

# Playwright

---

Let us play it - right

# Einführung

Wissenswertes  
zu Playwright

# Playwright

**Playwright** enables reliable end-to-end testing  
for modern web apps.

GET STARTED

 Star 67k+

# stateofjs

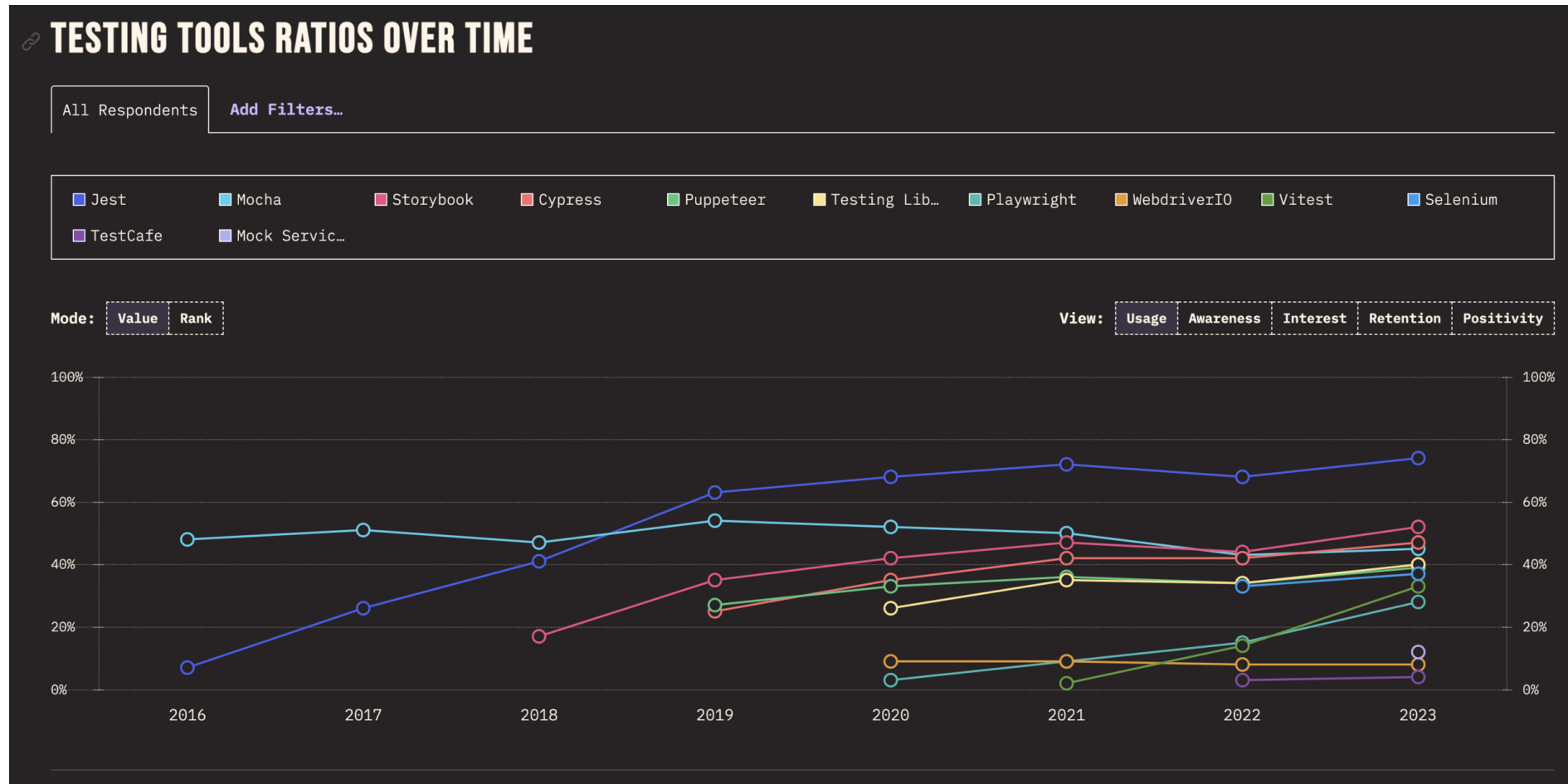
---

Um sich über die aktuellen Trends in der JavaScript Welt zu informieren, ist "stateofjs" eine wunderbare Quelle.

Die Resultate hängen immer ein Jahr zurück, sprich 2024.stateofjs.com wird es erst 2025 geben.

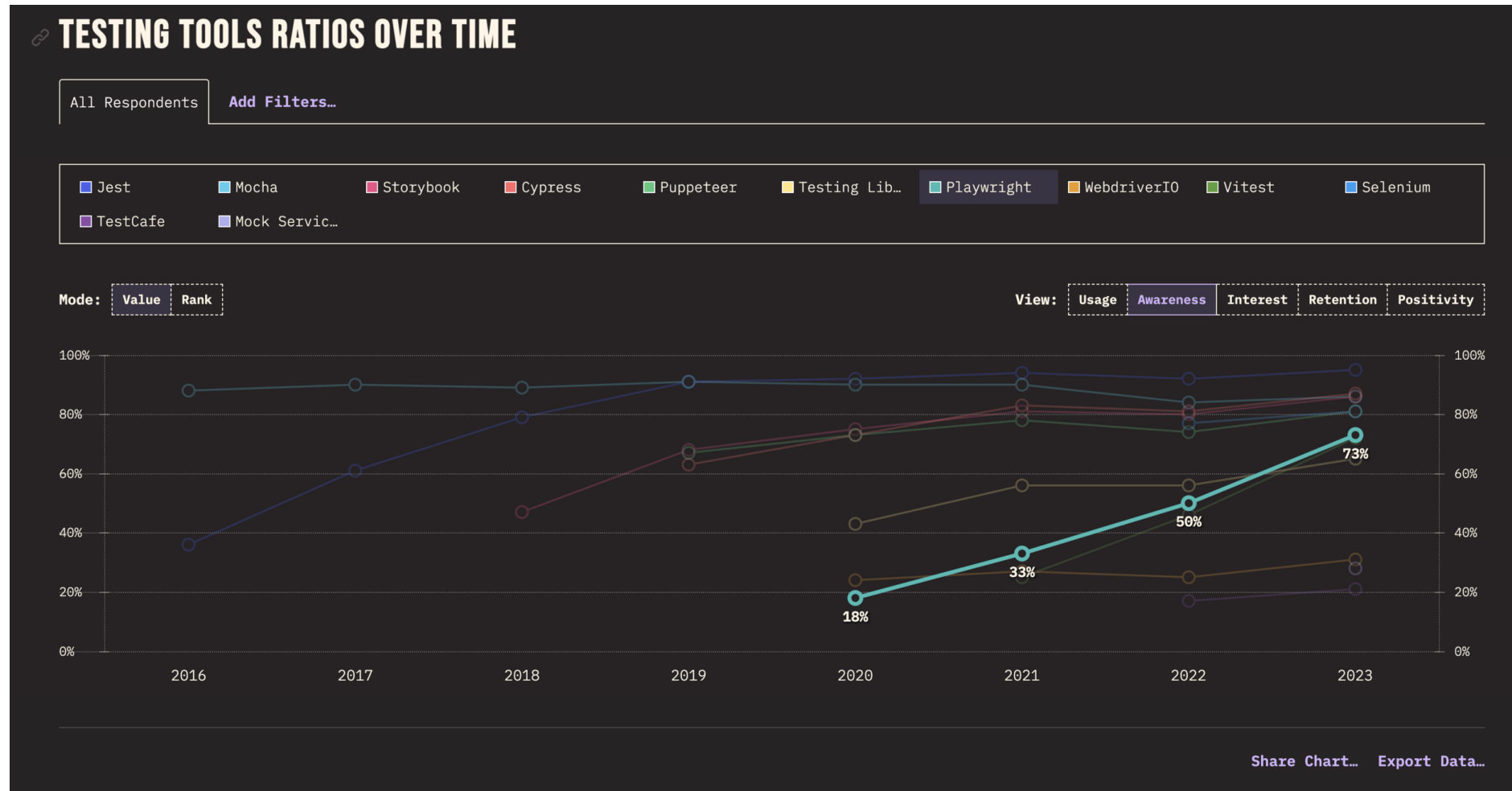
<https://2023.stateofjs.com/en-US/libraries/testing/>

# stateofjs - usage (general)



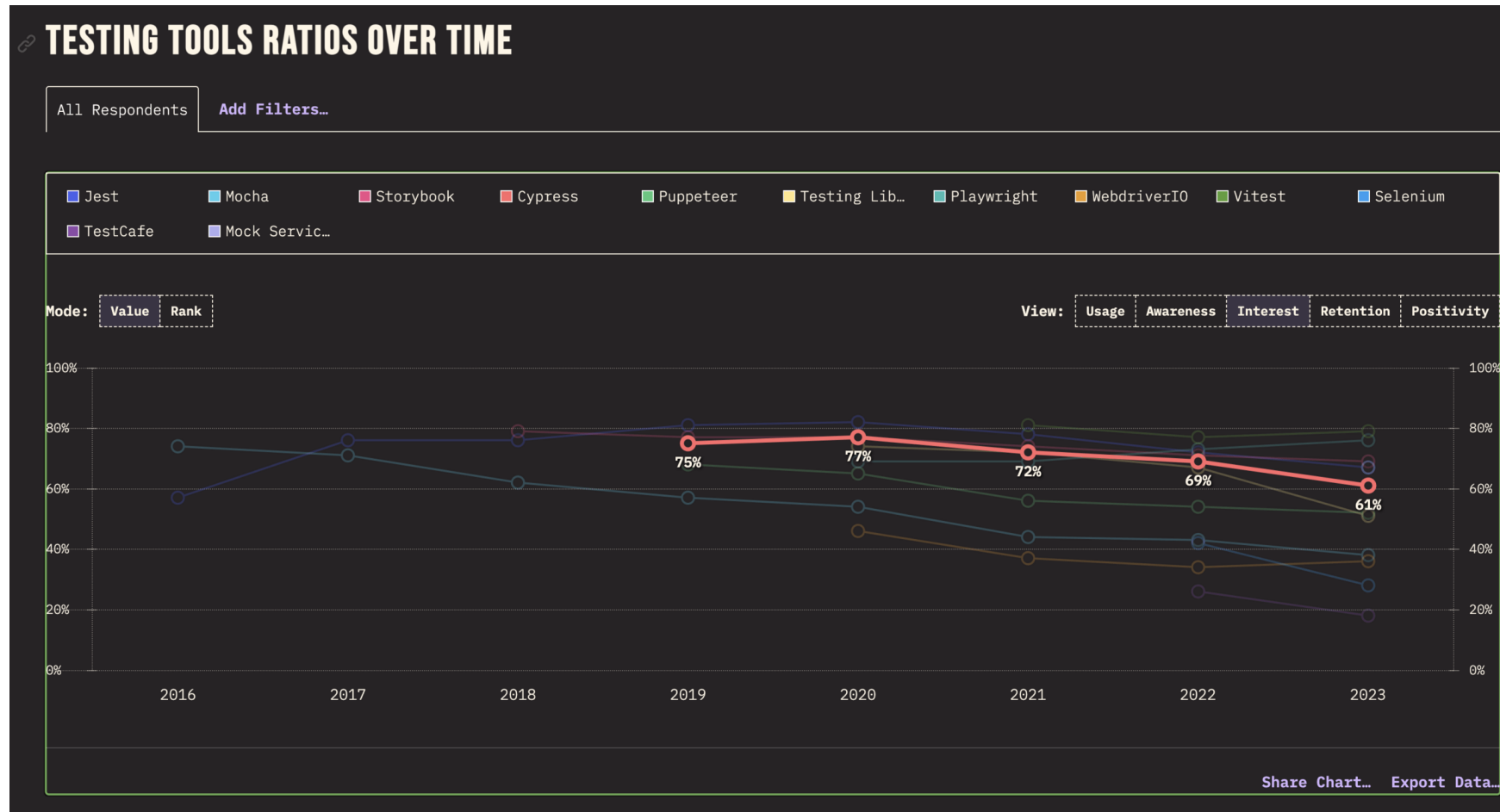
<https://2023.stateofjs.com/en-US/libraries/testing/>

# stateofjs - awareness playwright



<https://2023.stateofjs.com/en-US/libraries/testing/>

# stateofjs - interest in cypress



<https://2023.stateofjs.com/en-US/libraries/testing/>

# Playwright

Was ist die  
Geschichte  
hinter  
Playwright?



# Was ist Playwright

**2004**  
**Selenium**  
First release



**2020**  
**Playwright**

**2017**  
**Cypress**  
Public beta

# Playwright

## **Any browser • Any platform • One API**

**Cross-browser.** Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox.

**Cross-platform.** Test on Windows, Linux, and macOS, locally or on CI, headless or headed.

**Cross-language.** Use the Playwright API in [TypeScript](#), [JavaScript](#), [Python](#), [.NET](#), [Java](#).

**Test Mobile Web.** Native mobile emulation of Google Chrome for Android and Mobile Safari. The same rendering engine works on your Desktop and in the Cloud.

# Playwright

## **Resilient • No flaky tests**

**Auto-wait.** Playwright waits for elements to be actionable prior to performing actions. It also has a rich set of introspection events. The combination of the two eliminates the need for artificial timeouts - the primary cause of flaky tests.

**Web-first assertions.** Playwright assertions are created specifically for the dynamic web. Checks are automatically retried until the necessary conditions are met.

**Tracing.** Configure test retry strategy, capture execution trace, videos, screenshots to eliminate flakes.

# Playwright

## No trade-offs • No limits

Browsers run web content belonging to different origins in different processes. Playwright is aligned with the modern browsers architecture and runs tests out-of-process. This makes Playwright free of the typical in-process test runner limitations.

**Multiple everything.** Test scenarios that span multiple **tabs**, multiple **origins** and multiple **users**. Create scenarios with different contexts for different users and run them against your server, all in one test.

**Trusted events.** Hover elements, interact with dynamic controls, produce trusted events. Playwright uses real browser input pipeline indistinguishable from the real user.

**Test frames, pierce Shadow DOM.** Playwright selectors pierce shadow DOM and allow entering frames seamlessly.

# Playwright

## Full isolation • Fast execution

**Browser contexts.** Playwright creates a browser context for each test. Browser context is equivalent to a brand new browser profile. This delivers full test isolation with zero overhead. Creating a new browser context only takes a handful of milliseconds.

**Log in once.** Save the authentication state of the context and reuse it in all the tests. This bypasses repetitive log-in operations in each test, yet delivers full isolation of independent tests.

# Playwright

## Powerful Tooling

**Codegen.** Generate tests by recording your actions. Save them into any language.

**Playwright inspector.** Inspect page, generate selectors, step through the test execution, see click points, explore execution logs.

**Trace Viewer.** Capture all the information to investigate the test failure. Playwright trace contains test execution screencast, live DOM snapshots, action explorer, test source, and many more.

# Playwright

## Resilient • No flaky tests

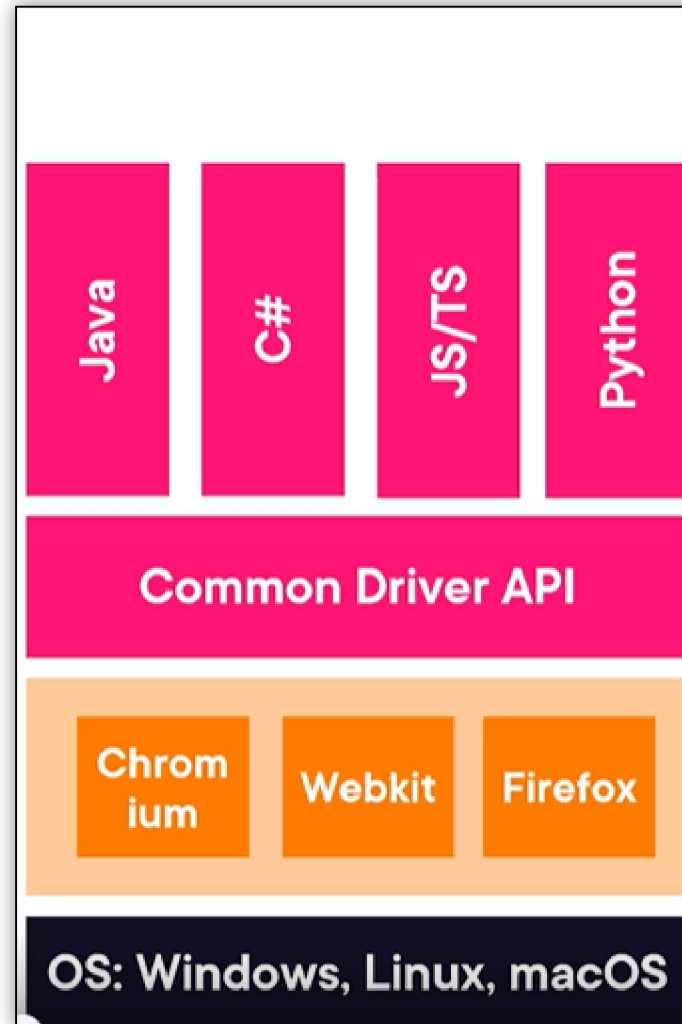
**Auto-wait.** Playwright waits for elements to be actionable prior to performing actions. It also has a rich set of introspection events. The combination of the two eliminates the need for artificial timeouts - the primary cause of flaky tests.

**Web-first assertions.** Playwright assertions are created specifically for the dynamic web. Checks are automatically retried until the necessary conditions are met.

**Tracing.** Configure test retry strategy, capture execution trace, videos, screenshots to eliminate flakes.

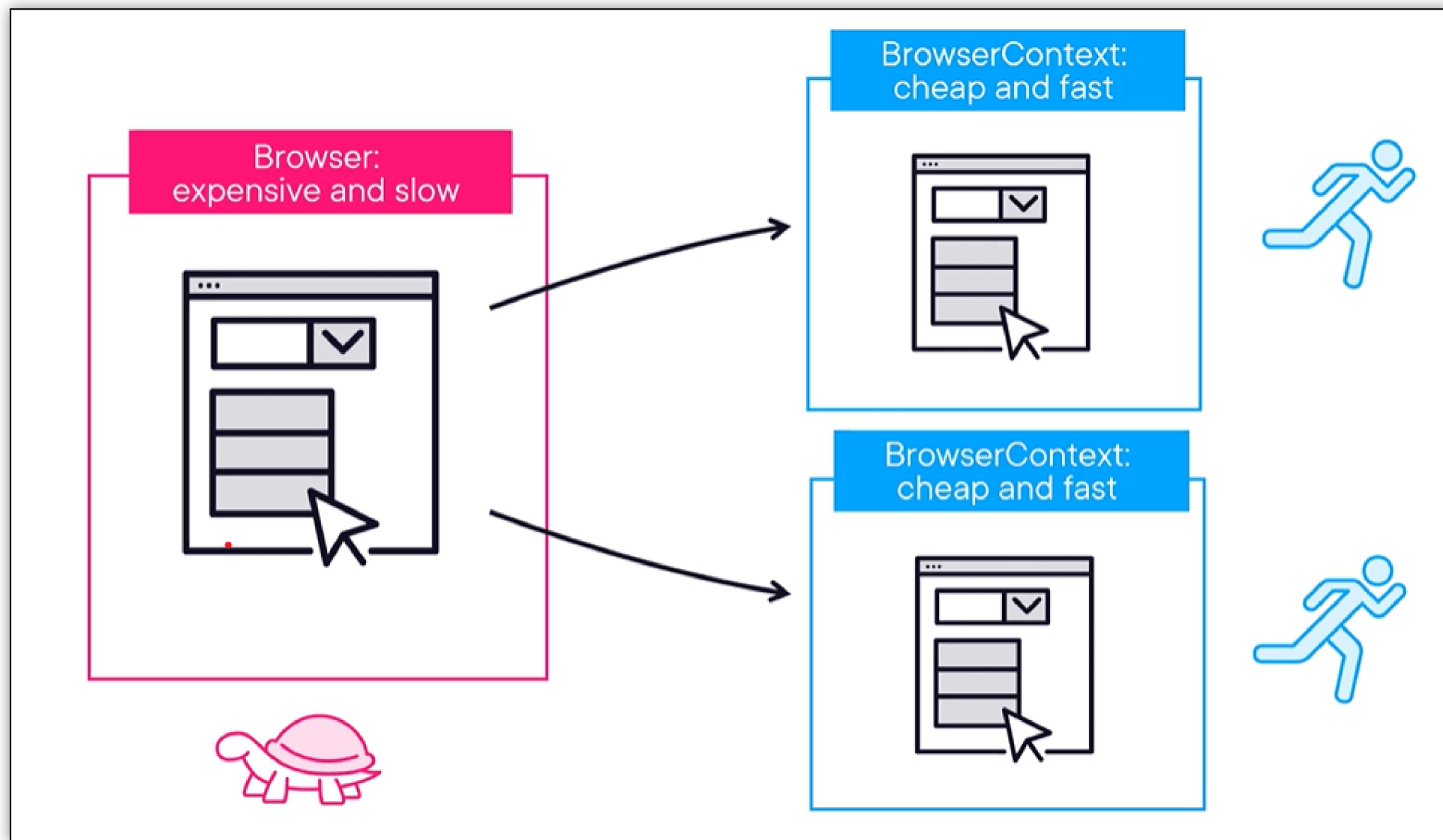
# One Team

---





# Context



# Autowait

## Auto-waiting

### Introduction

Playwright performs a range of actionability checks on the elements before making actions to ensure these actions behave as expected. It auto-waits for all the relevant checks to pass and only then performs the requested action. If the required checks do not pass within the given `timeout`, action fails with the `TimeoutError`.

For example, for `locator.click()`, Playwright will ensure that:

- locator resolves to exactly one element
- element is `Visible`
- element is `Stable`, as in not animating or completed animation
- element `Receives Events`, as in not obscured by other elements
- element is `Enabled`

Here is the complete list of actionability checks performed for each action:

Action	Visible	Stable	Receives Events	Enabled	Editable
<code>locator.check()</code>	Yes	Yes	Yes	Yes	-
<code>locator.click()</code>	Yes	Yes	Yes	Yes	-
<code>locator.dblclick()</code>	Yes	Yes	Yes	Yes	-
<code>locator.setChecked()</code>	Yes	Yes	Yes	Yes	-

# Playwright

im Vergleich  
mit Cypress  
und Selenium

# Selenium

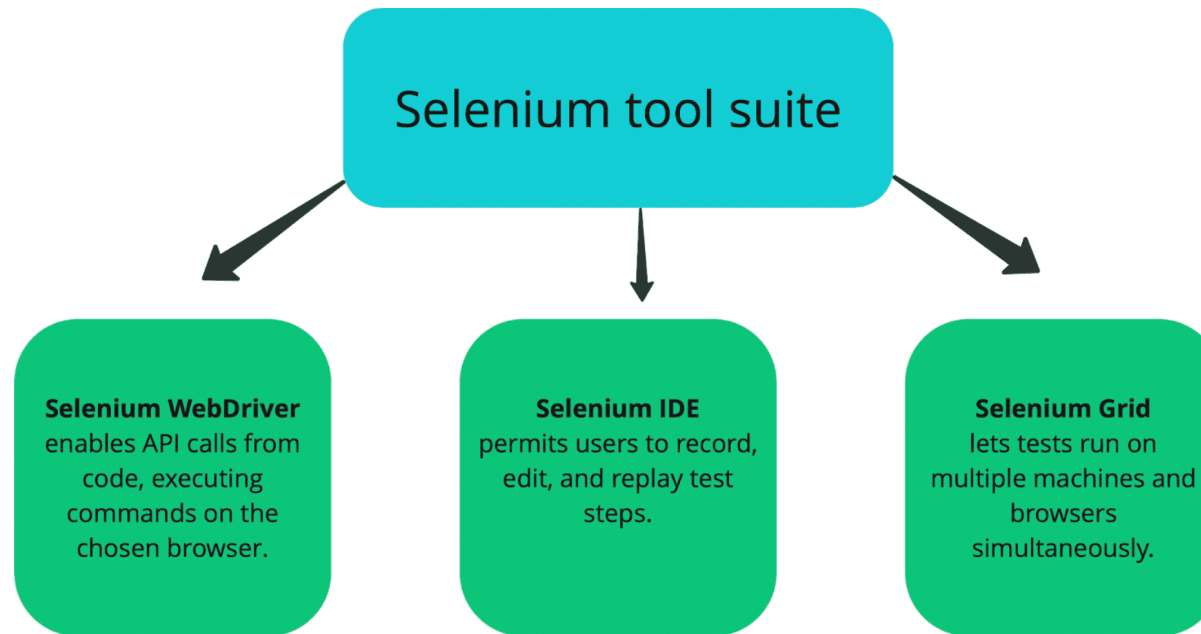
---

Die Geschichte des Selenium-Projekts begann im Jahr 2004. Selenium ist ein Open-Source-Testframework, das für automatisierte Tests von Webanwendungen auf verschiedenen Plattformen und Browsern eingesetzt wird.

Mit Selenium können Sie Anwendungen in den wichtigsten Browsern mit verschiedenen Programmiersprachen wie Java, JavaScript, C#, Ruby, PHP und Python automatisieren.

# Selenium

Selenium besteht aus mehreren verschiedenen Tools. Schauen wir uns die Tool-Suite im Detail an.



# Selenium

---

Selenium WebDriver. Diese Bibliothek ermöglicht API-Aufrufe aus dem Code heraus und führt Befehle auf dem gewählten Browser aus. Tester verwenden Selenium Locators, um Elemente zu finden, Testfälle zu erstellen und mit ihnen über WebDriver-APIs zu interagieren.

# Selenium

---

Selenium IDE. Mit diesem Tool können Benutzer Testschritte aufzeichnen, bearbeiten und wieder abspielen. Als Add-on für Firefox, Chrome und Edge vereinfacht die IDE die Erstellung und Ausführung automatisierter Tests ohne jegliche Programmierkenntnisse.

# Selenium

---

Selenium Grid. Mit diesem Tool können Tests auf mehreren Rechnern und Browsern gleichzeitig ausgeführt werden. Selenium Grid ist unverzichtbar für grosse Testsuiten, die schnell ausgeführt werden müssen, da die Tests auf verschiedenen Rechnern, Betriebssystemen und Browsern laufen. Es besteht aus einem zentralen Hub und mehreren Knotenpunkten, die die Testausführung auf verschiedene Umgebungen verteilen.



# Cypress

---

Das 2014 gegründete Unternehmen Cypress.io revolutionierte das Web-Testen durch die direkte Integration in den Browser für agile Echtzeit-Tests.

Ursprünglich als **internes Tool entwickelt**, entwickelte es sich zu einer weit verbreiteten Plattform, die von Unternehmen wie Slack, Netflix und Disney genutzt wird. Cypress bietet eine kostenlose Open-Source-App und eine abonnementbasierte Cypress Cloud, die umfassende Testtools und verwertbare Erkenntnisse liefert.

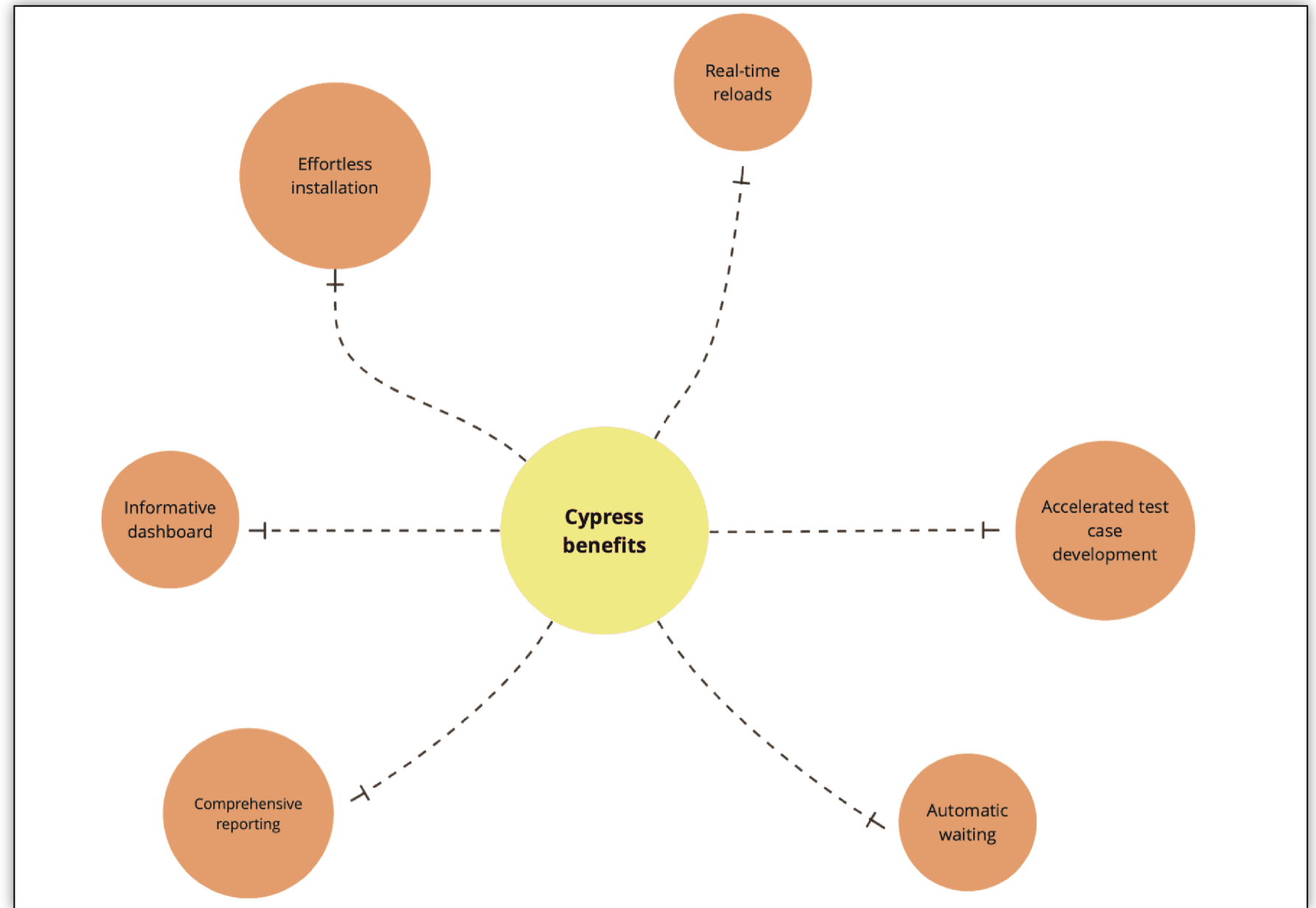
# Cypress

---

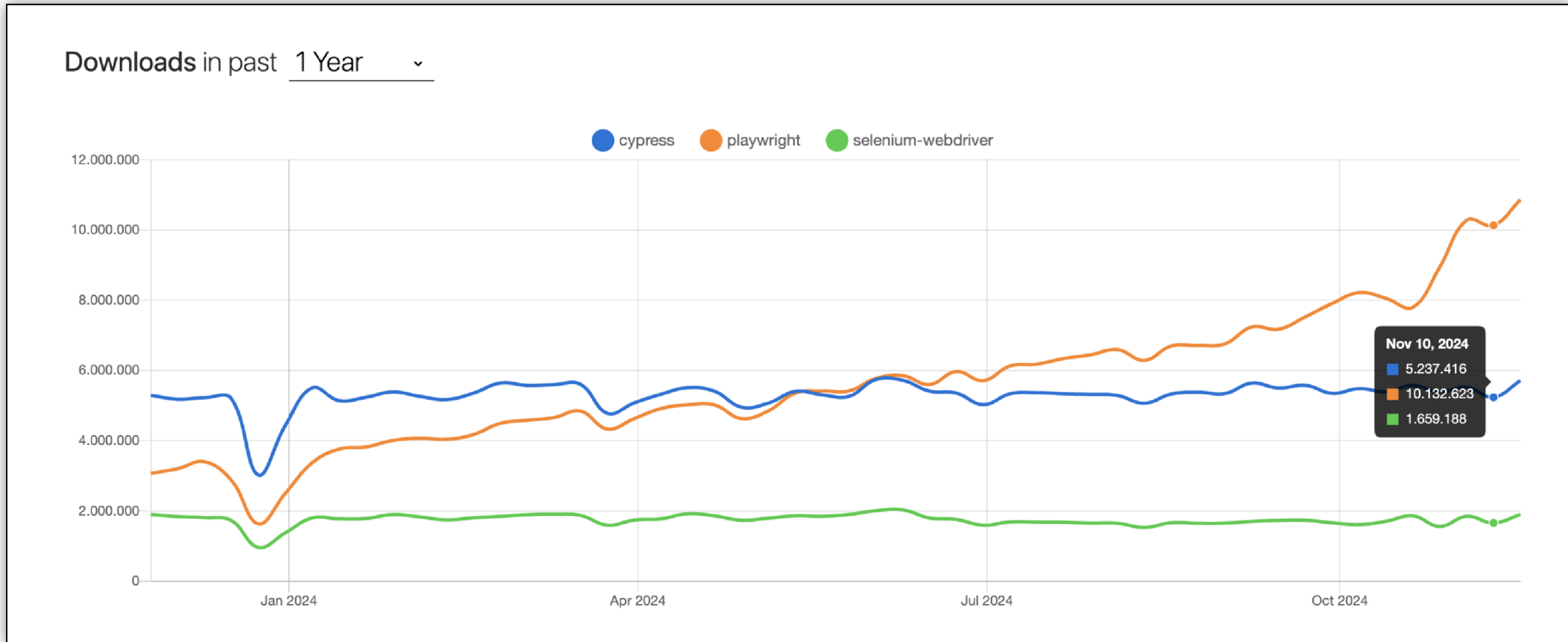
Eines der bemerkenswerten Merkmale von Cypress ist sein rationalisierter Einrichtungsprozess. Die meisten notwendigen Komponenten für die Erstellung von Testfällen sind in Cypress gebündelt, sodass keine weiteren Abhängigkeiten erforderlich sind.

Cypress arbeitet innerhalb des Browsers und führt Testsuiten schnell aus und übertrifft damit Frameworks wie Selenium.

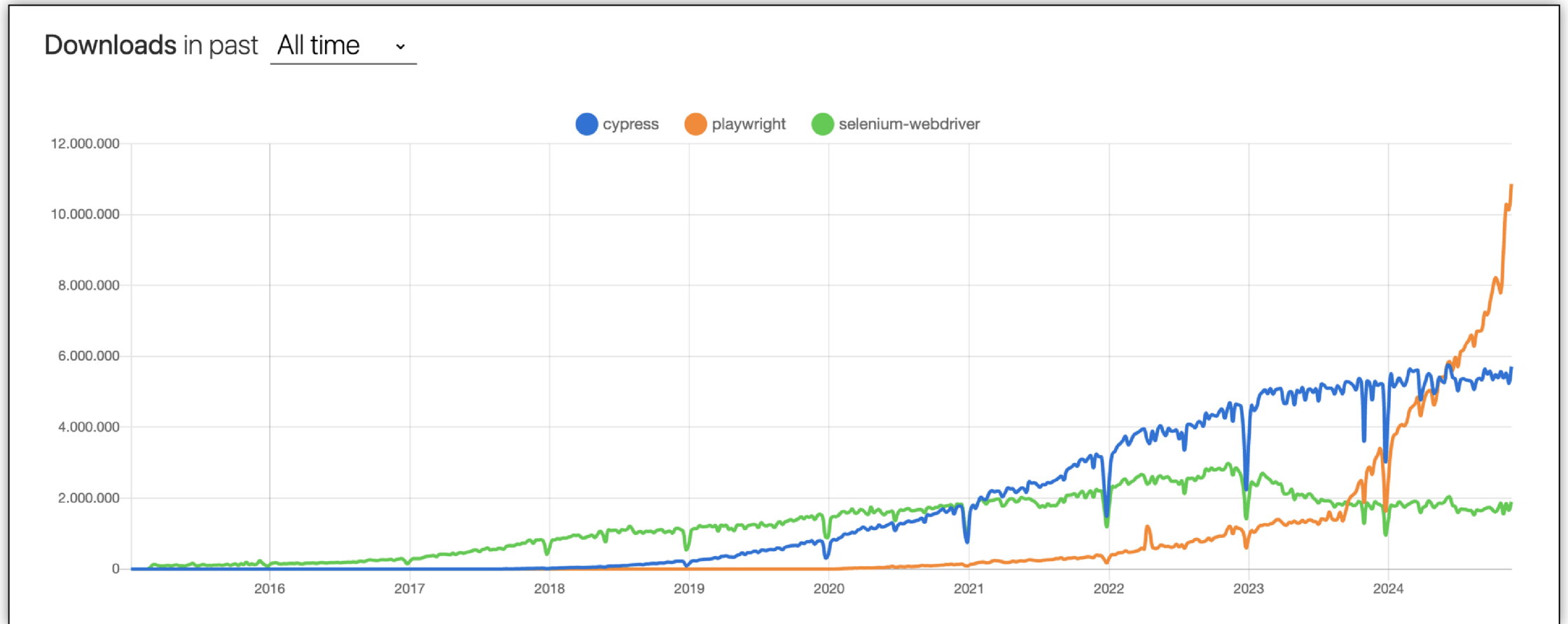
# Cypress















# Cypress - Selenium - Playwright



# Cypress - Selenium - Playwright



# Cypress - Selenium - Playwright

Stats			Stars	Issues	Version	Updated ?	Created ?	Size
	cypress	  	47.148	1.428	13.16.0	8 days ago	11 years ago	install size 24.9 MB
	playwright	  	67.225	679	1.49.0	9 days ago	10 years ago	install size 10.6 MB
	selenium-webdriver	  	30.822	218	4.27.0	2 days ago	12 years ago	install size 19.4 MB

# Hey ChatGPT

---

Würdest du Playwright, Cypress oder Selenium empfehlen?

- **Playwright** eignet sich hervorragend für moderne Web-Apps, schnelle Tests und Cross-Browser-Tests, wenn du mit verschiedenen Browsern arbeiten musst und auf Performance und moderne Features Wert legst.
- **Cypress** ist eine sehr gute Wahl, wenn du eine benutzerfreundliche Lösung für Tests mit schnellen Feedback-Zyklen und guter Debugging-Unterstützung benötigst, aber nur wenig Wert auf vollständigen Cross-Browser-Support legst.
- **Selenium** ist nach wie vor eine solide Wahl für die langfristige Unterstützung von Tests in einer Vielzahl von Browsern und bei komplexen Integrationen, jedoch könnte es in Bezug auf Geschwindigkeit und Benutzerfreundlichkeit hinter den neueren Tools zurückbleiben.

# Fragen

---





# Ende

Das war alles für dieses Kapitel

---