WebDrivers

Hexlet

Web Drivers

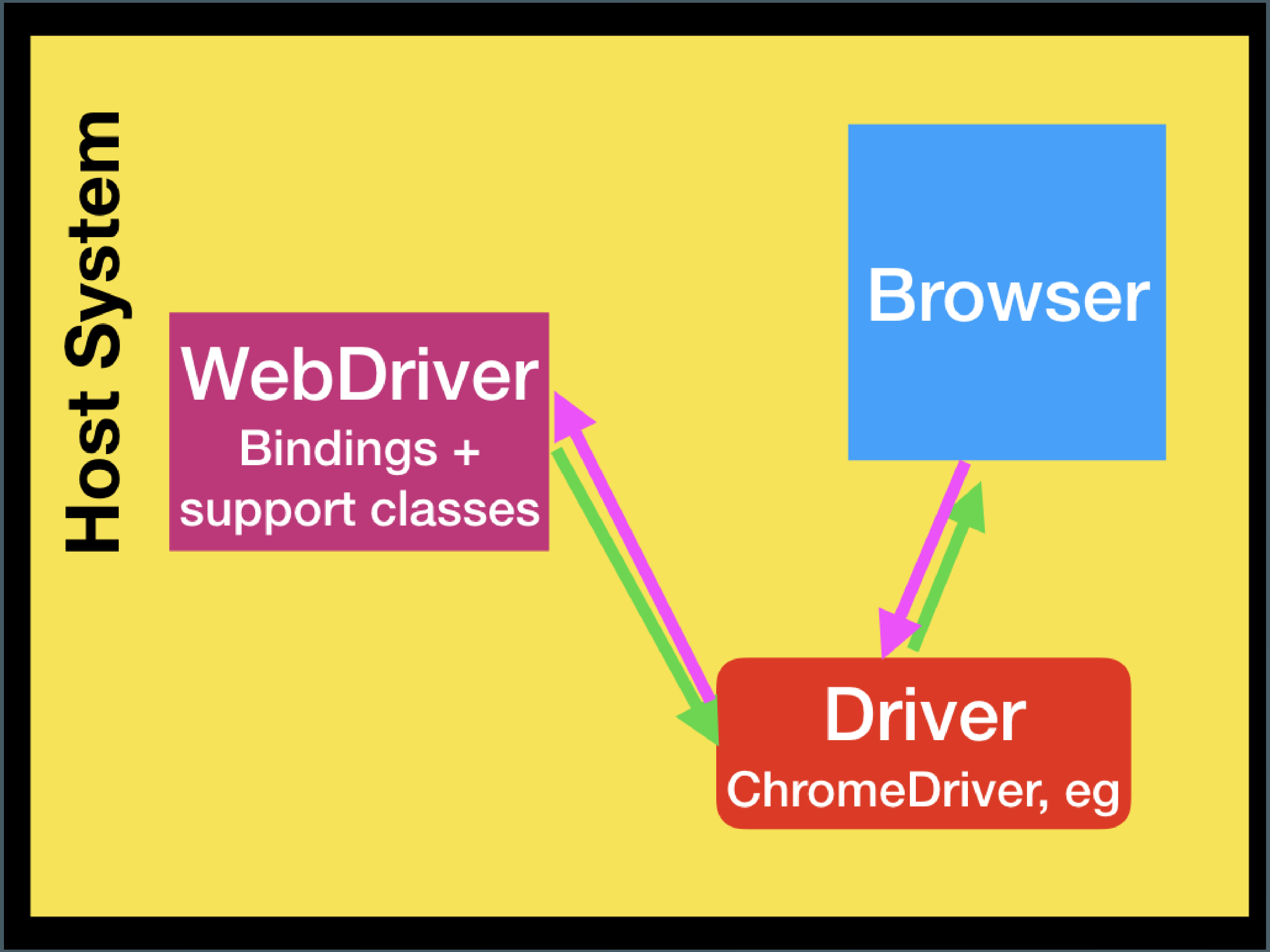
- Интерфейс удаленного управления, который позволяет анализировать и управлять браузером
- Платформо независимый (от языка тоже)
- Предоставляет набор интерфейсов для нахождения и управления элементами DOM
- Не имеет прямого отношения к тестированию

Selenium

- Popular framework to automate your test processes on your front-end applications
- Open-source testing solution
- Include:
  - Selenium IDE
  - Selenium WebDriver
  - Selenium Grid
  - Selenium Standalone Server

- Can be run on different platforms:
  - Windows
  - Linux
  - macOS
  - Android / IOS
  - Xamarin, React Native, or NativeScript
- Selenium WebDriver supports all major browsers
  - Chrome
  - IE
  - Edge
  - Firefox
  - Safari
  - Opera

- Programming languages supported by the Selenium project:
  - Java
  - C#
  - Ruby
  - Python
  - Kotlin
  - JavaScript
  - PHP
  - Perl / Objective-C / Haskell / R
- Different browsers require different drivers to allow WebDriver to communicate with and control them
- Many drivers and clients



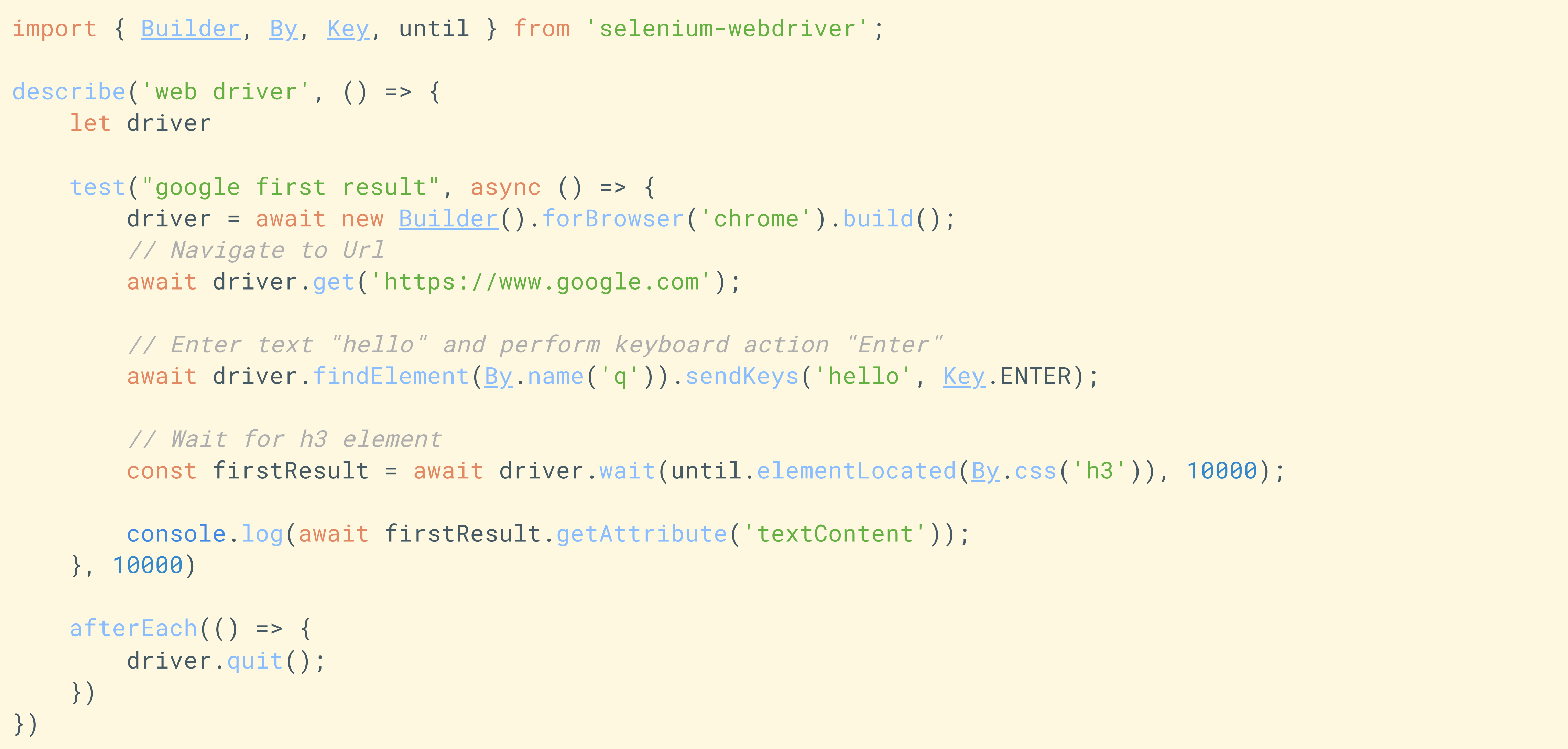
Host System
WebDriver
Bindings +
support classes
Browser
Driver
ChromeDriver, eg

How to use

- Install selenium and import it
  - `import { Builder, By, Key, until } from 'selenium-webdriver'`
- Create a WebDriver instance
  - `new Builder().forBrowser('chrome').build()`
- Navigate to a web page
  - `driver.get('https://www.google.com')`

- Locate an HTML element on the web page
    - specific attribute such as name or id (`By.id("loginForm")`)
    - tag name of the element, such as form or button
    - any CSS selector
    - `driver.findElement(By.name('q'))`
- Perform an action on HTML element
    - sendKeys
    - clear
    - submit
    - `driver.findElement(By.name('q')).sendKeys('hello', Key.ENTER)`

- Anticipate the browser response to the action
    - Implicit Waits
    - Explicit Waits
        - until an expected condition occurs on the web page
        - until a maximum wait time elapses
    - `driver.wait(until.elementLocated(By.css('h3'))`
- Conclude the test
    - `driver.quit()`



```javascript
import { Builder, By, Key, until } from 'selenium-webdriver';

describe('web driver', () => {
    let driver

    test("google first result", async () => {
        driver = await new Builder().forBrowser('chrome').build();
        // Navigate to Url
        await driver.get('https://www.google.com');

        // Enter text "hello" and perform keyboard action "Enter"
        await driver.findElement(By.name('q')).sendKeys('hello', Key.ENTER);

        // Wait for h3 element
        const firstResult = await driver.wait(until.elementLocated(By.css('h3')), 10000);

        console.log(await firstResult.getAttribute('textContent'));
    }, 10000)

    afterEach(() => {
        driver.quit();
    })
})
```

- Only used for web-based apps (not for desktop applications)
- Hard to code
- Big community
- No reporting capabilities
- Screenshot `await element.takeScreenshot()`

Cypress

- Javascript e2e testing framework
- 30k github stars
- Good documentation
- Open source
- Parallelization

Test runner

Mocha under hood

```
describe('logged in user', () => {
  beforeEach(() => {
    cy.login()
  })

  afterEach(() => {
    cy.logout()
  })

  it('tests', ...)
  it('more', ...)
  it('things', ...)
})
```

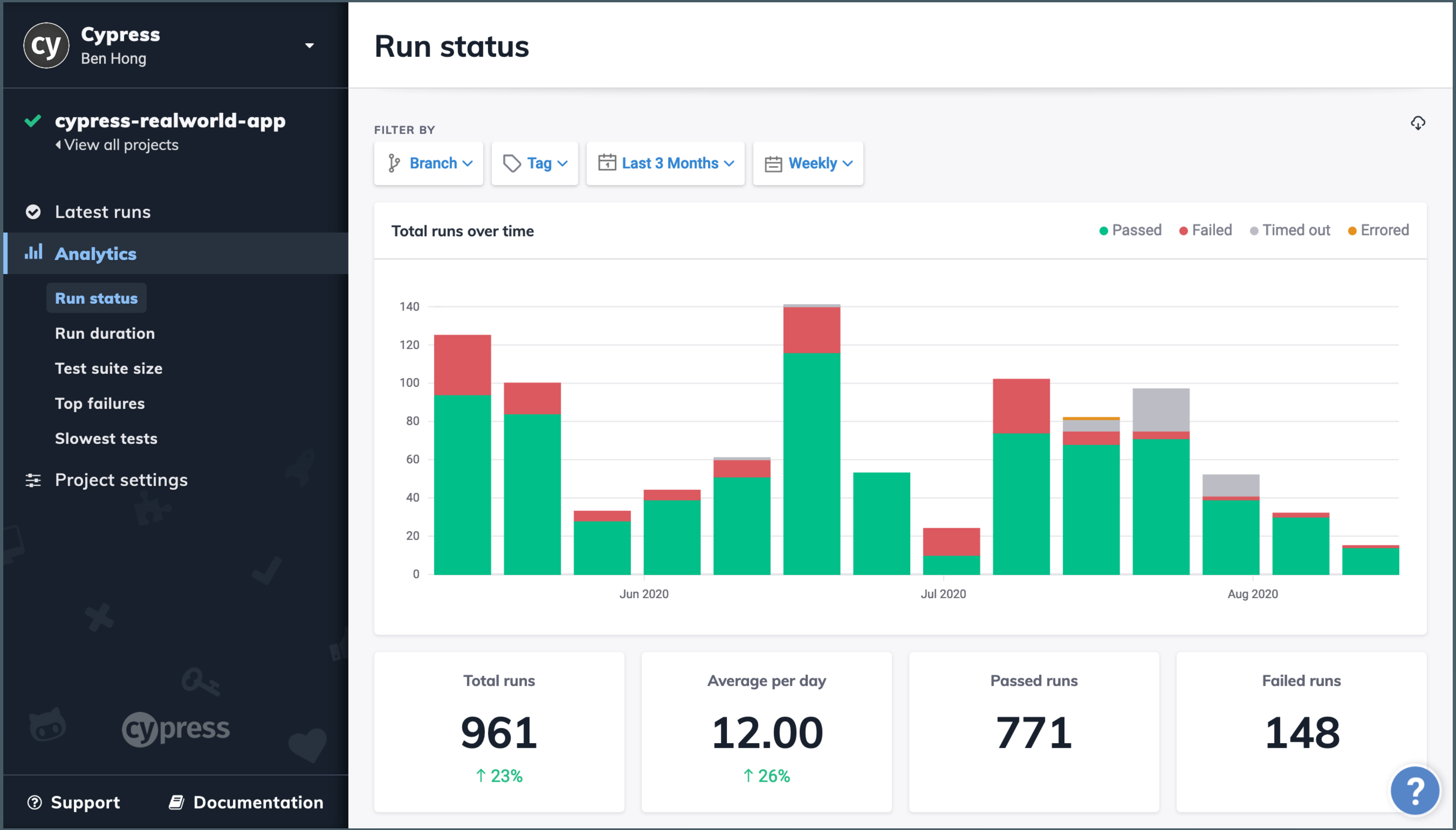
Component Testing library is still in Alpha

```
import * as React from 'react'
import { mount } from '@cypress/react'
import Button from './Button'

it('Button', () => {
  mount(<Button>Test button</Button>)
  cy.get('button').contains('Test button').click()
})
```

Assertions with Chai (Chai-jQuery, Sinon-Chai)

```
cy.get('li.selected').should('have.length', 3);
cy.get('form').find('input').should('not.have.class', 'disabled');
cy.get('textarea').should('have.value', 'foo bar baz');
cy.get(':radio').should('be.checked');
```

Analytics



Run status

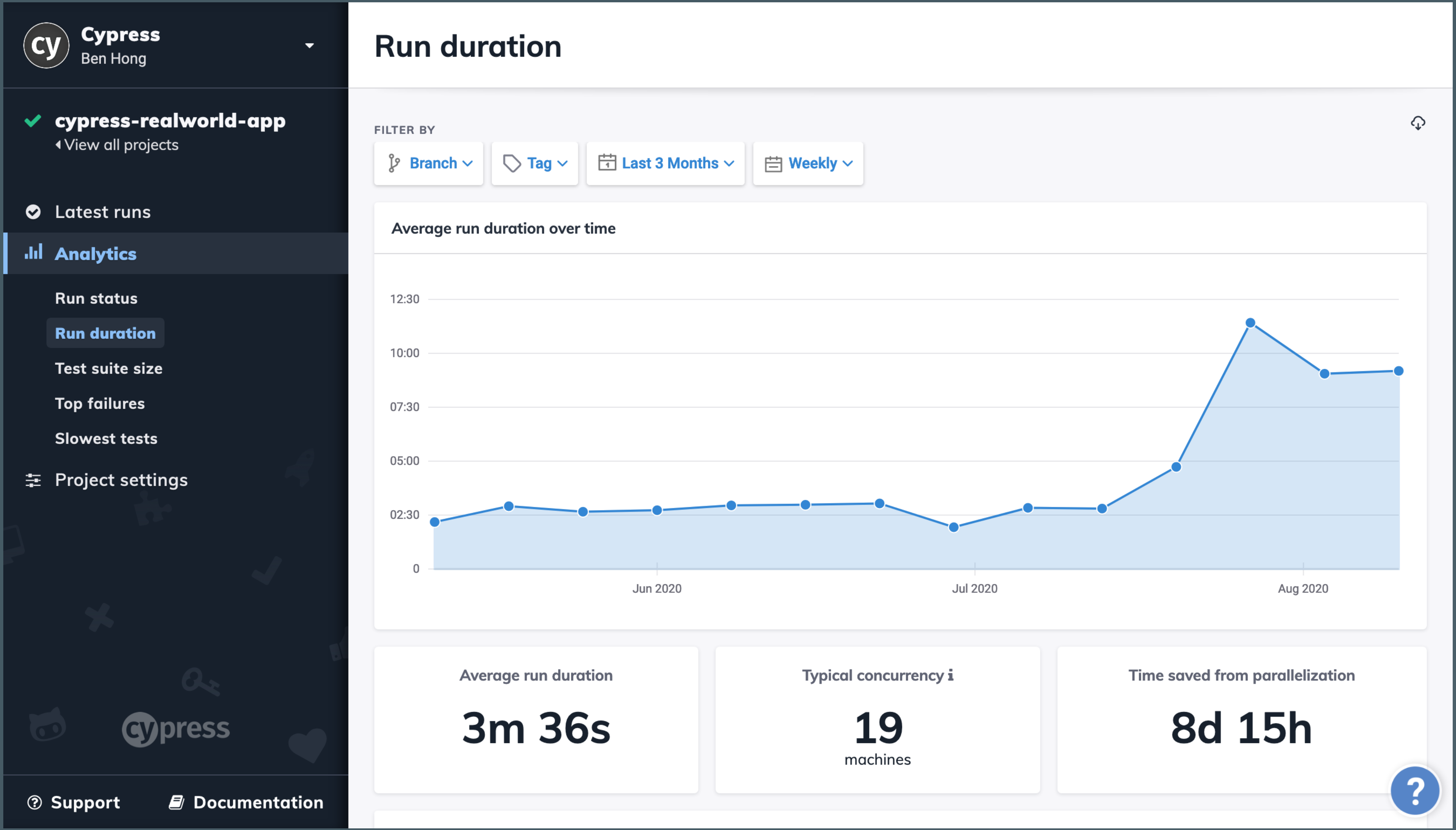
FILTER BY

Branch | Tag | Last 3 Months | Weekly

Total runs over time

● Passed ● Failed ● Timed out ● Errored

Total runs	Average per day	Passed runs	Failed runs
961 ↑ 23%	12.00 ↑ 26%	771	148



Run duration

FILTER BY

Average run duration over time

Average run duration

3m 36s

Typical concurrency i

19
machines

Time saved from parallelization

8d 15h

Spies Stubs and Clocks

```
cy.stub(obj, 'method');

cy.stub(obj, 'method').withArgs('bar').returns('foo');

cy.stub(obj, 'method').resolves('foo'); // promise

cy.stub(obj, 'method').rejects(new Error('foo'));
```

Screenshot

```
cy.screenshot();

cy.screenshot(fileName, options);
```

```
it('completes todo', () => {
  cy.visit('/')
  cy.get('.new-todo').type('write tests{enter}')
  cy.contains('.todo-list li', 'write tests').find('.toggle').check()

  cy.contains('.todo-list li', 'write tests').should('have.class', 'completed')

  // npm i cypress-plugin-snapshots
  cy.get('.todoapp').toMatchImageSnapshot({
    imageConfig: {
      threshold: 0.001,
    },
  })
})
```

```
it('adds todos', () => {

  cy.visit('/')

  cy.get('.new-todo')
    .type('write E2E tests{enter}')
    .type('add API tests as needed{enter}')

  // now confirm the server has 2 todo items
  cy.request('/todos')
    .its('body')
    .should('have.length', 2)
    .and((items) => {
      // ...
    })
})
```

- Firefox and Chrome-family browsers (до 2020 поддерживал только хром)
- Doesn’t use Selenium or WebDriver
- Realtime view progress
- Has native access to every single object (document, window, etc)
- Modify DOM elements directly
- Flaky Test Management ($)

- No multi-tab support
- jQuery-based API
- You have to re-run tests to run in another browser

Playwright

- Node.js library to automate Chromium, Firefox and WebKit with a single API
- Microsoft
- 23k github stars
- Open source
- Chromium, Webkit, and Firefox browsers
- Async / Await

```
const playwright = require('playwright');

(async () => {
  for (const browserType of ['chromium', 'firefox', 'webkit']) {
    const browser = await playwright[browserType].launch();
    const context = await browser.newContext();
    const page = await context.newPage();
    await page.goto('https://mail.ru/');
    await page.click('[data-testid="enter-password"]');
    await page.screenshot({ path: `mail-${browserType}.png` });
    await browser.close();
  }
})();
```

- JS / TS
- Python / Java / C#
- Verbose log
- Need test runner
- Simple to set up

- Scenarios that span multiple page, domains and iframes
- Auto-wait for elements to be ready before executing actions
  - click
  - fill
- Intercept network activity for stubbing and mocking network requests
- Emulate mobile devices, geolocation, permissions
- Support for web components via shadow-piercing selectors
- Native input events for mouse and keyboard
- Upload and download files

Screenshots

```
await page.screenshot({ path: 'screenshot.png' });
await page.screenshot({ path: 'screenshot.png', fullPage: true });

const elementHandle = await page.$('.header');
await elementHandle.screenshot({ path: 'screenshot.png' });
```

Video

```
const context = await browser.newContext({
  recordVideo: {
    dir: 'videos/',
    size: { width: 1024, height: 768 },
  }
});
```

```
const { webkit, devices } = require('playwright');
const iPhone11 = devices['iPhone 11 Pro'];

(async () => {
  const browser = await webkit.launch();
  const context = await browser.newContext({
    ...iPhone11,
    locale: 'en-US',
    geolocation: { longitude: 12.492507, latitude: 41.889938 },
    permissions: ['geolocation']
  });
  const page = await context.newPage();
  await page.goto('https://maps.google.com');
  await page.click('text="Your location"');
  await page.waitForRequest(/.*preview\/pwa/);
  await page.screenshot({ path: 'iphone-11.png' });
  await browser.close();
})();
```

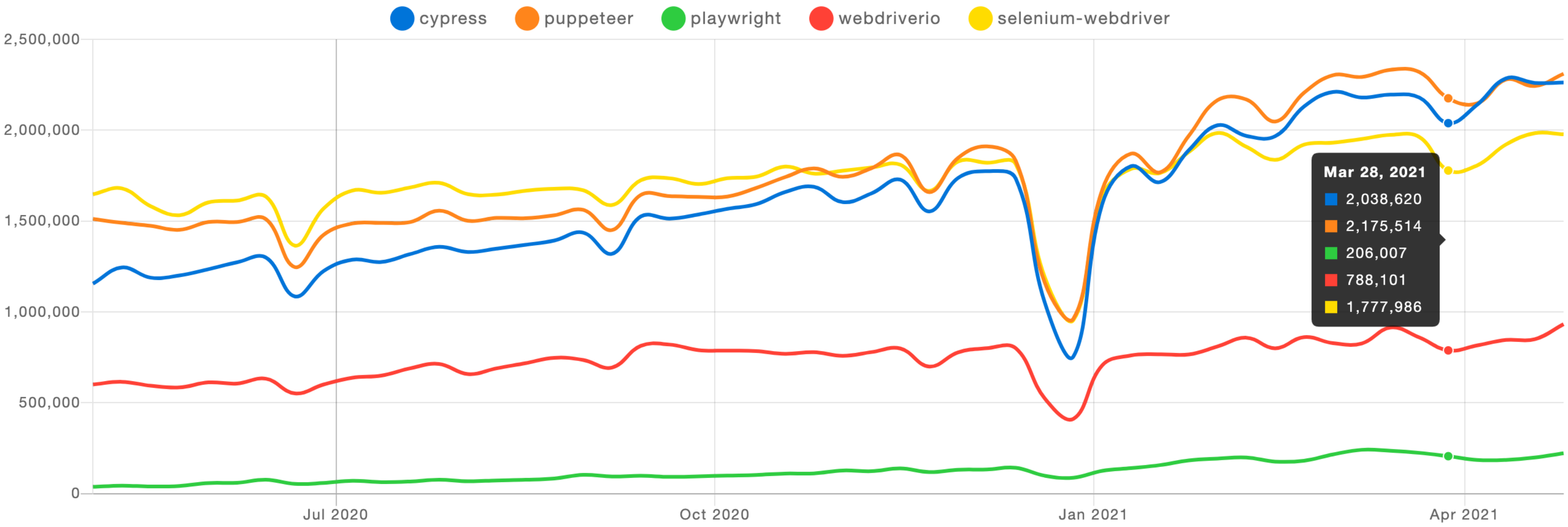
Puppeteer

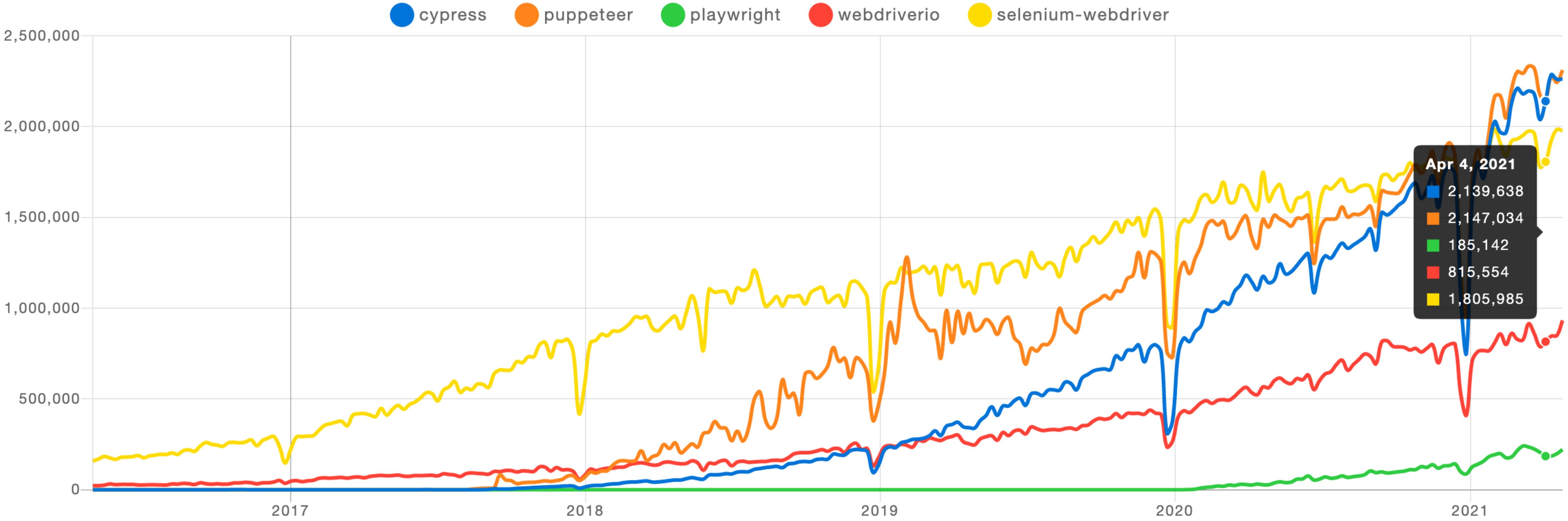
- Node library which provides a high-level API to control Chrome
- Google / Chrome Devtools Team
- 70k github stars
- Good documentation
- Puppeteer focuses on Chromium
- Firefox support is currently experimental
- Generate screenshots and PDFs of pages
- Automate form submission, UI testing, keyboard input, etc
- JS / TS

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://mail.ru');
  await page.click('[data-testid="enter-password"]');
  await page.screenshot({ path: 'example.png' });

  await browser.close();
})();
```

Playwright VS Puppeteer

- Selecting an element by text instead of by a CSS selector
- Working with Shadow DOM
- Waiting for elements to be available automatically
- Solid network validations and network mocking


cypress
puppeteer
playwright
webdriverio
selenium-webdriver
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
Jul 2020
Oct 2020
Jan 2021
Apr 2021
Mar 28, 2021
2,038,620
2,175,514
206,007
788,101
1,777,986


cypress
puppeteer
playwright
webdriverio
selenium-webdriver
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
2017
2018
2019
2020
2021
Apr 4, 2021
2,139,638
2,147,034
185,142
815,554
1,805,985

Домашнее задание

С помощью инструмента Puppeteer протестируйте приложение Simple Blog

```
hexlet program download frontend-testing-react webdrivers
```