Xebia | Xpirit

Automate your cloud application testing with Playwright

Randy Pagels
DevOps Architect | Xpirit USA

Microsoft Solutions Partner

GitHub Verified Partner

Let's connect

Randy-Pagels

@RandyPagels

PagelsR

https://xpirit.com

scan ME

https://qrco.de/beHAED

Let's connect!

Randy Pagels

DevOps Architect

rpagels@xpirit.com

Platinum Sponsors

improving
It's what we do. ™

FANDUEL

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Speaker Dinner Sponsor

Xebia | Xpirit

Gold Sponsors

atmosera®

tyler technologies

SAVE THE DATE

.NET Conf 2023

November 14 – 16

.NET 8 GA launch

80+ live sessions, 24-hr continuous broadcast

Live Q&A, games, prizes, digital swag

www.dotnetconf.net

Agenda

- Testing Challenges & Goals
- Playwright Overview
- Getting Started
- Authoring, Debugging, Running
- Additonal Built in Features
- Time Travel Experience
- Continuous Integration Testing
- Demos

Xebia | Xpirit

UI Testing Challenges

Consistent design across browsers! Dynamic content and maintaining performance!

- Testing is **hard**
- Testing takes **time to learn**
- Testing takes **time to build**
- Tests are **slow** – they take too long to run!
- Tests are **brittle** – they break whenever the app changes!
- Tests are **flaky** – they crash all the time!

Modern Testing Goals

Major goals for modern web testing

- Make testing **easy**
- Make testing **fun**
- Focus on **fast feedback loops** rather than certain types of tests.
- Make test development as **painless** as possible.
- Choose test tooling that naturally **complements dev workflows**.

Test Automation Practices

Tests should be automated when they give positive ROI

Advantages

- Tests can be **rerun at any time** the same way.
- Tests can run as **part of CI/CD**.
- **Setup** and **cleanup** can be **automated**.
- **Frameworks** can **provide** reports, logs, and screenshots.

ProTips!

- Each test should **focus** on **one** main **behavior** or variation.
- Tests should **run independently** of each other and in **any order**.
- Tests should be **self-descriptive** and **intuitively understandable**.
- Do not automate every test – use a risk-based strategy with ROI.
- Tests should run **in parallel**, especially in CI/CD.

Xebia | Xpirit

Playwright Overview

AN INTRODUCTION TO PLAYWRIGHT

@playwrightweb
https://playwright.dev

READY TO START YOUR PLAYWRIGHT RUN ?

5 RESOURCES TO HELP YOU JUMPSTART YOUR TEST ADVENTURE

1 READ THE DOCS
- https://aka.ms/playwright
- https://aka.ms/playwright/quickstart
- Install Playwright
- Run your first Test
- Explore Tools & Libraries

2 EXPLORE THE API
https://aka.ms/playwright/api
JS, Java, Python, .NET
- Playwright Test Runner
- Test Reporter
- Testing + Experimental

3 VISIT THE REPO
https://aka.ms/playwright/github
- Playwright Framework source
- Examples + Release docs
- Discussions

4 FOLLOW @playwrightweb
- Framework release updates
- Content + event announcements
- Community interactions

5 CHECKOUT DEMO CODE
https://aka.ms/playwright/demo
- Test script examples
- Github Actions integration
- Test Reporting outputs

WHAT IS PLAYWRIGHT?

0 AN OPEN SOURCE TESTING FRAMEWORK
- ENABLES RELIABLE END-TO-END TESTING AND AUTOMATION
- FOR MODERN WEB APPLICATIONS

https://github.com/microsoft/playwright
- TOOLING
- AUTOMATION
- API

CHECK OUT THE SOURCE

WHY SHOULD I BE USING PLAYWRIGHT?

1. ONE API, MULTI-EVERYTHING
2. RESILIENT, NO FLAKY TESTS
3. LIMITLESS, NO TRADEOFFS
4. ISOLATED, FAST EXECUTION
5. TOOLING, CREATE-DEBUG-ANALYZE

LET'S TALK ABOUT EACH OF THESE!

1 ONE API / MULTI EVERYTHING

- ANY BROWSER ANY PLATFORM
  Chromium, WebKit, Firefox
  MacOS, Linux, Windows
  LOCAL, HEADLESS, HEADED
- CROSS LANGUAGE
  JavaScript, TypeScript, Python, Java, .NET — libraries.
- TEST MOBILE WEB
  Native emulation
  - Chrome on Android
  - Mobile Safari on iOS
  - Same Rendering Engine on Cloud and Desktop.

PLAYWRIGHT – the Grey –
CHROMIUM WEBKIT FIREFOX
ONE API TO RULE THEM ALL
CROSS BROWSER TESTING
TEST AUTOMATION AT CLOUD SCALE

2 RESILIENT! NO FLAKY TESTS...

TESTING ENTRY
ALMOST READY FOR YOU .. HANG ON .. LET ME CHECK AGAIN ...
ARE YOU EXPECT-ing US?
TEST ACTIONS
REDUCE TEST COMPLEXITY INCREASE READABILITY + RELIABILITY
WEB PAGE

1 AUTO-WAIT
No more artificial timeouts to manage
- waits for elements to be ACTIONABLE before execution of test steps
- + RICH introspection events available!

2 WEB-FIRST ASSERTIONS
tailored for the dynamic web!
ASYNC AWAIT EXPECT
- convenience async matchers for assertions
- separate timeout (from test) for web first assert with default = 5 secs

3 TRACING
configurable strategy
retry? record traces? capture screenshot?
- checks conditions till met

3 NO TRADE-OFFS / LIMITS

TESTING API
TEST RUNNER API
TEST CONFIG OPTIONS

1 ALIGNED WITH MODERN WEB ARCHITECTURES
- RUN TESTS OUT OF PROCESS .. FREED FROM IN-PROCESS TEST RUNNER CONSTRAINT

2 MULTIPLE EVERYTHING
MULTIPLE ORIGINS
MULTIPLE TABS
MULTIPLE USERS
TEST ALL THE THINGS
ONE TEST MANY CONTEXTS
WRITE MORE COMPREHENSIVE TEST ACTIONS & SCRIPTS!
MORE REALISTIC USER ACTION FLOWS

3 TRUSTED EVENTS
I'M IN YOUR BROWSER – ACTING LIKE ANY USER!
- TEST HOVER ELEMENTS
- INTERACT WITH DYNAMIC CONTROLS
- USE REAL BROWSER INPUT PIPELINES

INDISTINGUISHABLE FROM REAL USERS!

4 TEST FRAMES
PIERCE SHADOW DOM WITH PLAYWRIGHT SELECTORS!

4 FULL ISOLATION / FAST EXECUTION

1 BROWSER CONTEXT
TEST SCRIPT
LIKE HAVING A BRAND NEW BROWSER PROFILE EACH TIME
- CREATES A BROWSER CONTEXT PER TEST IN JUST ms.
  - FULL TEST ISOLATION
  - ZERO OVERHEADS
  - LESS COMPLEXITY FOR TESTER

2 LOG IN ONCE!
TEST SCRIPT
1 STORE AUTH STATE
2 LOAD AUTH STATE (from stored location)
TEST NEW THING
- SAVE AUTHENTICATION STATE (tokens, cookies) FROM FIRST RUN – & REUSE IN ALL SUBSEQUENT TESTS
- BYPASS REPETITIVE LOGIN OPERATIONS (many third party sites don't want automated logins)
- STILL FULL ISOLATION FOR INDEPENDENT TESTS

5 POWERFUL TOOLING!

1 CODEGEN
INTERACTIVE SESSION
AUTO CAPTURED
- GENERATE TESTS BY RECORDING USER ACTION (FLOW)
- SAVE IN ANY SUPPORTED LANG. (Java, JS/TS, Python etc.)

2 INSPECTOR
TEST – DEBUG
INSPECTOR
- INSPECT PAGE, LOGS
- GENERATE SELECTOR
- STEP THROUGH EXEC
- SEE CLICK POINTS

3 TRACE VIEWER
- CAPTURE TEST PROGRESS (STATE, FAILURES, TIMING)
- SCREENCASTS, SNAPSHOTS, ACTION EXPLORER, TEST SOURCE (POST-MORTEM ANALYSIS)

ILLUSTRATED BY @SKETCHTHEDOCS

Playwright Overview

Open-source web test automation library on Node.js, which makes test automation easier for browsers based on Chromium, Firefox, and Webkit through a single API.

- ✓ **Cross-Browser**
  - Chromium, Firefox, and WebKit
- ✓ **Cross-language**
  - JavaScript, Python, and C#
- ✓ **Cross-platform**
  - Windows, Linux, and macOS
- ✓ Modern Asynchronous API
  - async/await
- ✓ Rich Element Interaction
  - wide range of built-in functions
- ✓ **Screenshots and Video Recording**
  - visual evidence of issues
- ✓ **Full Isolation – Fast Execution**
  - separate browser contexts

Reliable end-to-end testing for modern web apps

…by Microsoft

…and the people who made Puppeteer

https://playwright.dev/

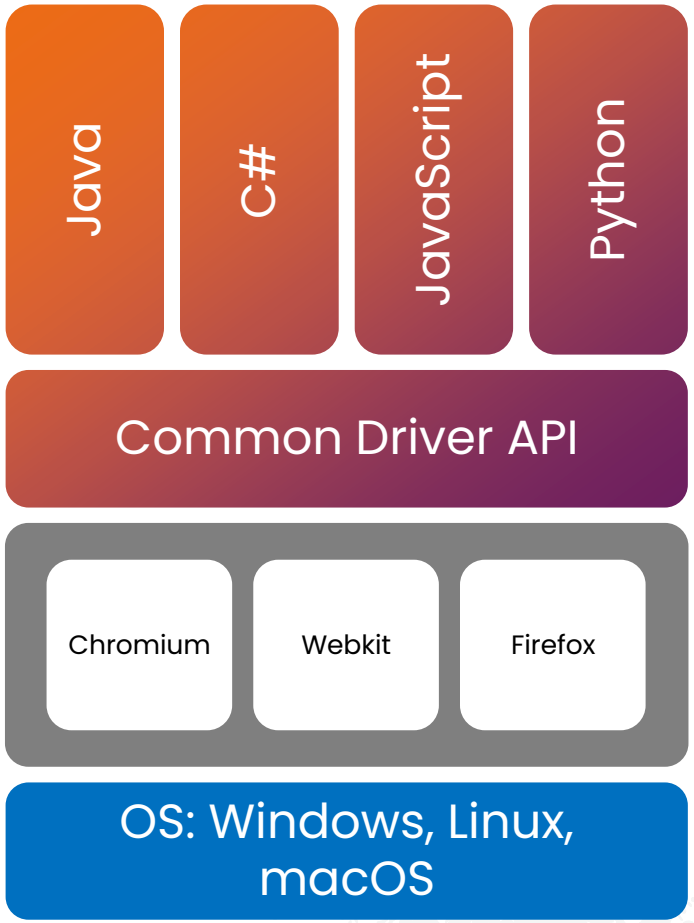
https://github.com/microsoft/playwright

Playwright Architecture

- Language Bindings

- Single Automation Protocol
- Unify all debugging protocols

- (Native) browser debugging protocols, e.g. ChromeDevTools Protocol, (Safari) Web Inspector



Getting Started

Setting up using VS Code

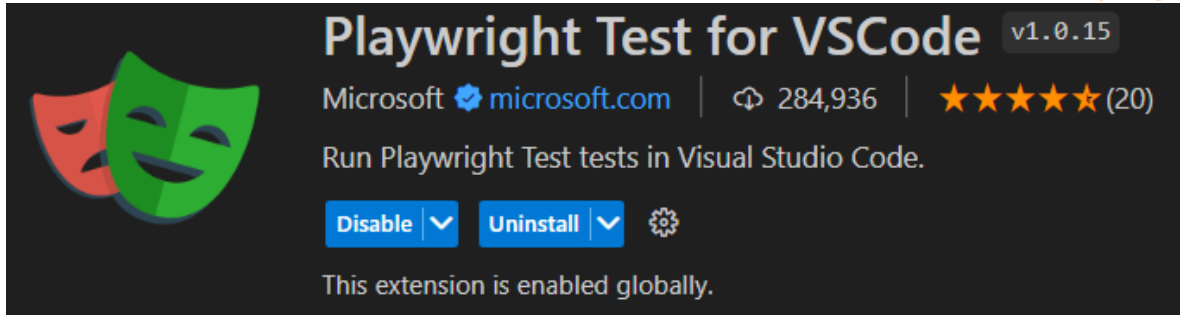
Install Extension

Install Playwright
- npx init playwright@latest

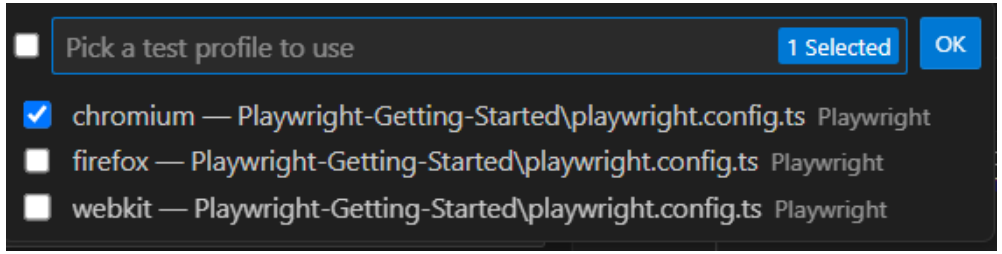
Run Tests
- npx playwright test
- npx playwright test --ui

Test Reports
- npx playwright show-report

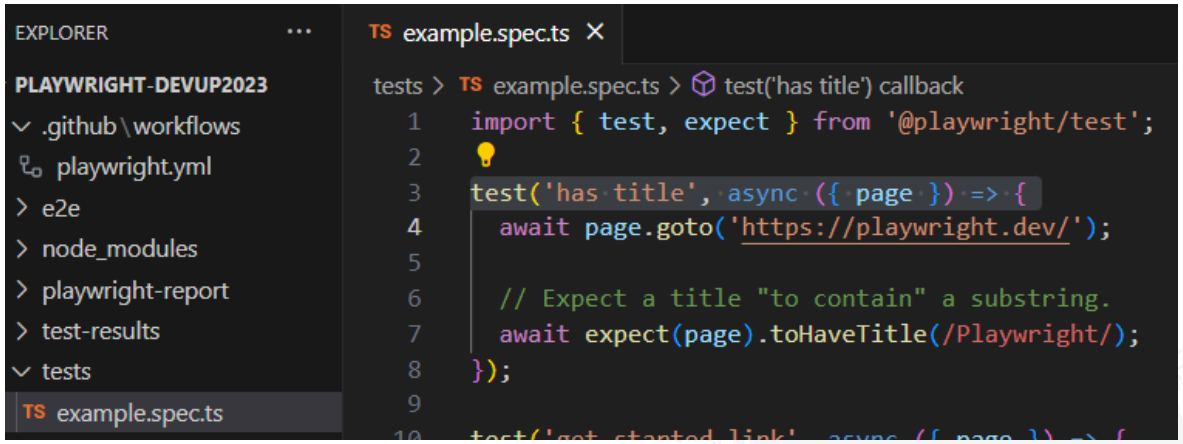
1



2



3



Test Generator

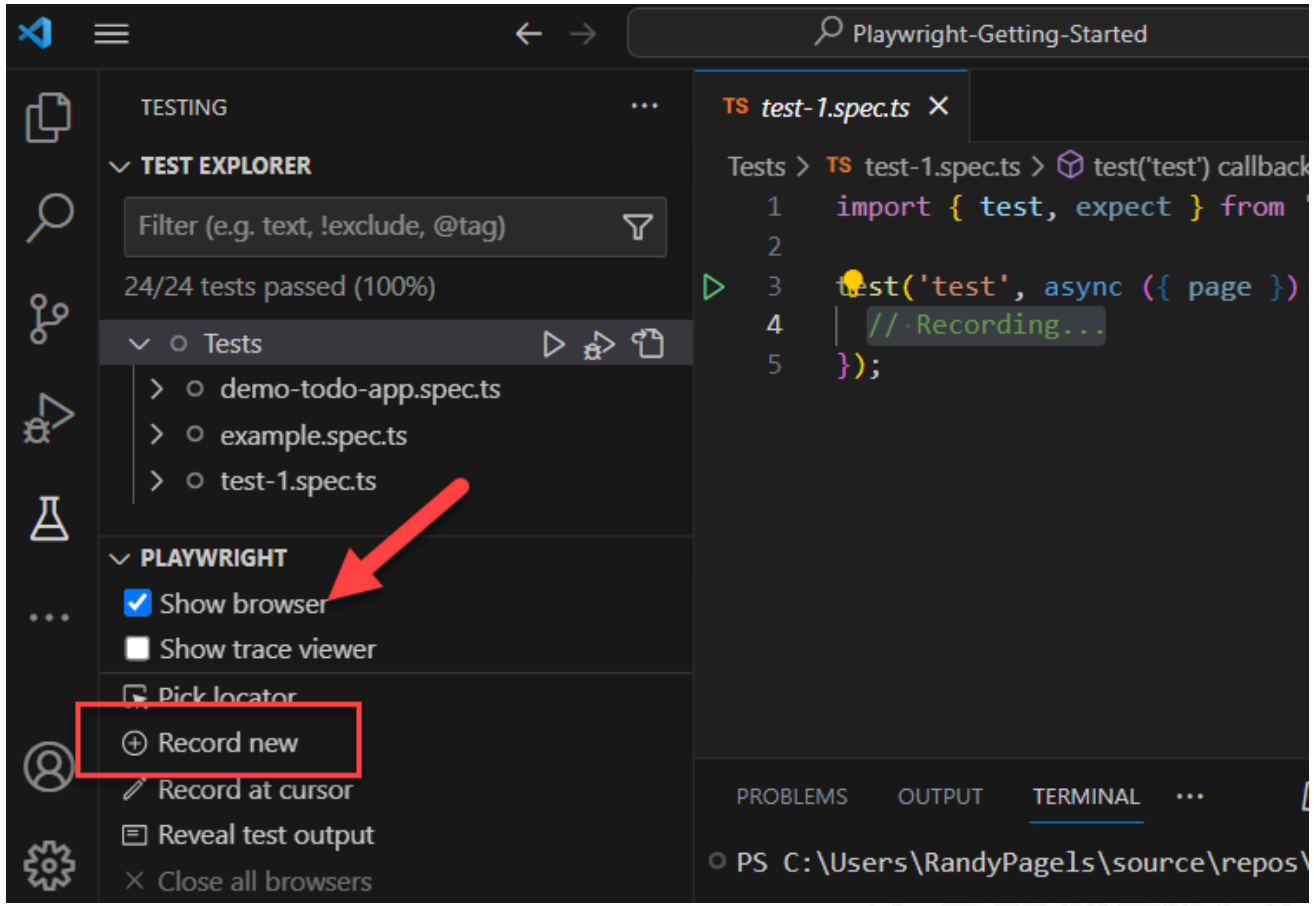
Playwright Test Generator is a GUI tool that helps you explore recorded Playwright traces after the script has ran.

CLI

- npx playwright codegen

VS Code

- Use Plugin, “Record Now”



Xebia | Xpirit

Trace Viewer

Playwright Trace Viewer is a GUI tool that helps you explore recorded Playwright traces after the script has ran.

CLI

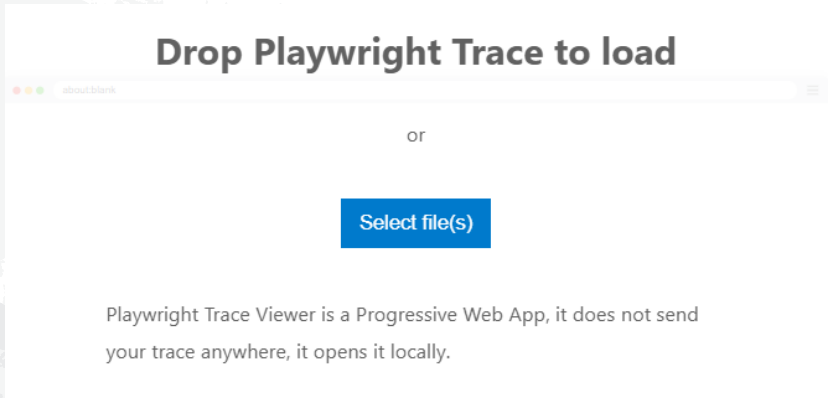
- npx playwright test test.ts --trace on

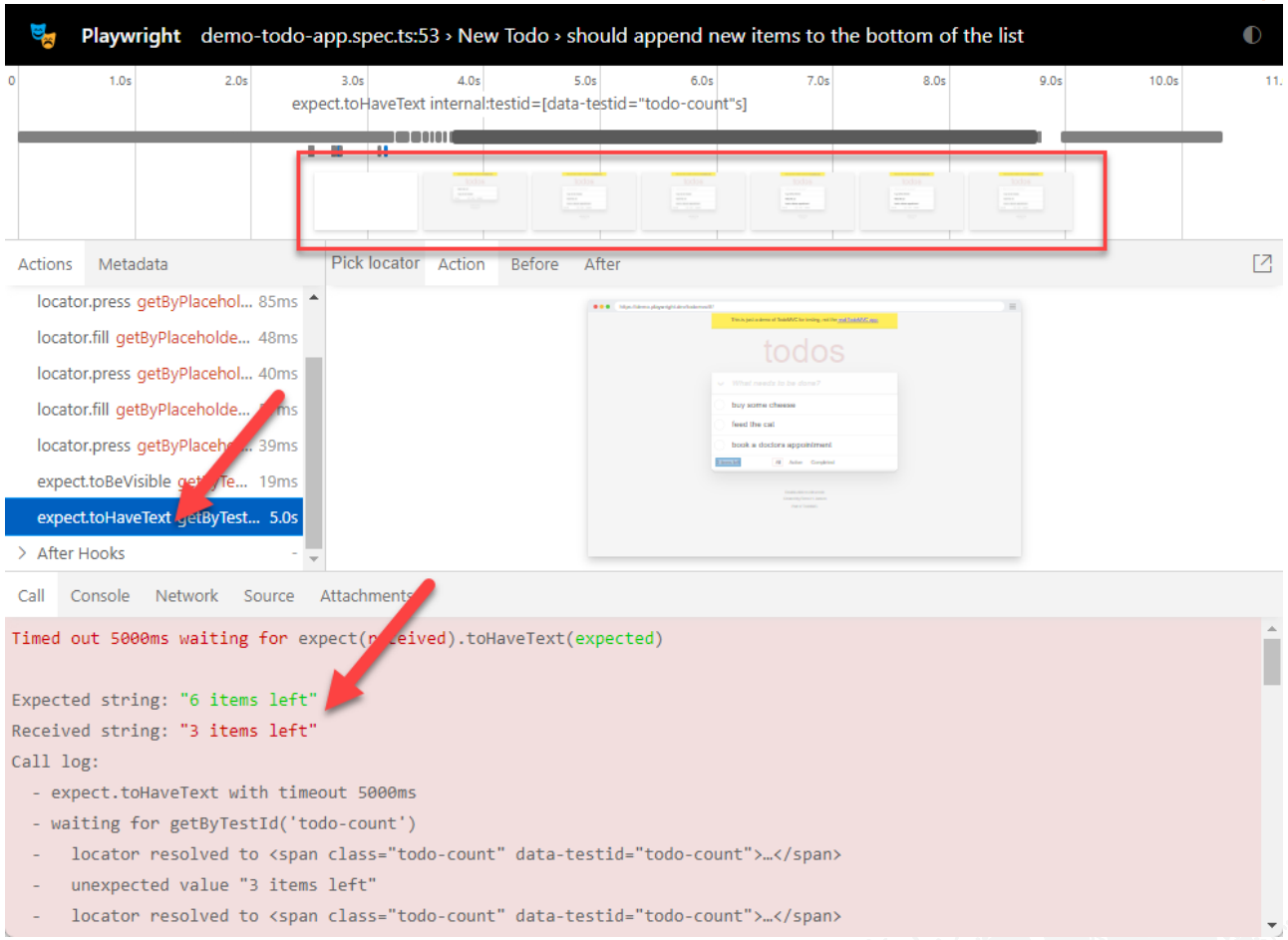
VS Code

- Use Plugin

Hosted Trace File Viewer

- https://trace.playwright.dev





Xebia | Xpirit

Report List

Playwright Test comes with a few built-in reporter.

HTML Report

- npx playwright test --reporter=**html**

List Report

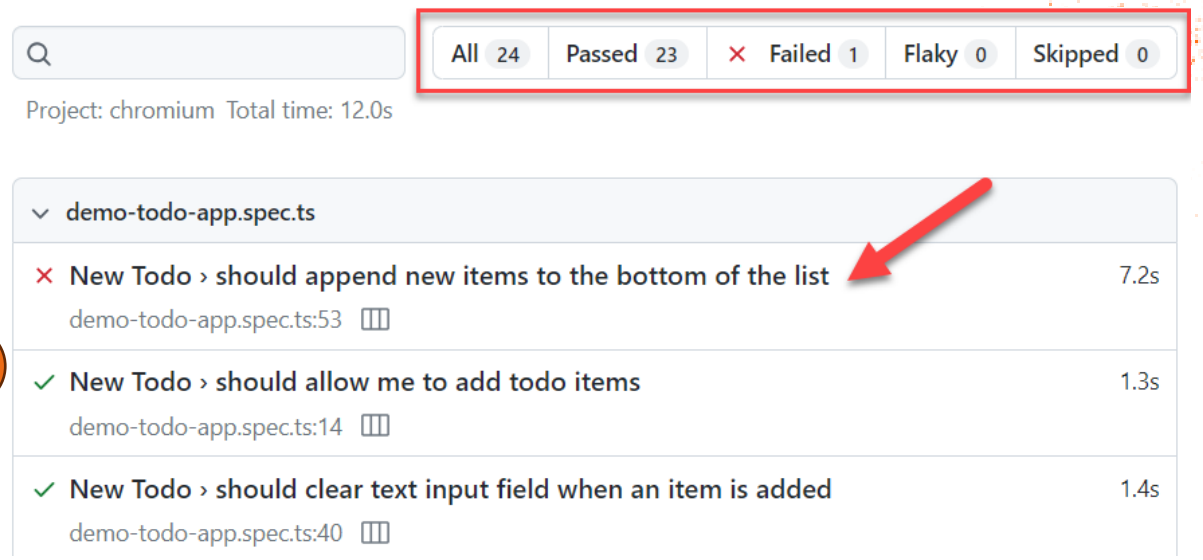
- npx playwright test --reporter=**list**

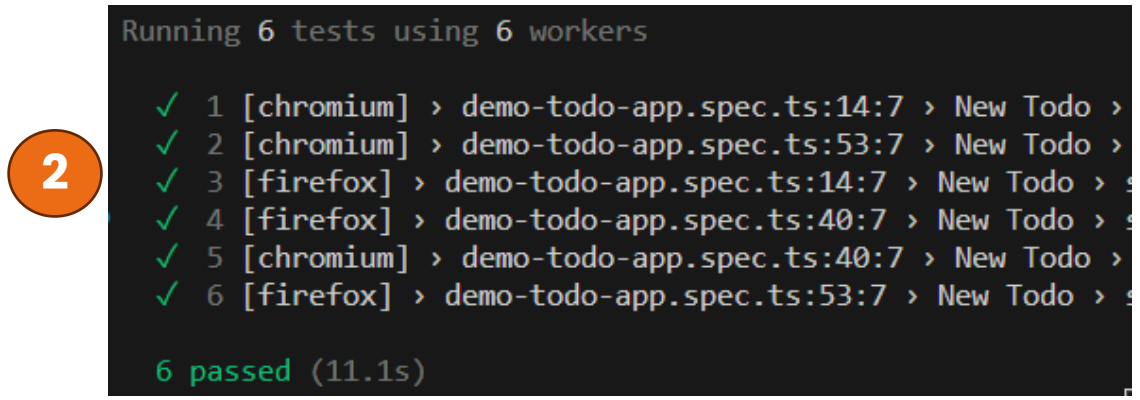
Line Report

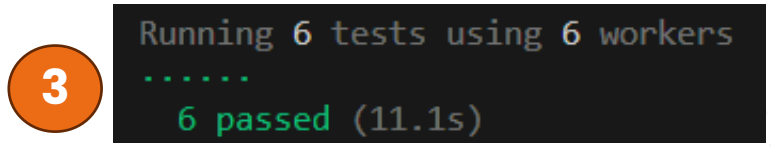
- npx playwright test --reporter=**line**

Dot Report

- npx playwright test --reporter=**dot**







Xebia | Xpirit

Playwright UI Mode

Explore, run and debug tests with a time travel experience complete with watch mode

CLI

npx playwright test testname.ts --ui

DOM Inspection

- page structure, inspect elements, view attributes.

Console Logs

- JavaScript-generated logs, errors, and warnings.

Network Monitoring

- Requests, timings, inspect headers, payloads.

Interactivity

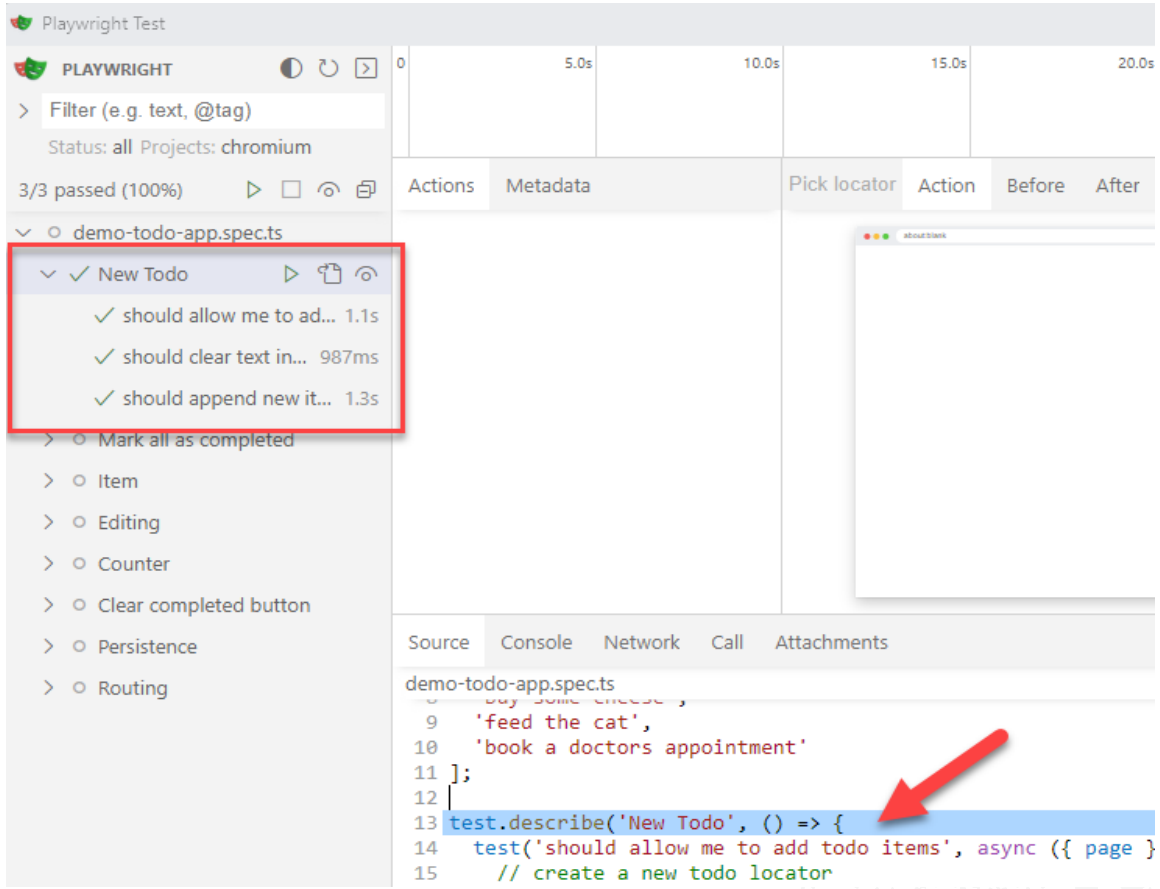
- Trigger events, clicks, real-time page responses.

Element Highlighting

- Hover over elements to highlight them.

Source Code Access

- View HTML, CSS, JS source for debugging.



Xebia | Xpirit

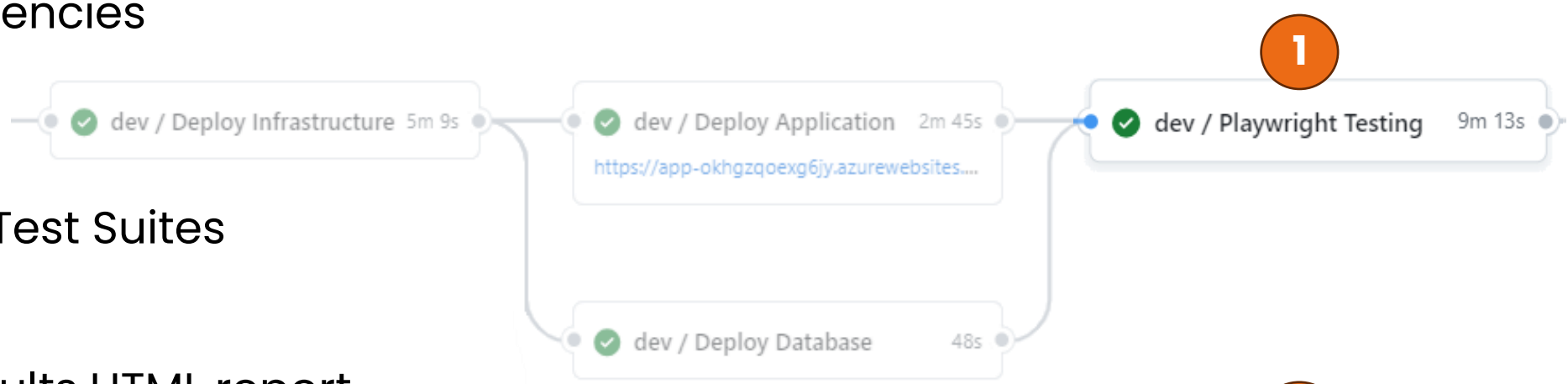
Xebia | Xpirit

Continuous Integration Testing

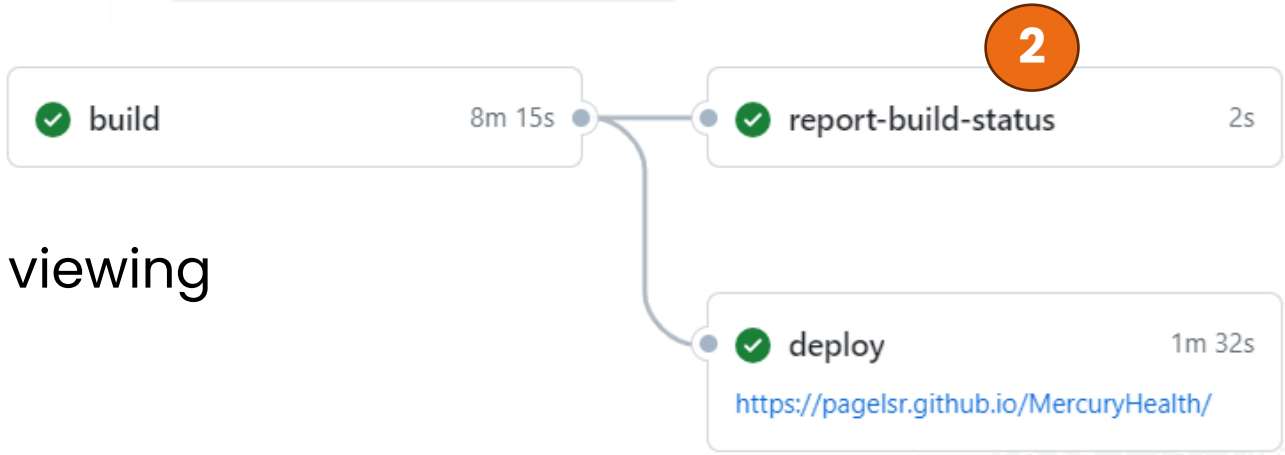
Continuous Testing

Tests run on each push and pull request into the main/master branch

- Installs all dependencies
- Installs Playwright
- Installs Browsers
- Run all the tests
- Parallelization for Test Suites



- Generate Test Results HTML report
- Generate Screenshots
- Generate Video Recordings
- Generate Debug Trace file
- Deploy static files to GitHub Pages for viewing



Xebia | Xpirit

Demos

Xebia | Xpirit

Azure AD Authentication

- Store the credentials of test accounts using environment variables.

```
test(`example test`, async ({ page }) => {
  // ...
  await page.getByLabel('User Name').fill(process.env.USERNAME);
  await page.getByLabel('Password').fill(process.env.PASSWORD);
});
```

- Set the values of these variables to use your CI system's secret management or Azure Key Vault.

```
testE2E:
  name: Run Playwright E2E tests
  runs-on: ubuntu-latest
  container: mcr.microsoft.com/playwright:v1.27.0-jammy
  env:
    USERNAME: ${{secrets.USERNAME}}
    PASSWORD: ${{secrets.PASSWORD}}
```

- Local Development? Use .env files and add to .gitignore.

```
# .env file
STAGING=0
USERNAME=me
PASSWORD=secret
```

Note: MFA cannot be fully automated and requires manual intervention.

Tool Comparison

Playwright is not the only browser automation tool out there

Selenium	Cypress	Playwright
Popular among testers	Popular among developers	So hot right now
C#, Java, JavaScript, Ruby, Python	JavaScript only	C#, Java, JavaScript, Python
All major full browsers	Chrome, Edge, Firefox, Electron	Chromium+, Firefox, WebKit
WebDriver protocol	In-browser JavaScript	Debug protocols
Just a tool – requires a framework	Full, modern framework	Full, modern framework
Can be slow/flaky if waiting is poor	Visual execution but can be slow	Typically the fastest execution
Open source, standards, & gov	Open source from Cypress	Open source from Microsoft

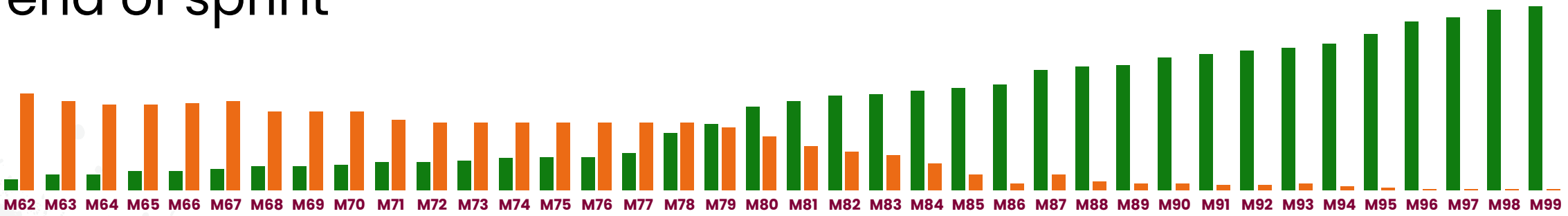
https://automationpanda.com/2023/07/28/modern-web-testing-with-playwright/

Azure DevOps team

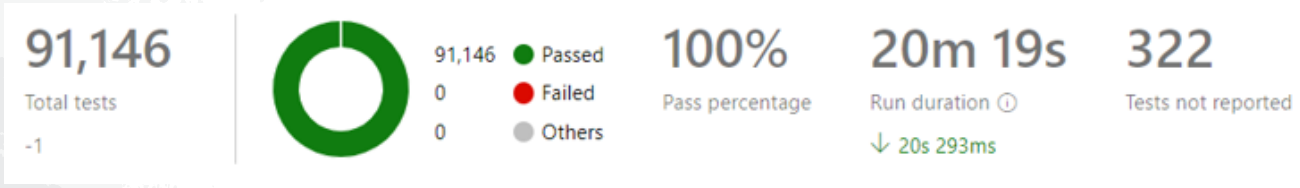
Balancing UI vs Unit Testing in your strategy to maximizes test coverage and effectiveness.

Long running UI functional tests at end of sprint

Shifted to unit tests from automated UI functional tests



ADO PRs require 100% pass rate before merging



ADO Unit Tests in build pipeline for past 14 days

Microsoft Playwright Testing

Join the waitlist for Microsoft Playwright Testing private preview

- New Azure service for running Playwright tests.
- Cloud enabled to run Playwright tests at scale.
- High parallelization across operating system-browser combinations.
- Get tests done much faster.
- Speed up delivery of features without sacrificing quality.

Private Preview

To learn more about Microsoft Playwright Testing, refer to **https://aka.ms/mpt/private-preview-blog**.

Xebia | Xpirit

WANT TO LEARN ABOUT THE PLAYWRIGHT CLI?

The "--help" option on a command will give you usage documentation

```
(base) nitya@nityas-mbp ~ % npx playwright show-trace --help
Usage: npx playwright show-trace [options] [trace]

Show trace viewer

Options:
  -b, --browser <browserType>  browser to use, one of cr, chromium, ff, firefox, wk, webkit (default: "chromium")
  -h, --help                   display help for command

Examples:

  $ show-trace https://example.com/trace.zip
```

npx playwright show-trace --help

SCREENSHOTS TAKEN FOR PLAYWRIGHT v1.17.1 — try them out!

OPEN PAGE WITH DESIRED BROWSER OR EMULATED DEVICE WITH DESIRED CONTEXT LOAD/SAVE STATE!

1

```
(base) nitya@nityas-mbp ~ % npx playwright open --help
Usage: npx playwright open [options] [url]

open page in browser specified via -b, --browser

Options:
  -b, --browser <browserType>     browser to use, one of cr, chromium, ff, firefox, wk, webkit (default: "chromium")
  --channel <channel>             Chromium distribution channel, "chrome", "chrome-beta", "msedge-dev", etc
  --color-scheme <scheme>         emulate preferred color scheme, "light" or "dark"
  --device <deviceName>           emulate device, for example  "iPhone 11"
  --geolocation <coordinates>     specify geolocation coordinates, for example "37.819722,-122.478611"
  --ignore-https-errors           ignore https errors
  --load-storage <filename>       load context storage state from the file, previously saved with --save-storage
  --lang <language>               specify language / locale, for example "en-GB"
  --proxy-server <proxy>          specify proxy server, for example "http://myproxy:3128" or "socks5://myproxy:8080"
  --save-storage <filename>       save context storage state at the end, for later use with --load-storage
  --save-trace <filename>         record a trace for the session and save it to a file
  --timezone <time zone>          time zone to emulate, for example "Europe/Rome"
  --timeout <timeout>             timeout for Playwright actions in milliseconds (default: "10000")
  --user-agent <ua string>        specify user agent string
  --viewport-size <size>          specify browser viewport size in pixels, for example "1280, 720"
  -h, --help                      display help for command
```

npx playwright open --help

2

```
(base) nitya@nityas-mbp ~ % npx playwright codegen --help
Usage: npx playwright codegen [options] [url]

open page and generate code for user actions

Options:
  -o, --output <file name>        saves the generated script to a file
  --target <language>             language to generate, one of javascript, test, python, python-async, csharp (default: "test")
  -b, --browser <browserType>     browser to use, one of cr, chromium, ff, firefox, wk, webkit (default: "chromium")
  --channel <channel>             Chromium distribution channel, "chrome", "chrome-beta", "msedge-dev", etc
  --color-scheme <scheme>         emulate preferred color scheme, "light" or "dark"
  --device <deviceName>           emulate device, for example  "iPhone 11"
  --geolocation <coordinates>     specify geolocation coordinates, for example "37.819722,-122.478611"
  --ignore-https-errors           ignore https errors
  --load-storage <filename>       load context storage state from the file, previously saved with --save-storage
  --lang <language>               specify language / locale, for example "en-GB"
  --proxy-server <proxy>          specify proxy server, for example "http://myproxy:3128" or "socks5://myproxy:8080"
  --save-storage <filename>       save context storage state at the end, for later use with --load-storage
  --save-trace <filename>         record a trace for the session and save it to a file
  --timezone <time zone>          time zone to emulate, for example "Europe/Rome"
  --timeout <timeout>             timeout for Playwright actions in milliseconds (default: "10000")
  --user-agent <ua string>        specify user agent string
  --viewport-size <size>          specify browser viewport size in pixels, for example "1280, 720"
  -h, --help                      display help for command
```

npx playwright codegen --help

EFFORTLESS TEST SCRIPT CREATION FTW!

- SAVE SCRIPT TO FILE
- PICK TARGET LANGUAGE
- CONFIGURE CONTEXT

3

```
Usage: npx playwright install [options] [browser...]

ensure browsers necessary for this version of Playwright are installed

Options:
  --with-deps  install system dependencies for browsers
  -h, --help   display help for command

Examples:
  - $ install
    Install default browsers.

  - $ install chrome firefox
    Install custom browsers, supports chromium, chrome, chrome-beta, msedge, msedge-beta, msedge-dev, firefox, webkit.
```

npx playwright install --help

INSTALL BROWSERS (+ SPECIFIC CHANNEL DIST.) ON DEMAND

USEFUL FOR TESTING AGAINST CUTTING EDGE BROWSER RELEASES!

LEARN MORE ABOUT PLAYWRIGHT COMMAND LINE TOOLS & OPTIONS
playwright.dev/docs/cli

```
(base) nitya@nityas-mbp ~ % npx playwright screenshot --help
Usage: npx playwright screenshot [options] <url> <filename>

capture a page screenshot

Options:
  --wait-for-selector <selector>  wait for selector before taking a screenshot
  --wait-for-timeout <timeout>    wait for timeout in milliseconds before taking a screenshot
  --full-page                     whether to take a full page screenshot (entire scrollable area)
  -b, --browser <browserType>     browser to use, one of cr, chromium, ff, firefox, wk, webkit (default: "chromium")
  --channel <channel>             Chromium distribution channel, "chrome", "chrome-beta", "msedge-dev", etc
  --color-scheme <scheme>         emulate preferred color scheme, "light" or "dark"
  --device <deviceName>           emulate device, for example  "iPhone 11"
  --geolocation <coordinates>     specify geolocation coordinates, for example "37.819722,-122.478611"
  --ignore-https-errors           ignore https errors
  --load-storage <filename>       load context storage state from the file, previously saved with --save-storage
  --lang <language>               specify language / locale, for example "en-GB"
  --proxy-server <proxy>          specify proxy server, for example "http://myproxy:3128" or "socks5://myproxy:8080"
  --save-storage <filename>       save context storage state at the end, for later use with --load-storage
  --save-trace <filename>         record a trace for the session and save it to a file
  --timezone <time zone>          time zone to emulate, for example "Europe/Rome"
  --timeout <timeout>             timeout for Playwright actions in milliseconds (default: "10000")
  --user-agent <ua string>        specify user agent string
  --viewport-size <size>          specify browser viewport size in pixels, for example "1280, 720"
  -h, --help                      display help for command
```

npx playwright screenshot--help

4

CAPTURE page screenshot with emulated device or specific contexts!!

- DEVICE TYPE
- COLOR SCHEME etc.
- GEO LOCATION

START HERE

THE PLAYWRIGHT CLI

```
(base) nitya@nityas-mbp ~ % npx playwright --help
Usage: npx playwright [options] [command]

Options:
  -V, --version                   output the version number
  -h, --help                      display help for command

Commands:
  open [options] [url]            open page in browser specified via -b, --browser
  codegen [options] [url]         open page and generate code for user actions
  install [options] [browser...]  ensure browsers necessary for this version of Playwright are installed
  install-deps [browser...]       install dependencies necessary to run browsers (will ask for sudo permissions)
  cr [options] [url]              open page in Chromium  ] same as open -b {cr|ff|wk}
  ff [options] [url]              open page in Firefox   ]
  wk [options] [url]              open page in WebKit    ]
  screenshot [options] <url> <filename>  capture a page screenshot
  pdf [options] <url> <filename>  save page as pdf
  show-trace [options] [trace]    Show trace viewer
  test                            Run tests with Playwright Test. Available in @playwright/test package.
  show-report                     Show Playwright Test HTML report. Available in @playwright/test package.
  help [command]                  display help for command
```

npx playwright --help

6

8. launch HTML reporter
7. launch Test runner
6. launch trace viewer
5. generate PDFs of page
1. open pages and emulate mobile contexts
2. generate test scripts from your interactions!
3. install browsers and dependencies
4. generate screenshots of page

USE PLAYWRIGHT CLI TO..

READ THE DOCS

playwright.dev/docs/test-cli

npx playwright test

Complete set of Playwright Test options is available in the configuration file. Following options can be passed to a command line and take a priority over the configuration file:

- `--headed`: Run tests in headed browsers. Useful for debugging.
- `--browser`: Run test in a specific browser. Available options are `"chromium"`, `"firefox"`, `"webkit"` or `"all"` to run tests in all three browsers at the same time.
- `--debug`: Run tests with Playwright Inspector. Shortcut for `PWDEBUG=1` environment variable and `--timeout=0 --maxFailures=1 --headed --workers=1` options
- `-c <file>` or `--config <file>`: Configuration file. If not passed, defaults to `playwright.config.ts` or `playwright.config.js` in the current directory.
- `-C <dir>` or `--config <dir>`: Directory with the tests to run without configuration file.
- `--forbid-only`: Whether to disallow `test.only`. Useful on CI.
- `-g <grep>` or `--grep <grep>`: Only run tests matching this regular expression. For example, this will run `'should add to cart'` when passed `-g="add to cart"`.
- `--grep-invert <grep>`: Only run tests **not** matching this regular expression. The opposite of `--grep`.
- `--global-timeout <number>`: Total timeout for the whole test run in milliseconds. By default, there is no global timeout.
- `--list`: List all the tests, but do not run them.
- `--max-failures <N>` or `-x`: Stop after the first N test failures. Passing `-x` stops after the first failure.
- `--output <dir>`: Directory for artifacts produced by tests, defaults to `test-results`.
- `--project <name>`: Only run tests from one of the specified projects. Defaults to running all projects defined in the configuration file.
- `--quiet`: Whether to suppress stdout and stderr from the tests.
- `--repeat-each <N>`: Run each test N times, defaults to one.
- `--reporter <reporter>`: Choose a reporter: minimalist `dot`, concise `line` or detailed `list`. See reporters for more information.
- `--retries <number>`: The maximum number of retries for flaky tests, defaults to zero (no retries).
- `--shard <shard>`: Shard tests and execute only selected shard, specified in the form `current/all`, 1-based, for example `3/5`.
- `--timeout <number>`: Maximum timeout in milliseconds for each test, defaults to 30 seconds.
- `--update-snapshots` or `-u`: Whether to update snapshots with actual results instead of comparing them. Use this when snapshot expectations have changed.
- `--workers <number>` or `-j <number>`: The maximum number of concurrent worker processes that run in parallel.

DEBUG

REPORTS

SHARDS!!

7

THE "playwright test" command will be the one you use most often!

playwright test --debug will launch the INSPECTOR tool!

npx playwright test

npx playwright show-report

8

THINK PARALLELIZATION

DID YOU KNOW "GREP" HELPS!

ONLY RUN TESTS IN SCRIPT FILE IF THEY MATCH THE CLI-given EXPRESSION!!

THE PLAYWRIGHT TEST RUNNER WORKS WITH A STATIC TEST CONFIGURATION FILE THAT PROVIDES TEST RUN CONTEXT

- PROJECTS
- RETRIES
- FIXTURES (use)
- TIMEOUT
- DIRECTORIES (various)
- METADATA

OR YOU CAN DYNAMICALLY CONFIG THIS VIA CLI!

```
(base) nitya@nityas-mbp ~ % npx playwright pdf --help
Usage: npx playwright pdf [options] <url> <filename>

save page as pdf

Options:
  --wait-for-selector <selector>  wait for given selector before saving as pdf
  --wait-for-timeout <timeout>    wait for given timeout in milliseconds before saving as pdf
  -b, --browser <browserType>     browser to use, one of cr, chromium, ff, firefox, wk, webkit (default: "chromium")
  --channel <channel>             Chromium distribution channel, "chrome", "chrome-beta", "msedge-dev", etc
  --color-scheme <scheme>         emulate preferred color scheme, "light" or "dark"
  --device <deviceName>           emulate device, for example  "iPhone 11"
  --geolocation <coordinates>     specify geolocation coordinates, for example "37.819722,-122.478611"
  --ignore-https-errors           ignore https errors
  --load-storage <filename>       load context storage state from the file, previously saved with --save-storage
  --lang <language>               specify language / locale, for example "en-GB"
  --proxy-server <proxy>          specify proxy server, for example "http://myproxy:3128" or "socks5://myproxy:8080"
  --save-storage <filename>       save context storage state at the end, for later use with --load-storage
  --save-trace <filename>         record a trace for the session and save it to a file
  --timezone <time zone>          time zone to emulate, for example "Europe/Rome"
  --timeout <timeout>             timeout for Playwright actions in milliseconds (default: "10000")
  --user-agent <ua string>        specify user agent string
  --viewport-size <size>          specify browser viewport size in pixels, for example "1280, 720"
  -h, --help                      display help for command
```

npx playwright pdf --help.

5

SAVE PAGE AS PDF CUSTOMIZE YOUR BROWSER, CONTEXT!

BUILT IN ABILITY TO WAIT FOR SELECTOR (RELIABILITY) OR TIMEOUT (STABILITY) TO SAVE!

SKETCHNOTES BY @NITYA

HTTPS://BIT.LY/LEARN-PLAYWRIGHT

Resources

Documentation
- https://playwright.dev/

Slides
https://github.com/PagelsR/Talks

Source | Issues | Discussion
- https://github.com/microsoft/playwright

Social Media
- https://aka.ms/playwright/twitter
- https://aka.ms/playwright/youtube

Microsoft Playwright Testing
- https://aka.ms/mpt/private-preview-blog

Test Automation with Playwright with Visual Guide Cheatsheet
- https://www.azurestaticwebapps.dev/blog/devtools-playwright

Thank you!