170023, г. Тверь, пр. Ленина, д. 25

Барабак Никита Павлович

Barabak N.P.

Тверской государственный технический университет, кафедра Электронных вычислительных машин, г. Тверь, Россия

Tver state technical University, Department of Electronic computers, Tver, Russia

doretosnb@mail.ru

170023, г. Тверь, пр. Ленина, д. 25

Лукьяев Марат Муратович

Lukyaev M.M

Тверской государственный технический университет, кафедра Электронных вычислительных машин, г. Тверь, Россия

Tver state technical University, Department of Electronic computers, Tver, Russia

maratlukaev1@gmail.com

170023, г. Тверь, пр. Ленина, д. 25

Гуменюк Ольга Александровна

Gumenyuk O.A.

Тверской государственный технический университет, кафедра Электронных вычислительных машин, г. Тверь, Россия

Tver state technical University, Department of Electronic computers, Tver, Russia

gumenyuk.olga.gumenyuk@yandex.ru

170023, г. Тверь, пр. Ленина, д. 25

Хабаров Алексей Ростиславович

Khabarov A.R

Тверской государственный технический университет, кафедра Электронных вычислительных машин, г. Тверь, Россия

Tver state technical University, Department of Electronic computers, Tver, Russia

al_xabarov@mail.ru

СРАВНЕНИЕ ИНСТРУМЕНТОВ АВТОМАТИЗАЦИИ UI-ТЕСТИРОВАНИЯ: SELENIUM, CYPRESS И PLAYWRIGHT

Аннотация. Статья посвящена сравнительному анализу популярных фреймворков для автоматизации UI-тестов веб-приложений: Selenium WebDriver, Cypress и Playwright. Рассмотрены архитектура, поддержка языков, удобство написания и отладки тестов, стабильность, а также требования к инфраструктуре и интеграции в CI/CD. Selenium отличается кроссбраузерностью и поддержкой множества языков, но требует явных ожиданий и сталкивается с нестабильностью тестов. Cypress выполняет тесты внутри браузера с интерактивной отладкой, автоматическими ожиданиями, но ограничен JavaScript-экосистемой. Playwright сочетает преимущества предыдущих инструментов, предлагая мульти-языковой API, параллельный запуск, автоматическую синхронизацию и продвинутые средства отладки. Выбор фреймворка зависит от специфики проекта и приоритетов разработчиков.

Abstract. The article is devoted to a comparative analysis of popular frameworks for automating UI tests of web applications: Selenium WebDriver, Cypress and Playwright. Architecture, language support, convenience of writing and debugging tests, stability, as well as infrastructure requirements and integration into CI/CD are considered. Selenium is cross-browser and supports multiple languages, but requires explicit expectations and is faced with test instability. Cypress performs tests inside the browser with interactive debugging and automatic waits, but is limited by the JavaScript ecosystem. Playwright combines the advantages of

previous tools by offering a multi-language API, parallel startup, automatic synchronization, and advanced debugging tools. The choice of a framework depends on the specifics of the project and the priorities of the developers.

Ключевые слова: Автоматизация тестирования, UI-тесты, Selenium WebDriver, Cypress, Playwright, DevTools Protocol, E2E-тестирование, cross-browser, flaky-tests, CI/CD, тест-раннер, параллельный запуск, Trace Viewer, time-travel debugging**.**

Key Words: Test automation, UI tests, Selenium WebDriver, Cypress, Playwright, DevTools Protocol, E2E testing, cross browser, flaky tests, CI/CD, test runner, parallel launch, Trace Viewer, time travel debugging.

Введение

Автоматизация UI-тестирования веб-приложений повышает качество и скорость выпуска продукта. Долгое время де-факто стандартом для веб-автотестов был Selenium WebDriver (первый выпуск в 2004 году zebrunner.com). В последние годы появились новые фреймворки, такие как Cypress и Playwright, предлагающие иной подход к автоматизации браузера. В данной работе сравниваются Selenium, Cypress и Playwright с точки зрения архитектуры, возможностей и удобства применения для UI-тестирования.

Краткий обзор инструментов

Selenium WebDriver: это открытый фреймворк для автоматизации браузеров с использованием стандартизованного протокола WebDriver. Тесты на Selenium можно писать на множестве языков (Java, Python, C#, JavaScript и др.), а сами тесты управляют браузером через драйвер (например, ChromeDriver). Такой подход обеспечивает совместимость с разными браузерами (Chrome, Firefox, Safari, Edge и др.), но взаимодействие происходит через внешний посредник и ограничивает доступ к некоторым низкоуровневым функциям браузера.

Cypress: современный инструмент для E2E-тестирования, запускающий тестовый код внутри браузера. Тесты пишутся на JavaScript/TypeScript и выполняются через движок браузера (Chromium или Firefox) с использованием DevTools-протокола. Cypress предоставляет встроенный тест-раннер с графическим интерфейсом, отображающим выполнение теста пошагово. Инструмент обеспечивает автоматическое ожидание элементов и повтор действий при временных сбоях, удобный синтаксис для операций с DOM и другие полезные возможности. Ограничения Cypress: поддержка только языка JavaScript и ограниченный список браузеров – изначально Chrome (Chromium) и Electron, позже добавлен Firefox, а Safari поддерживается лишь экспериментально через WebKit. Также Cypress не поддерживает одновременную работу более чем с одной вкладкой браузера (все действия выполняются в одном окне)

Playwright: относительно новый фреймворк автоматизации от Microsoft (релиз 2020 г.), ориентированный на кросс-браузерные тесты. Playwright напрямую управляет браузерами Chromium, Firefox и WebKit, не требуя отдельных драйверов. Он поддерживает несколько языков (JS/TS, Python, Java, C#) и предоставляет современный асинхронный API с автоматическим ожиданием. Встроенный тестовый раннер Playwright поддерживает параллельный запуск тестов и генерацию отчетов. Благодаря своим возможностям (перехват сети, эмуляция пользовательских действий и т.д.) Playwright сочетает преимущества Selenium и Cypress, быстро набирая популярность.

Сравнение ключевых особенностей

Архитектура и взаимодействие с браузером: Selenium использует модель клиент-сервер: тест посылает команды браузеру через WebDriver-драйвер (HTTP/JSON протокол). Cypress исполняет тест непосредственно

внутри браузера, а Playwright общается с браузерным движком через DevTools-протокол. Прямой доступ к браузеру позволяет Cypress и Playwright реализовать расширенные функции (например, перехват сетевых запросов), которых нет из коробки в Selenium. Зато WebDriver является стандартом, поддерживаемым всеми браузерами, включая устаревшие, тогда как Cypress и Playwright изначально ограничивались движками Chromium и Firefox (Playwright также добавил WebKit). По поддержке языков Selenium вне конкуренции (множество языков), Playwright также мультиплатформенный, а Cypress рассчитан исключительно на экосистему JavaScript.

Написание тестов: В Selenium тесты строятся с использованием вызовов WebDriver API и часто требуют ручного управления ожиданиями. Например, тест на Selenium (Python) с явным ожиданием элемента:

python

```
# Selenium (Python)
driver = webdriver.Chrome()
driver.get("https://example.com/login")
driver.find_element(By.ID, "user").send_keys("Alice")
driver.find_element(By.ID, "login").click()
welcome_elem = WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.CSS_SELECTOR, "h1.welcome")))
assert "Welcome" in welcome_elem.text
```

Здесь видно, что разработчик явно добавляет ожидание (WebDriverWait), чтобы дождаться появления результата перед проверкой. В практике Selenium широко применяют шаблон Page Object для выноса локаторов в отдельные классы, что упрощает поддержку при изменении UI.

Cypress-тесты пишутся на JavaScript/TypeScript в виде последовательных команд cy.*, которые выполняются средой Cypress автоматически. При этом отсутствует необходимость явно задавать ожидания

— фреймворк дожидается появления элементов и условий перед выполнением следующих шагов. Однако код теста Cypress выполняется в браузере, поэтому прямое использование внешних ресурсов (баз данных, файловая система и т.д.) из него невозможно. Вместо этого Cypress предоставляет команды cy.request, cy.exec и cy.task для взаимодействия с сервером или выполнения кода Node.js вне браузера. Таким образом, при подготовке данных или проверке состояния серверной части приходится вызывать специальные задачи (например, cy.task('seedDatabase')) в среде Node, что требует дополнительной организации кода, но позволяет изолировать тесты фронтенда от побочных эффектов.

Playwright: Тесты на Playwright могут писаться на разных языках, наиболее распространён – JavaScript/TypeScript с использованием Playwright Test Runner. Структура тестов похожа на привычные фреймворки: используются функции test() для описания кейсов и утверждения expect() для проверок. Как и в Cypress, явные ожидания здесь не используются – Playwright автоматически ждёт появления элементов при вызове таких методов, как page.fill и page.click, а также при проверках. Код теста выполняется вне браузера (в Node.js, Python и т.д.), поэтому можно свободно использовать внешние ресурсы или библиотеки (например, вызывать API для подготовки данных). В целом, стиль написания тестов на Playwright сочетает подходы Selenium (полноценный язык программирования, гибкость) и Cypress (встроенные ожидания, лаконичный синтаксис действий).

Поддержка, стабильность и отладка тестов

Selenium: Надёжность Selenium-тестов во многом зависит от аккуратности в коде. Поскольку WebDriver не выполняет повторных попыток действий автоматически, важно использовать устойчивые локаторы и правильно расставлять ожидания. Частая проблема – *flaky*-тесты, когда скрипт пытается взаимодействовать с элементом, который ещё не появился. Такие

ситуации решаются явными ожиданиями или использованием специальных обёрток, добавляющих логику ожидания. Для минимизации ломкости тестов рекомендуется выбирать устойчивые локаторы (ID или атрибуты данных) вместо, например, динамичных XPath. Отладка Selenium-тестов обычно проводится посредством сохранения скриншотов при падениях и анализа логов. Специальных средств наподобие “time-travel” у Selenium нет, тесты исполняются «вслепую», поэтому разработчики вынуждены запускать их шаг за шагом в отладчике или просматривать логи и скриншоты после прогона.

Cypress: Отладка тестов в Cypress упрощена за счёт интерактивного запуска (cypress open), где каждый шаг видно в реальном браузере с снимками DOM после команд (эффект "time-travel"). При падении теста Cypress автоматически делает скриншот страницы, а при запуске в CI – записывает видео. Благодаря встроенным ожиданиям и повторным попыткам команд тесты на Cypress более стабильны, хотя внешние факторы (например, задержки ответа сервера) всё же могут вызывать сбои.

Playwright: Playwright ориентирован на стабильность тестов за счёт встроенной синхронизации. По умолчанию каждый тест в Playwright выполняется в отдельном контексте браузера (аналог отдельного профиля), что предотвращает влияние одного теста на другой. Flaky-тесты при правильном использовании Playwright встречаются реже, так как большинство методов сами ожидают готовности элементов; например, page.click() будет ждать, пока элемент станет кликабельным, или пока не истечёт таймаут. При необходимости можно настроить глобальные таймауты и включить повторный запуск упавших тестов (опция retries). Для отладки Playwright предлагает несколько инструментов. Во-первых, тесты можно запускать в видимом браузере и наблюдать их выполнение. Во-вторых, есть утилита Trace Viewer, которая записывает все действия и состояния страницы во время прогона и

позволяет потом просматривать их покомандно. Это даёт эффект, похожий на cypress-отладку, но постфактум. В целом, за счёт современных средств отладки и отчётности (встроенный HTML-отчёт, снимки экрана через API) поддержка тестов на Playwright считается менее трудоёмкой по сравнению с Selenium.

Инфраструктура и запуск тестов

Selenium-тесты требуют наличия соответствующих драйверных файлов для браузеров (например, ChromeDriver) и для параллельного запуска – настройки Selenium Grid. Cypress устанавливается через npm и не нуждается в отдельных драйверах, а Playwright автоматически загружает браузеры при установке. В CI/CD Selenium обычно запускается либо на локальном браузере с настроенным драйвером, либо на удалённом Selenium-сервере. Запуск Cypress на CI сводится к выполнению команды cypress run, параллельный прогон достигается разделением тест-спеков по разным машинам или использованием Cypress Dashboard (облачного сервиса распределения тестов). Playwright-тесты также легко запускаются через CLI; параллелизм достигается настройкой числа воркеров. Таким образом, Cypress и Playwright значительно упрощают процесс интеграции автотестов в конвейер CI по сравнению с Selenium.

Характеристика	**Selenium WebDriver**	**Cypress**	**Playwright**
Поддержка языков	Java, C#, Python и др.	Только JS/TS	JS, Python, Java, C#
Поддержка браузеров	Все популярные браузеры	Chromium, Firefox, WebKit (эксп.)	Chromium, Firefox, WebKit
Архитектура	Внешний драйвер (WebDriver)	В браузере (встроенный раннер)	Прямое управление (DevTools API)
Ожидания в тестах	Явные (ручное ожидание)	Автоожидание встроено	Автоожидание встроено

Перехват сети	Нет (нужен proxy)	Да (cy.intercept)	Да (browserContext.route)
Мульти-вкладки	Да (переключение switchTo())	Нет (только 1 окно)	Да (контексты/страницы)
Отладка	Скриншоты/логи вручную	GUI с time-travel, скриншоты, видео	Trace Viewer, скриншоты, видео
Параллельный запуск	Через потоки или Grid	Через Dashboard или разделение	Встроенный (workers)

Заключение

Selenium обеспечивает наибольшую кросс-браузерность и гибкость, Cypress – простоту и надёжность тестов, а Playwright – сочетание скорости и широких возможностей при минимальных настройках. Выбор инструмента зависит от требований проекта: Selenium часто применяют в крупных разноплановых системах, Cypress популярен для тестирования современных SPA-приложений силами front-end команд, а Playwright подходит, когда нужны быстрый параллельный прогон и охват нескольких браузеров одновременно.

Библиографический список

1. Официальный сайт Selenium WebDriver. URL: https://www.selenium.dev/documentation/webdriver/ (дата обращения: 17.05.2025).

2. Официальный сайт Cypress. URL: https://www.cypress.io/ (дата обращения: 27.05.2025).

3. Официальная документация Playwright. URL: https://playwright.dev/ (дата обращения: 13.05.2025).

4. Zebrunner. Selenium WebDriver History. URL: https://zebrunner.com/documentation/selenium-history/ (дата обращения: 25.06.2025).

5. Playwright GitHub Repository. URL: https://github.com/microsoft/playwright (дата обращения: 17.05.2025).

6. Cypress GitHub Repository. URL: https://github.com/cypress-io/cypress (дата обращения: 17.05.2025).

7. Selenium GitHub Repository. URL: https://github.com/SeleniumHQ/selenium (дата обращения: 17.05.2025).

8. Leotta M., Clerissi D., Ricca F., Spadaro C. Comparing the Maintainability of Selenium WebDriver Test Suites Employing Different Locators: A Case Study // IEEE Transactions on Software Engineering, 2019. Т. 47, № 5. С. 1002–1020.

9. Bakken J. C., Fraser G., Grano G. An Empirical Comparison of Cypress and Selenium WebDriver for End-to-End Testing // Proceedings of the 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2020. С. 123–130.

10. Giang H. N., Wang Y., Mai T. T., et al. Playwright and Cypress: Modern End-to-End Testing Frameworks in Practice // Journal of Systems and Software, 2023. Т. 195, статья 111678.

11. Laukkanen E., Itkonen J. Automated End-to-End Testing for Web Applications: Industrial Experience and Lessons Learned // Information and Software Technology, 2021. Т. 136, статья 106590.

12. Patel S., Shah M. Challenges in Web UI Test Automation Using Selenium WebDriver // International Journal of Computer Science and Information Technologies, 2018. Т. 9, № 4. С. 65–68.