. Setting up tests** | | |
| Setup | Cypress is all in one, with Cypress we get multiple tools in one installation: test framework Mocha, assertion libraries Chai, Sinon, jQuery and other library utilities, just doing an 'npm install cypress'. It's less time to set up locally | It takes more time for researching and installing which test frame works, reporters, test runners, assertions, and services to use. However, it is more flexible in choosing the tools to use |
| Supported Languages | JavaScript | JavaScript |
| Supported Browsers | Chrome variants (Canary, Chrome, Chromium, Electron) | Cross browser testing |
| Mobile support | No | Yes |
| Third-party services integration | There are no third-party integrations | It had the option to integrate with services like BrowserStack and Sauce Labs for cross-browser and device testing |
| Popularity | Cypress with 17.5K GitHub stars and 1K GitHub forks | WebdriverIO with 5.4K GitHub stars and 1.6K GitHub forks |
| Documentation | Very good! Cypress documentation looks very nice. It is also much more in-depth and broad. It covers most of what we want to know, without the need to go looking elsewhere on the web | Good! A good documentation with detailed instructions, logical organization, rich example projects |
| **Conclusion**: In this task, both Cypress and WebdriverIO have their own strengths. Besides, they also have limitations, but not significantly.  Result: **Cypress: 1 – WebdriverIO: 1** | | |
| **2. Writing tests** | | |
| API/ Commands/ Syntax | Tests written in Cypress are easy to read and understand. It supports a lot of APIs and they come fully baked, on top of tools we are familiar with already.  Besides, it also has limitations such as:  - No multiple domain support. We can only visit a single unique domain per test.  - Requires workarounds to test with iframes – There are some issues around handling iframes with Cypress. https://github.com/cypress-io/cypress/issues/136 | WebdriverIO is simple and easy to write syntax. It also supports multi domains in a test as well as can work with iframes |
| **Conclusion**: In this task, both Cypress and WebdriverIO are easy to write and understand, they both are strong in their own purpose  Result: **Cypress: 1 – WebdriverIO: 1** | | |
| **3. Running tests** | | |
| Start running tests | Only one step to start running tests in Cypress. We just need to open Cypress and run the test once. After that, when we change something in code, the test is run automatically in the browser | With WebdriverIO, we need 2 steps to start running tests - Run browser driver  - Run the tests in WebdriverIO for each test run |
| Test running | Cypress runs in the context of the browser and in the same run loop as the device under test. Cypress executes the vast majority of its commands inside the browser, so there is no network lag and gives us the ability to respond to the application's events in real time | WebdriverIO runs outside of the browser where the application is running. It has three processes: - WebdriverIO  - A browser driver, such as ChromeDriver, GeckoDriver (for Firefox), EdgeDriver, SafariDriver, etc.  - The browser itself.  That's why it easily has network lag problems. |
| Cypress will automatically wait for the application to reach this state before moving on. We are completely insulated from fussing with manual waits or retries. Cypress automatically waits for elements to exist and will never yield us stale elements that have been detached from the DOM.  => Cypress is less flaky, tests passed way more consistently | Before we can start writing code to assert that the test is correct, we need to ensure that whatever elements we need to interact with are visible and are in a state to accept simulated input. We need to write custom waitForSomething code helpers  => A test can exhibit different behavior when executed with the same inputs at different times |
| **Conclusion**: With Cypress GUI, test runs are easier and more visible than WebdriverIO. The test results in Cypress are also more stable and reliable than in WebdriverIO  Result: **Cypress: 1 – WebdriverIO: 0** | | |
| **4. Debugging** | | |
| Test Runner | Cypress test runner has GUI. When running the Cypress GUI, one has access to all of the browser console to see some helpful output, time travel debug and pause at certain commands in the command log to see before and after screenshots, inspect the DOM with the selector playground, and discern right away at which command the test failed. That's really helpful for debugging a test When we opted to run the Cypress tests in headless mode, we got console errors, screenshots, and video recordings. | WebdriverIO test runner using command line interface. We observe the tests run over and over in a browser and then the console errors shown. It would not point us toward and would sometimes even lead us astray from where the test really failed in the code.  With WebdriverIO we only had some screenshots for failed tests |
| **Conclusion**: with Cypress GUI, Cypress is more powerful for debugging than WebdriverIO. Besides, it can debug directly from familiar tools like Developer Tools  Result: **Cypress: 1 – WebdriverIO: 0** | | |
| **Final result: Cypress: 4 – WebdriverIO: 2** | | |

The download trend of Cypress and WebdriverIO in past 1 year. We can see that number of Cypress downloads are growing more than WebdriverIO downloads. At the beginning of the year, they are the same but now, Cypress are 2 times higher. [2]



**Practical experiment**

To verify the above comparisons, I conducted a practical experiment: automation tests for Homepage page, using both Cypress and WebdriverIO [3]. There are a total of 16 tests used to test the modules and sub-modules on the Homepage page and their operation. The tests in Cypress and WebdriverIO have exactly the same scenarios and follow the same steps. There are some different behaviors between Cypress and WebdriverIO that occurred in this experiment:

- ***Steps of setting up and running tests:***

Setting up:

* Cypress: installing with “npm install cypress”, all necessary dependencies are installed
* WebdriverIO: installing webdriverio and then other necessary dependencies, browser driver.

Running test:

* Cypress: open Cypress: “$(npm bin)/cypress open” and select the test to run for the first time. After that, the test is run automatically each time code is changed
* WebdriverIO:
  + Run chromedriver
  + Run tests for every test run

- ***The number of lines of code of Cypress and WebdriverIO*** are not much different. With every command in the WebdriverIO, we can almost find a corresponding command in Cypress to replace. But in some cases, Cypress is better at having some extra commands such as checking/ unchecking checkbox. That makes Cypress code is easier and shorter to write.

Cypress code:

unselectDefaultHomepageCheckbox(){  
 this.scrollToHomepageCheckBox();  
 ***cy***.wait(500);  
 this.homepageBox.check();  
 ***cy***.wait(500);  
 this.defaultHomepageBox.uncheck();  
 this.save.click();  
}

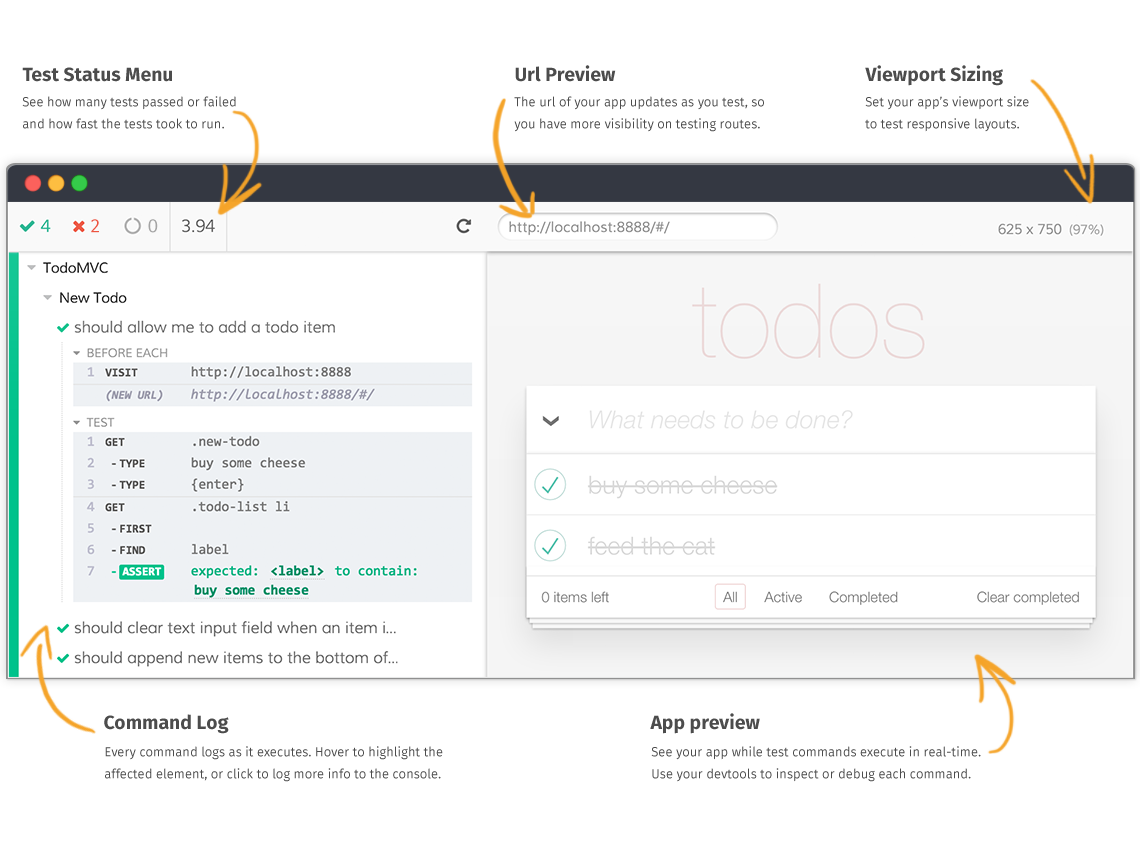
WebdriverIO code:

unselectDefaultHomepageCheckbox(){  
 this.scrollToHomepageCheckBox();  
 this.homepage.waitForVisible();  
 if (!this.homepageBox.isSelected())  
 {  
 this.homepage.click();  
 this.save.click();  
 }  
 else  
 {  
 this.defaultHomepage.waitForVisible();  
 if (this.defaultHomepageBox.isSelected())  
 {  
 this.defaultHomepage.click();  
 this.save.click();  
 }  
 }  
}

- ***The test runner***: this is a highlight of Cypress compared to WebdriverIO.

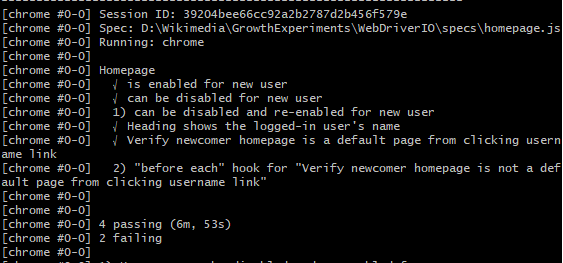
As we can see in the screenshot [4], when the test runs, the commands that are executed are shown in the side panel, while they are being executed. Moreover, once the test ends, we can click on each command in the side panel, and we can see a screenshot of how the page looked like when the command was executing. This is called Time traveling.

This is phenomenal for debugging a test! Cypress screenshots and seeing how the tests progress is a marvelous way to debug the tests, and a marvelous way to start to understand why the test failed. The whole experience helps in writing, understanding, and debugging tests, and in analyzing their results.

- ***Test run time***: I ran the 16 tests of Cypress and WebdriverIO 10 times and took the average value. The results are: ***Cypress took 690 seconds*** to complete the entire tests while the ***WebdriverIO took 590 seconds*** to complete. WebdriverIO runs a little faster than Cypress (however, one WebdriverIO test run did not finish yet)

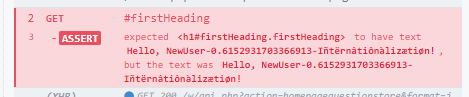
- ***Consistent Results***: Cypress has ***the same result in 10 test runs*** while WebdriverIO has ***6 different results in 10 test runs***. Even in one test run, WebdriverIO was unable to complete all 16 tests when it failed at "before each" hook of sixth test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test run** | **Cypress** | | **WebdriverIO** | |
| **Time** | **Failed Tests** | **Time** | **Failed Tests** |
| 1st | 900.5 | 4, 10 | 560 | 3, 6, 13 |
| 2nd | 653.61 | 4, 10 | 566 | 3, 13 |
| 3rd | 690.43 | 4, 10 | 554 | 3, 6, 13 |
| 4th | 695.42 | 4, 10 | 567 | 2, 3, 6, 13 |
| 5th | 698.05 | 4, 10 | 413 | 3, could not run from 6th because beforeEach hook failed |
| 6th | 709.31 | 4, 10 | 739 | 2, 3, 6 |
| 7th | 715.68 | 4, 10 | 641 | 3, 6, 13 |
| 8th | 556.77 | 4, 10 | 601 | 2, 6, 13 |
| 9th | 717.12 | 4, 10 | 583 | 2, 3, 6, 13 |
| 10th | 565.72 | 4, 10 | 674 | 3, 6, 13 |
| **Average** | **690.261** |  | **589.8** |  |

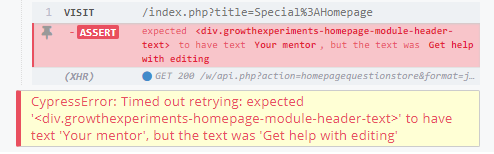


- ***Special failed cases***: there are some special failed cases in Cypress which are passed in WebdriverIO

* Verify text of username in heading



* Xpath was changed in Cypress browser leads to test failure



**Conclusion:**

As we can see, on the surface, although the two tools Cypress and WebdriverIO – seem similar, there are actually many differences between the two. Although Cypress still has some limitations compared with WebdriverIO, but for end-to-end testing, these limitations are mostly not a problem, and enable Cypress to give frontend developers the advantages that they do need: speed and consistency. In my opinion, **Cypress is better than WebdriverIO in end-to-end testing**

**References:**

[1] <https://www.cypress.io/how-it-works>

[2] <https://www.npmtrends.com/cypress-vs-webdriverio>

[3] <https://github.com/simkawaii/GrowthExperiement_Cypress_WebdriverIO>

[4] <https://docs.cypress.io/guides/core-concepts/test-runner.html#Overview>