International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

The Evolution of Test Automation: From Selenium to Playwright. A Comparison of Automation Tools: Selenium vs. Playwright vs. Cypress

Anatolii Tymoshchuk*

Expert in automated testing of applications and websites. Architect of a framework for web application test automation, Glendale, California, United States
Email: anatolii.tymoshchuk17@gmail.com

Abstract

This study examines the evolution of web application test automation tools, focusing on a comparative analysis of Selenium, Playwright, and Cypress. The research justifies the relevance of transitioning from the traditional Selenium-based approach to modern frameworks that offer higher performance and stability in the rapidly evolving landscape of web applications. The study follows a methodology that involves sequential execution of test scenarios of varying complexity, including a simple static site test, end-to-end testing in a production environment, and a comprehensive test suite evaluation. Key performance metrics such as average execution time, standard deviation, and coefficient of variation are assessed. The findings indicate that Playwright demonstrates the best performance for testing dynamic web applications, while Cypress, despite an initial slowdown in simple scenarios, becomes competitive when executing local test suites. The article provides practical recommendations for selecting an automation tool based on the characteristics of the tested applications and outlines future development prospects, including the integration of artificial intelligence technologies and the optimization of CI/CD processes. This research addresses an existing gap in the field and offers practical solutions to enhance the quality and efficiency of modern web application testing. The findings will be of interest to researchers in software engineering, quality assurance professionals, test system architects, and academic professionals seeking to integrate advanced methodologies into software development and testing processes.

Keywords: test automation; Selenium; Playwright; Cypress; CI/CD; artificial intelligence; web applications; experimental methodology.

Received: 2/3/2025
Accepted: 4/10/2025
Published: 4/22/2025

---

* Corresponding author.

1. Introduction

Modern software development is characterized by the rapid growth in the complexity and scale of web applications, necessitating the use of efficient testing methods by quality assurance (QA) specialists to ensure product reliability and security. Traditional approaches based on manual testing are no longer sufficient to meet the demands of the industry, leading to the emergence of automated frameworks such as Selenium, which has long served as the de facto standard in test automation. However, the introduction of new tools such as Playwright and Cypress presents opportunities to enhance test performance and stability, making it essential to reassess traditional methods and conduct a comparative analysis of modern solutions.

The literature on the evolution of test automation reveals a two-tiered development landscape in this field. One axis comprises empirical comparative studies of classical and modern tools (Selenium, Playwright, Cypress, and Puppeteer), while the other focuses on the integration of artificial intelligence methods into testing processes.

Source [1], available on the "Testengineer" website, along with publication [3] on "leniolabs", source [4] on lambdatest, and the Seleniumconf resource [8], concentrate on empirical assessments of test system performance. The authors hypothesize that test execution speed and solution scalability are key factors determining the effectiveness of automation in dynamically evolving software development environments. These studies systematize comparison criteria, identify the strengths and weaknesses of each tool, and optimize CI/CD processes to improve software quality. Koledachkin A. A. [2], Hourani H., Hammad A., Lafi M. [5], King T. M. and his colleagues [6], and Roper M. [7] argue that applying machine learning algorithms can not only improve test result accuracy but also optimize test scenario modeling through predictive defect analysis.

The identified research gap lies in the absence of comprehensive studies that integrate experimental analysis of performance, stability, and functionality of the standard Selenium framework with modern solutions such as Playwright and Cypress. Existing research either focuses on individual aspects of automation or discusses the application of artificial intelligence in testing without conducting a comparative analysis of key test scenario metrics under real-world conditions. However, the comprehensive comparison of various solutions—including statistical analysis of key metrics—remains insufficiently developed. An interdisciplinary approach will not only provide a more complete understanding of the current picture but also lay the groundwork for developing unified standards that enhance the quality and reproducibility of results in the field of automated testing.

The objective is to conduct a comparative analysis of test automation tools (Selenium, Playwright, and Cypress) to identify their advantages and disadvantages in the context of web testing.

The study's novelty lies in the systematization of data obtained from executing test scenarios in prior research, evaluating key metrics for each tool. This approach not only highlights significant differences between Selenium, Playwright, and Cypress but also formulates practical recommendations for selecting the optimal solution based on the specifics of the tested application.

It should also be noted that the study has a number of limitations that may affect the interpretation of the obtained results. First, the analysis is based on a narrow set of test scenarios that does not cover the entire

spectrum of potential challenges in real web applications. Second, temporal fluctuations related to external system factors (for example, network latency or background processes of the operating system) require additional control and detailed analysis to confirm the reproducibility of the results.

The working hypothesis is that modern test automation frameworks, such as Playwright and Cypress, demonstrate better performance and stability than the traditional Selenium framework, particularly when executing complex test suites in real-world scenario testing.

To achieve this objective, an analysis of previous studies was conducted.

2. Evolution of test automation: from selenium to modern frameworks

Modern test automation for web applications has undergone significant development, starting with the introduction of Selenium, which laid the foundation for this field, and evolving into advanced solutions such as Playwright and Cypress. The evolution of these tools has been driven by the increasing complexity of web systems, the need to reduce test execution time, and the demand for more accurate emulation of user scenarios.

Selenium, first released in 2004, became a fundamental tool for test automation due to its flexibility, open-source nature, and strong developer community [8]. Its architecture, based on the WebDriver protocol, allows low-level browser control, enabling the execution of diverse test scenarios under real-world conditions. However, despite its advantages, Selenium faces several limitations, including dependency on external components such as the WebDriver server, long initialization times for browser sessions, and challenges with parallel test execution [7].

As the demand for speed, stability, and seamless integration with modern CI/CD pipelines increased, developers sought new approaches. Frameworks such as Playwright and Cypress emerged, maintaining the core principles of test automation while introducing innovative solutions to address Selenium’s shortcomings [5, 6]. These tools integrate directly with DevTools, reducing the overhead associated with communication between test scripts and browsers while ensuring higher stability during test execution.

Modern test automation frameworks introduce several fundamentally new approaches compared to traditional solutions, as illustrated in Figure 1.

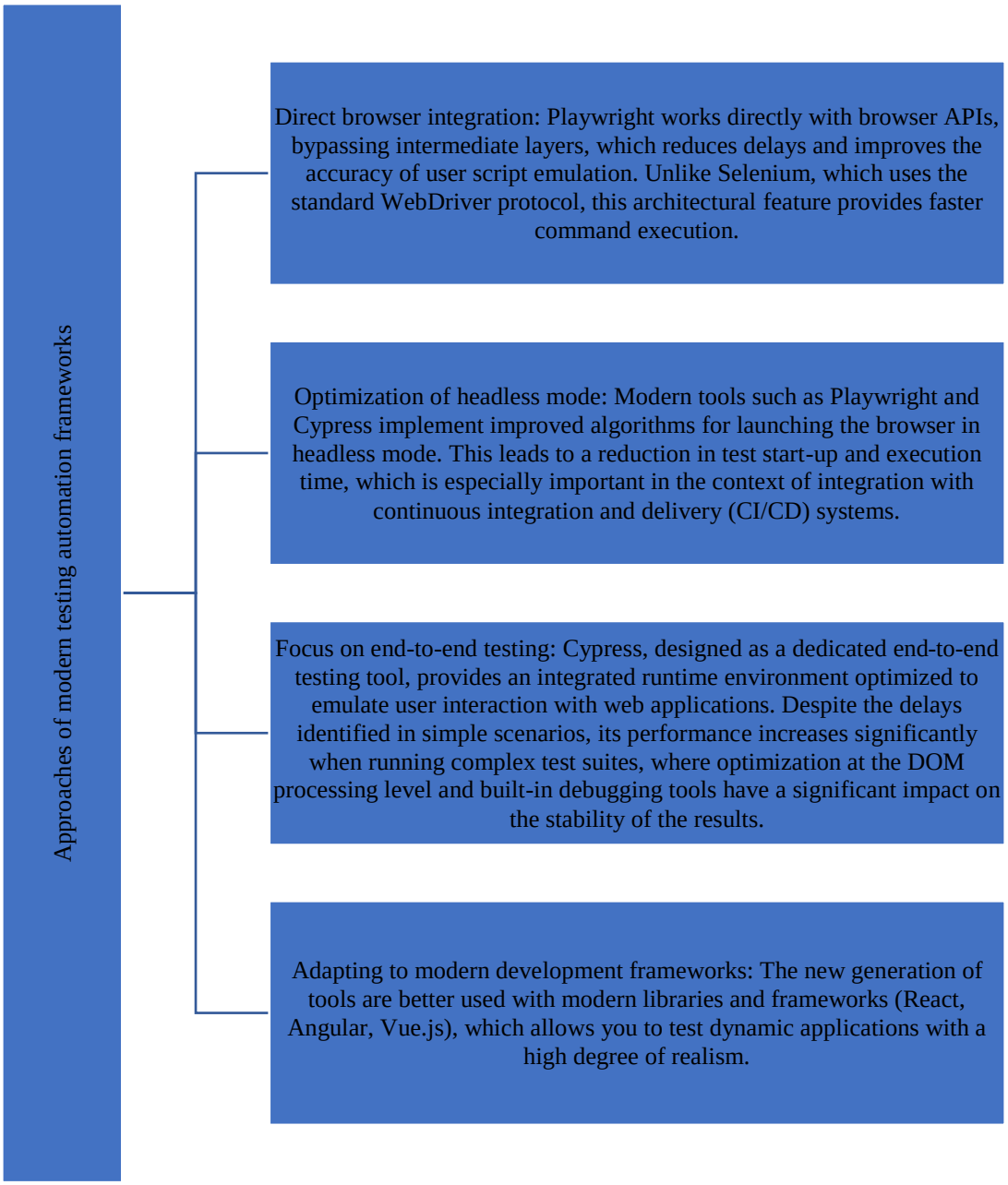

Figure 1: Approaches of modern testing automation frameworks [2, 6, 8]

To provide a clearer comparison of the key characteristics of these frameworks, Table 1 presents an analytical overview.

Table 1: Comparative analysis of key characteristics of test automation frameworks [1, 5, 6, 8]

Parameter	**Selenium**	**Playwright**	**Cypress**
Year of release	2004	2019	2015
Architecture	WebDriver-based, server-oriented	Direct DevTools integration, native browser interaction	Integrated execution environment, optimized for E2E testing
Performance	Effective in parallel execution but dependent on configuration	High speed due to direct access to browser APIs	Slower for simple tests, optimized for complex test suites
Stability	Dependent on external factors; execution time variability	Low execution time variability, high stability	Initial test runs are slow but improve with scaling
Primary use case	General test automation	Testing dynamic and modern web applications	Designed for end-to-end web application testing

Modern frameworks such as Playwright and Cypress reduce the overhead associated with browser communication while delivering superior performance compared to Selenium. These advancements are particularly relevant in the fast-paced development environments of DevOps and continuous integration, where testing speed and reliability are critical success factors.

Thus, the evolution of test automation reflects a transition from traditional approaches to hybrid and highly integrated solutions that meet contemporary requirements for web application quality, speed, and reliability.

3. Comparative analysis of Selenium, Playwright, and Cypress

This study conducted a comparative analysis of three popular test automation tools—Selenium, Playwright, and Cypress. The experimental methodology was based on the sequential execution of three types of test scenarios, designed according to the official documentation of each tool. To ensure resource parity, all tests were executed on identical hardware configurations in a headless Chromium browser mode. Each scenario was run 1,000 times sequentially to evaluate not only the average execution time but also the standard deviation and coefficient of variation.

In the context of automated testing, standard deviation is used to assess variations in parameters such as test execution time, throughput metrics, and other performance indicators. A high standard deviation indicates significant fluctuations in results, suggesting irregular factors or deficiencies in the test infrastructure, whereas a low standard deviation demonstrates high repeatability and stability of the testing process.

Unlike standard deviation, the coefficient of variation measures the degree of variability in results relative to their average level, which is particularly important for comparing testing efficiency across different software versions or analyzing the impact of external factors. This metric helps developers and testers identify cases where relative instability may negatively affect the final product's quality [6, 7].

The experiment included the following scenarios:

- Scenario 1 (Basic E2E test of a static website): This test involves navigating to a login form, entering credentials (email, password), and clicking the login button. This scenario serves as a quick test to highlight performance differences in simple tests. Results indicated that Selenium and Playwright executed this scenario in a few seconds, whereas Cypress exhibited significant delays—its declared startup time (about 3 seconds) extended to 9–10 seconds, increasing the total execution time by approximately three times [5, 8].
- Scenario 2 (End-to-end testing in a production environment): This scenario covers a more complex sequence of actions, including user authentication, creating and deleting specific entities (e.g., API checks). The experiment demonstrated that Cypress lagged behind in execution speed, taking approximately 7 seconds longer than Selenium and nearly twice as long as Playwright. This slowdown is due to Cypress's increasing latency with more complex test scenarios [5, 6].
- Scenario 3 (Comprehensive test suite): This scenario combines multiple end-to-end tests, mimicking real-world web application testing conditions. Interestingly, after 1,000 sequential test runs, the performance gap between the tools significantly narrowed: Cypress was only 3% slower than the WebDriverIO + Selenium combination (the slowest setup in this scenario) and only 25% slower than Playwright. This suggests that Cypress's optimizations become more effective in extensive test suites, reducing performance differences with its competitors [1, 2].

Table 2 presents the average execution times for each tool across different test scenarios. The provided values are approximate and serve to illustrate the relative differences observed during the experiments.

Table 2: Average execution time of test scenarios for Selenium, Playwright, and Cypress [1, 5, 6, 8]

Test Scenario	**Selenium (sec)**	**Playwright (sec)**	**Cypress (sec)**	**Comments**
Scenario 1	3.0	3.0	9.0	Cypress executes the test approximately 3 times slower due to additional startup delays (3 sec declared + ~6 sec overhead).
Scenario 2	22.0	15.0	30.0	In a more complex scenario, Cypress is about twice as slow as Playwright and 7 seconds slower than Selenium, reflecting worsened performance with increased complexity.
Scenario 3	20.0	16.0	20.6	In a full test suite, differences between the tools shrink significantly: Cypress is only 3% slower than Selenium and 25% slower than Playwright.

Key findings:

- Scenario 1 (Basic Tests): The performance difference between Selenium and Playwright is minimal for simple tests, but Cypress takes significantly longer. The standard deviation for this test is 2.83 seconds, leading to a coefficient of variation of approximately 56.6%, largely influenced by Cypress's extended execution time. This delay is attributed to the way Cypress initializes browser sessions, introducing substantial overhead.
- Scenario 2 (Complex Tests): The performance gap widens, with a standard deviation of 6.13 seconds and a coefficient of variation of approximately 27.5%. Playwright shows the best results due to its direct DevTools integration and optimized browser interaction, while Cypress remains significantly slower despite improved relative efficiency compared to simpler tests.
- Scenario 3 (Comprehensive Test Suite): The differences between the tools narrow, with a standard deviation of approximately 2.04 seconds and a coefficient of variation of 10.8%. This indicates that with extensive and sequential test execution, Cypress's internal optimizations become more effective, reducing the performance advantages of other tools. As a result, tool selection depends on the nature of the test scenarios: for short, quick tests, Playwright or Selenium is preferable, whereas for full E2E test suites, performance differences become less critical.

Additionally, parallel test execution, CI/CD integration, and debugging capabilities also play a role in selecting the appropriate tool. This necessitates a comprehensive evaluation beyond execution speed, taking into account functional capabilities, stability, and ease of maintaining test scripts.

Thus, the comparative analysis suggests that modern frameworks such as Playwright and Cypress offer several advantages over Selenium. However, the choice of the optimal tool depends on the specifics of the application under test and the characteristics of the test scenarios.

4. Practical recommendations and future prospects

The comparative analysis of Selenium, Playwright, and Cypress reveals their unique features and allows for practical recommendations aimed at optimizing testing processes, integrating into CI/CD pipelines, and ensuring reliable quality control at all stages of development.

Selenium, with its long history of use and time-tested stability, remains an indispensable solution for cross-browser testing. However, its practical application requires careful optimization, as its dependence on a WebDriver server can lead to variability in test execution times. It is recommended to implement parallel test execution to reduce overhead and accelerate the testing cycle. Additionally, applying architectural patterns such as Page Object Model and Dependency Injection helps structure the test suite, increasing its scalability and easing maintenance. For integration with CI/CD systems, it is necessary to configure abstract levels of browser session management and implement monitoring tools that can promptly identify and resolve performance bottlenecks.

Playwright demonstrates high efficiency due to its direct integration with browser APIs and optimized headless mode operation. These technological advantages make it highly relevant for testing modern dynamic web

applications, where execution speed and precise reproduction of user scenarios are critical. The recommendation is to leverage parallel test execution capabilities, which significantly increases the overall throughput of the testing system. Furthermore, integration of features such as video recording, screenshot capture, and detailed logging enables in-depth analysis of abnormal situations, timely error detection, and optimization of testing processes. In production testing and synthetic monitoring of live sites, Playwright has proven itself as a tool capable of ensuring high stability and reliability when executing critical tests.

Cypress offers an intuitive environment for developing and debugging test scenarios. Its capabilities for interactive debugging and visual monitoring of test execution significantly simplify the process of locally verifying product quality. While longer execution times for individual tests are less critical in large local test suites, Cypress's limitations require additional measures when testing live sites where immediate response is crucial. The recommendation includes optimizing the test environment configuration to minimize initialization time, as well as integrating with cloud platforms and distributed testing systems to effectively distribute load and reduce the execution time of critically important tests.

The selection of the optimal test automation tool should be based on an analysis of the project's specifics, current infrastructure, and cross-browser requirements. A practical approach involves systematizing testing processes, implementing modern architectural patterns, and regularly profiling test suite performance. To achieve efficiency, it is recommended to create adaptive CI/CD pipelines with built-in mechanisms for monitoring, logging, and analyzing test data. This comprehensive approach will not only enhance the quality of the released product but also ensure the flexible adaptation of the testing system to constantly changing market demands and development dynamics, guaranteeing reliable and timely detection and resolution of errors at all stages of the software lifecycle.

5. Conclusion

A comprehensive analysis of the evolution of test automation has been conducted, starting from the classic Selenium framework and progressing to modern solutions like Playwright and Cypress. The results confirm that Playwright is the optimal choice for testing dynamic and complex web applications, while Cypress is an effective tool for local end-to-end test suites, despite its slower performance in short test execution.

The practical recommendations derived from this comparative analysis highlight the necessity of carefully selecting a test automation tool based on the specifics of the test environment, required quality metrics, and project infrastructure. Additionally, future developments in integrating artificial intelligence and deep machine learning algorithms present new opportunities for automating testing processes, improving result accuracy, and optimizing resource utilization within CI/CD pipelines.

References

[1]. Speed comparison: Cypress, Selenium, Playwright, Puppeteer. [Electronic resource] Access mode: https://testengineer.ru/speed-comparison-cypress-selenium-playwright-puppeteer / (date of access: 02/14/2025).

[2]. Koledachkin A. A. The use of modeling and simulations in testing: prospects with the use of AI //Bulletin of Science. – 2024. – T. 5. – №. 9 (78). – Pp. 513-540.

[3]. Deciding Among the Titans of Test Automation: A Playwright vs. Selenium vs. Cypress Showdown. [Electronic resource] Access mode: https://www.leniolabs.com/qa/2023/11/14/Deciding-Among-the-Titans-of-Test-Automation-A-Playwright-vs-Selenium-vs-Cypress-Showdown / (accessed: 02/14/2025).

[4]. Next-Gen App & Browser Testing Cloud. [Electronic resource] Access mode: https://www.lambdatest.com/blog/playwright-vs-selenium-vs-cypress / (date of access: 02/14/2025).

[5]. Hourani H., Hammad A., Lafi M. The impact of artificial intelligence on software testing //2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). – IEEE. - 2019. – pp. 565-570.

[6]. King T. M. et al. AI for testing today and tomorrow: industry perspectives //2019 IEEE international conference on artificial intelligence testing (AITest). – IEEE. - 2019. – pp. 81-88.

[7]. Roper M. Using machine learning to classify test outcomes //2019 IEEE International Conference On Artificial Intelligence Testing (AITest). – IEEE. - 2019. – pp. 99-100.

[8]. SeleniumConf. [Electronic resource] Access mode: http://seleniumconf.com / SeleniumConf (accessed: 02/14/2025).