CLOUDFLARE

# Telescope

## Cross-Browser Test Agent

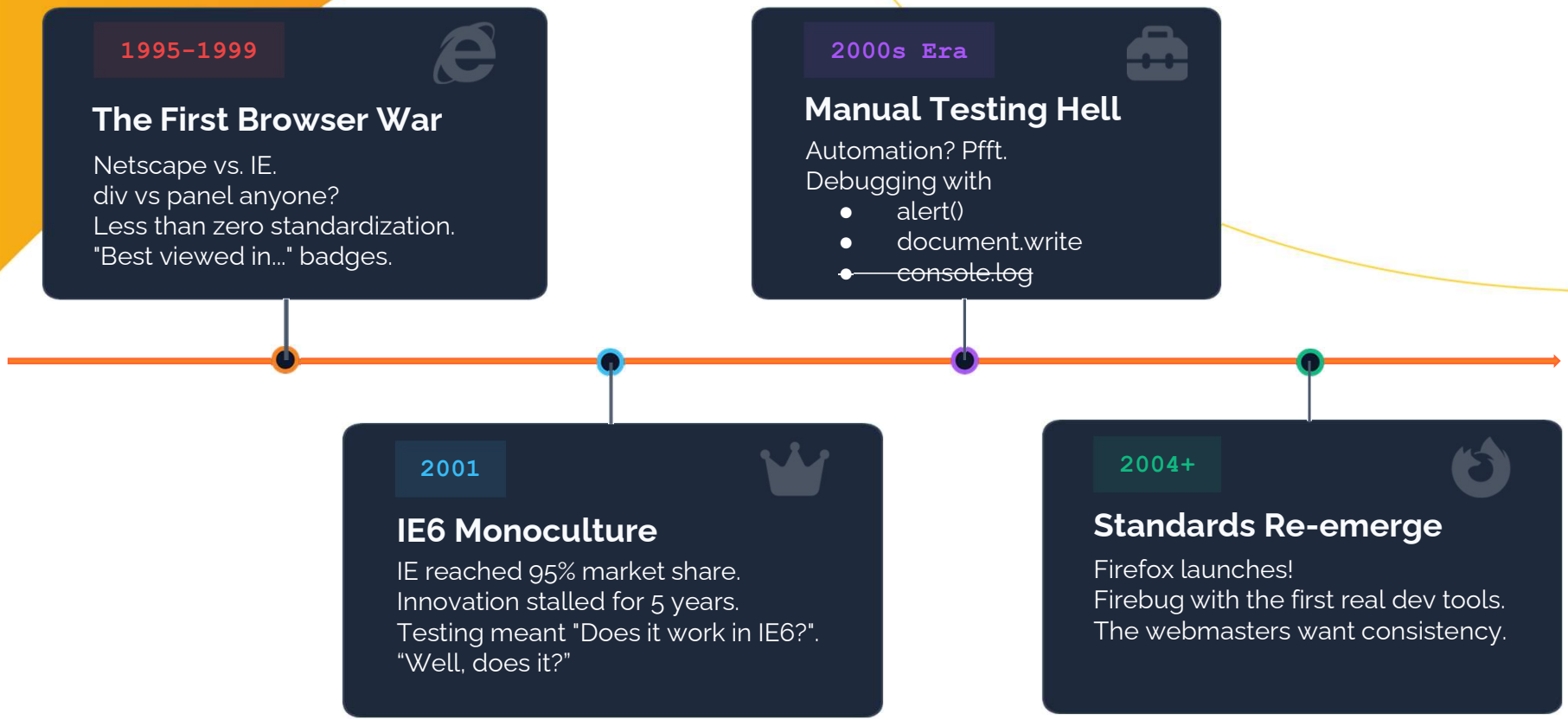
By Mike Kozicki

github.com/cloudflare/telescope

telescopetest.io

# The Early Days Browser Wars

CLOUDFLARE



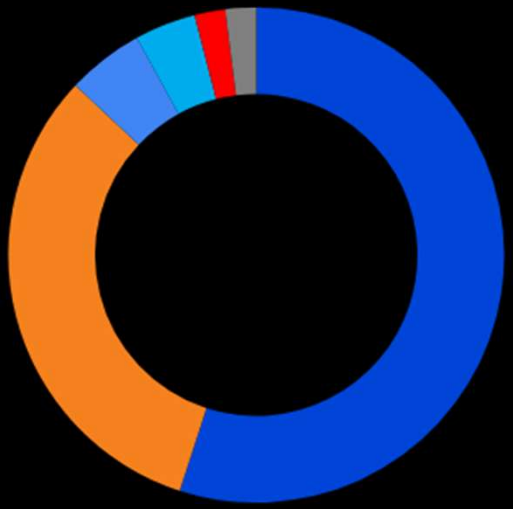
# Market Share Circa 2009

**Desktop Traffic**

**99.3%**

Internet usage was overwhelmingly tethered to desktop computers.


**Desktop Browser Share**




### IE Monoculture

Corporate environments and legacy apps mandated strict IE compatibility, often ignoring standards.

### Mobile? Niche.

With < 1% share, mobile performance testing was virtually non-existent in mainstream workflows.

DATA SOURCE: STATCOUNTER GLOBAL STATS

# WebPageTest - a performance revolution

01

### Open Source Origins

Created by Patrick Meenan at AOL and open-sourced in 2008, giving web developers access to top notch performance diagnostics from his basement.

02

### Real Browser Testing

Revolutionized the industry by moving away from simulated traffic to "synthetic" testing using real browser engines, revealing the actual user experience.

03

### The IE Standard

Initially focused heavily on Internet Explorer to address the primary bottleneck of the era, becoming the de-facto standard for web performance analysis.

04

### Acquisition

Acquired by Catchpoint in 2019. Catchpoint acquired by Logic Monitor in 2025. Future of WPT cloudy.

CLOUDFLARE

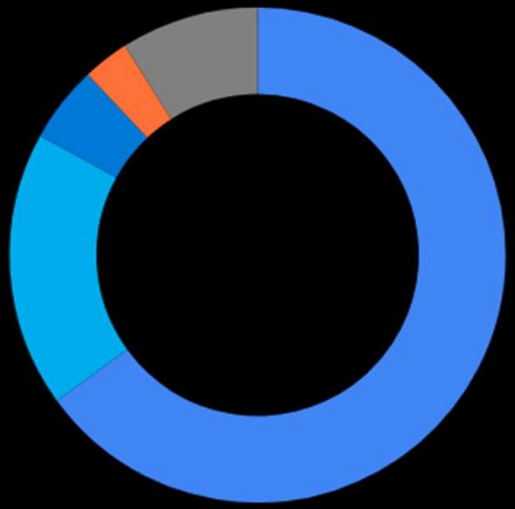
# The Current Landscape Circa 2026

**Mobile Traffic**

## 51.2%

Mobile devices have surpassed desktop, demanding rigorous performance testing on constrained hardware.


**Global Browser Share**



- Chrome (65%)
- Safari (18%)
- Edge (5%)
- Firefox (3%)

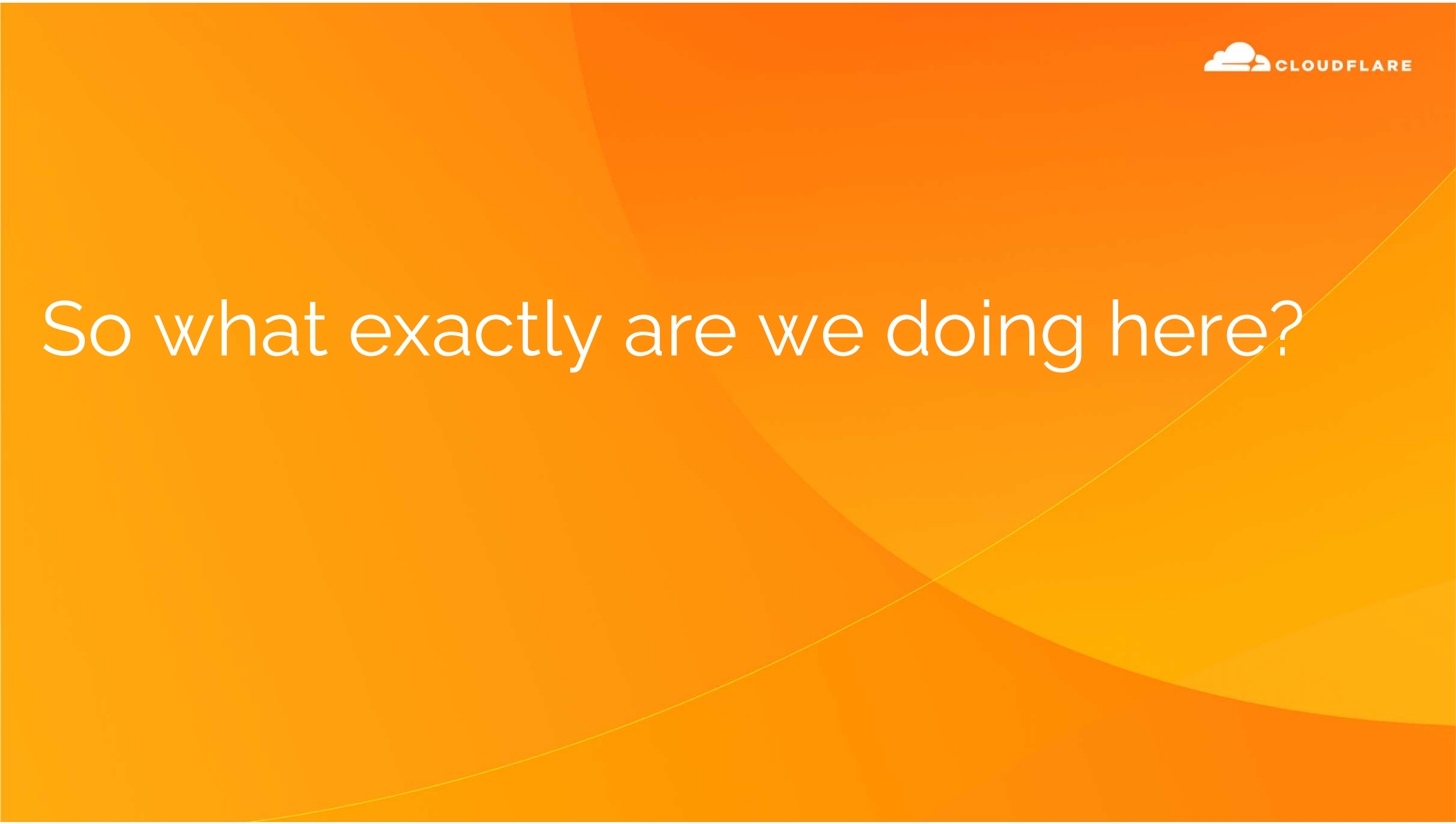
### Safari's Power

Safari's dominance on high-value iOS devices makes WebKit a critical target for performance optimization.

### Multi-Engine

Testing across Chromium, WebKit, and Gecko is essential to catch engine-specific rendering and JS issues.

DATA SOURCE: STATCOUNTER 2025


CLOUDFLARE
So what exactly are we doing here?

# Telescope

Cloudflare's
fully supported,
**community-driven**, cross-
browser,
test agen
*(soon with a full UI)*

github.com/cloudflare/telescope

telescopetest.io

# Why now?

The Gap

### Legacy Tools are Aging Out

Traditional tools built for the desktop-first era struggle with modern SPAs, hydration, and complex authentication flows.

### Interaction Readiness

Mobile users are impatient. Focus on improving Interaction to Next Paint (INP) by reducing long tasks that delay tap responses.

### Shift to CI/CD

Teams need deterministic agents that can block regressions in the pipeline, not just monitor them in production.

### Viewport Constraints

Test with `--width 360` to ensure responsive layouts adapt correctly. Metrics like CLS often spike on smaller screens due to ad injection or image shifts.

**4X Slowdown**

CPU Throttling

Essential for mimicking mid-tier Android devices

**360px**

Narrow Viewport

Validates tap targets and prevents horizontal scroll

**3G**

Network Profile

Exposes heavy payload issues and latency

# Browser Support

01

### Chromium Family

Native support for Google Chrome, Microsoft Edge, Chrome Beta, and Canary builds for the majority of users.

02

### Gecko Engine

Full Firefox support with advanced configuration capabilities. T

03

### WebKit (Safari)

Uses Playwright's WebKit build to emulate Safari behavior in Apple's ecosystem for macOS and iOS-like results.

# Quick start

```
user@telescope:~/tests

$ git clone https://github.com/cloudflaretelescope
$ cd telescope
$ npm install
$ npm run build

# Run your first test
$ npx . -u https://example.com
```

### Quick Start

The simplest command requires only a URL (`-u`). By default, it runs on Chrome. Use `-b` to switch browsers.

### Discovery

Use the --help flag at any time to see the full list of arguments, defaults, and descriptions.

### Troubleshooting

Use --debug to see Playwright internals or increase --timeout for heavy pages (default is 30s).

# Disable JavaScript

```
user@telescope:~/tests

# Verify core content delivery & HTML performance
$ npx . -u https://example.com --disableJS

# Basic usage: load site without scripts
$ npx . -u https://example.com -b firefox --disableJS

# Mobile fallback check (no JS + narrow viewport)
$ npx . -u https://example.com --width 375 --disableJS
```

### Progressive Enhancement

Ensure your site remains functional and readable even if JavaScript fails to load, is blocked by extensions, or times out.

### Core Content Validation

Verify that critical CSS and HTML content are delivered and rendered correctly before hydration takes place.

### HTML-First Performance

Isolate the performance cost of your JavaScript bundles by measuring the raw HTML/CSS rendering speed (FCP).

# Viewport & Filmstrip Customization

```
user@telescope:~/tests

# Test Mobile Portrait (e.g. iPhone SE)
$ npx . -u https://example.com --width 375 --height 667

# Standard Laptop Viewport
$ npx . -u https://example.com --width 1366 --height 768

# Tablet / iPad Portrait
$ npx . -u https://example.com --width 768 --height 1024

# Capture high-fps filmstrip
$ npx . -u https://example.com --frameRate 2
```

### Responsive Breakpoints

Verify layout shifts and element visibility across critical mobile and desktop breakpoints.

### Pixel Precision

Define exact dimensions to simulate specific hardware constraints or windowed environments.

### Filmstrip Control

Adjust --frameRate to capture granular loading states for visual analysis.

# Cookies and Headers

```
user@telescope:~/tests

# Set a single custom cookie
$ npx . -u https://example.com -c '{ "name":
"session_id", "value": "xyz123" }'

# Set multiple cookies (Array of objects)
$ npx . -u https://example.com -c '[{"name": "user",
"value": "test"}, {"name": "geo", "value": "us"}]'

# Inject custom Headers (e.g., Auth tokens)
$ npx . -u https://example.com -h '{"Authorization":
"Bearer token_abc123"}'

# Scoping cookies to specific paths
$ npx . -u https://example.com -c '{ "name": "beta",
"value": "true", "path": "/app" }'
```

### Cookie Injection

Use -c or --cookies. Accepts a JSON object or an array of objects. Defaults to the test URL domain if not specified.

### Custom Headers

Use -h or --headers. Essential for testing behind API gateways, WAFs, or passing auth tokens.

### Use Case: Auth State

Avoid logging in via UI during the test. Inject the session cookie directly to start the test in a "Logged In" state instantly.

# Network Throttling

```
user@telescope:~/tests

# Simulate a standard 4G Mobile connection
$ npx . -u https://example.com --connectionType 4g

# Test behavior on slow 3G (Emerging Markets)
$ npx . -u https://example.com --connectionType 3gslow

# High-speed Fiber/Cable (Baseline)
$ npx . -u https://example.com --connectionType cable
```

**Mobile Profiles**

Includes 4g, 3gfast, 3gslow, and 2g. Affects both latency and throughput.

**Desktop Profiles**

Includes cable, dsl, and fios. Typically lower latency.

**Why Throttling Matters?**

Most dev environments are on fast Wi-Fi. Throttling exposes race conditions and heavy asset loading issues.

# CPU Throttling

```
user@telescope:~/tests

# Simulate a mid-range mobile device (4x slower)
$ npx . -u https://example.com --cpuThrottle 4

# Combine with mobile viewport & network for some real-
world
$ npx . -u https://example.com --cpuThrottle 4 --
connectionType 3g --width 375

# Stress test: Extreme low-end device (6x slower)
$ npx . -u https://example.com --cpuThrottle 6
```

### The Throttle Factor

Use –cpuThrottle. The browser's main thread execution will run approximately X times slower than your host machine.

### Mobile Simulation

Factor 4x is the industry standard (Lighthouse default) for simulating mid-range Android devices on powerful dev laptops.

### Why it Matters

Reveals main-thread bottlenecks, JavaScript hydration delays, and UI unresponsiveness that fast CPUs hide.

# Browser Flags and overrides

```
user@telescope:~/tests

# Pass Chromium CLI arguments (Chrome/Edge)
$ npx . -u https://example.com --flags 'disable-
features=InterestCohortAPI'

# Set Firefox specific user preferences
$ npx . -u https://example.com -b firefox --firefoxPrefs
'{"network.trr.mode":2}'

# Override hostname resolution (DNS spoofing)
$ npx . -u https://example.com --overrideHost
'{"www.example.com":"uat.example.com"}'
```

### Chromium Flags

Pass raw command-line switches to Chrome or Edge instances. Useful for enabling experimental features or disabling specific APIs.

### Firefox Preferences

Inject custom values into the about:config settings for the Gecko engine, allowing granular control over browser behavior.

### Host Overrides

Map production domains to staging IPs or local environments without modifying your machine's hosts file. Essential for pre-prod testing.

# Domain Blocking

```
user@telescope:~/tests

# Block specific domains completely
$ npx . -u https://example.com --blockDomains 'ads.com,tracker.io'

# Block by URL substring match (more aggressive)
$ npx . -u https://example.com --block 'gtm.js,recaptcha'
```

**--blockDomains**

Takes a comma-separated list of fully qualified domain names. Useful for isolating specific vendors like ads.google.com.

**--block (Substring)**

Blocks any request URL containing the string. Powerful for blocking file types or patterns, e.g., .mp4 or tracking.

**Impact Analysis**

Run tests twice: once with all scripts, once with blockers. The difference in LCP and TBT reveals the "cost" of third parties.

# Device Type

```
user@telescope:~/tests

# Emulate the specified device.
$ npx . -u https://example.com --deviceType "Galaxy S24"

# Override the "Galaxy S24" height, keeps the width
$ npx . -u https://example.com --deviceType "Galaxy S24" \
  --height 120

# Override the "Galaxy S24" height and width
$ npx . -u https://example.com --deviceType "Galaxy S24" \
  --height 120 - width 200
```

### Many Devices

Emulate all supported Playwright devices. Available at Playwright Devices.

### Override Height and/or Width

Override the default device height or width for further customization

### Parity with Playwright version

Updated Playwright allows immediate access to new devices.

# Directory processing

```
user@telescope:~/tests

# ZIP, upload to the provided URL
$ npx . -u https://example.com --zip \
   —uploadUrl https://example.com

# Upload zip to provided URL, delete Zip file
$ npx . -u https://example.com  \
   —uploadUrl https://example.com
```

**Zlp files**

Create a zip archive of results files to save some space

**Upload**

Provide a URL endpoint and POST the results zip file.

**Save space on your machine**

Run a cleanup job to remove old unarchived  or uploaded tests.

# Data Exports & Reports

```
user@telescope:~/tests

# ZIP, generate html report and open
$ npx . -u https://example.com --zip -html --openHtml

# Generate an email ready page
$ npm run visualize :test_id

# View generated artifacts in results folder
$ ls -R ./results/:test_id
config.json
<text_id>.zip
console.json
engine.jpg
index.html
metrics.json
pageload.har
resources.json
screenshot.png
video.webm
visual_report.html
./filmstrip
Frame_1366x768_1.jpg          frame_1366x768_3.jpg
               frame_1366x768_3.jpg
```

### HAR Export

Standard HTTP Archive format. Contains full network waterfall data compatible with external viewers.

### HTML Reports

Self-contained visual reports including filmstrips, Core Web Vitals, and resource breakdowns.

### Portable Artifacts

Use --zip to bundle video, screenshots, and logs for CI/CD artifact storage.

CLOUDFLARE

# Reporting

## JSON Metrics

Detailed timing data and resource waterfall information for programmatic parsing.

`metrics.json`

`resources.json`

`console.json`

## HAR File

Full HTTP Archive format recording of all network interactions during page load.

**Standard Format**

Compatible with HAR Viewers

## Visuals & Filmstrip

Final state screenshots plus a frame-by-frame video record of the user experience.

`video.mp4` `/filmstrip/`

## HTML Report

Interactive, shareable dashboard summarizing all metrics and visual data in one file.

**Shareable**

Auto-open with --openHtml

## CLI Utilities

Built-in helpers for managing results.

CLOUDFLARE

# Metrics Collection

01

### Largest Contentful Paint (LCP)

Measures loading performance. It marks the point in the page load timeline when the page's main content has likely loaded.

02

### Cumulative Layout Shift (CLS)

Quantifies visual stability. It measures the total of all individual layout shift scores for every unexpected layout shift that occurs.

03

### Interaction to Next Paint (INP)

Assesses overall responsiveness. It measures the latency of every tap, click, and keyboard interaction throughout the page lifespan.

04

### Programmatic Access

All metrics are automatically exported to metrics.json and accessible via the API for automated regression checks.

# Sample use cases

github.com/cloudflare/telescope

telescopetest.io

# CI/CD Integration

.github/workflows/perf-test.yml

```
ame:Performance Audit
on:[push]
jobs:telescope-check:runs-on:ubuntu-latest
steps:
  - name:Checkout Code
  uses:actions/checkout@v3
    - name:Install & Run Telescope
  run: npm ci
    npx playwright install# Run test and generate report/zip
    npx . -u https://example.com -b chrome --html --zip
  - name:Upload Artifacts
  uses:actions/upload-artifact@v3
  with:name:perf-results
  path:results/*.zip
```

**GitHub Actions Ready**

Runs seamlessly on standard runners. Just install dependencies, Playwright browsers, and execute.

**Artifact Retention**

Use --zip to package results. Upload them as build artifacts to debug failed runs later.

**Fail on Regression**

Parse metrics.json in a subsequent step to fail the build if LCP exceeds your budget.

# 3rd-Party Audits

audit@telescope:~/performance-audit

```
# 1. Establish Baseline (Control Run)
$ npx . -u https://news.example.com

# 2. Block Ads & Analytics (Experiment Run)
$ npx . -u https://example.com --blockDomains \
"doubleclick.net,ga.com"

# 3. Block Specific Tag Manager Scripts
$ npx . -u https://news.example.com --block "gtm.js,fbevents.js"
```

**A/B Comparison**

Compare metrics between runs. Large deltas indicate 3rd-parties blocking the main thread.

**Blocking Strategies**

Use --blockDomains for entire vendors (e.g., ads) or --block for specific file substrings.

**Waterfall Diagnosis**

Analyze the pageload.har export to see how many requests and bytes were saved.

# Regression Testing

### Establish Baselines

Define performance standards per browser engine and network profile to understand your "normal".

### Fail on Thresholds

Automatically block builds if metric deltas (e.g., LCP > 200ms increase) exceed defined limits.

### Diagnose with Artifacts

Store HARs, videos, and profiles to immediately identify why a regression occurred.

# Performance Benchmarking

### Surface Bottlenecks

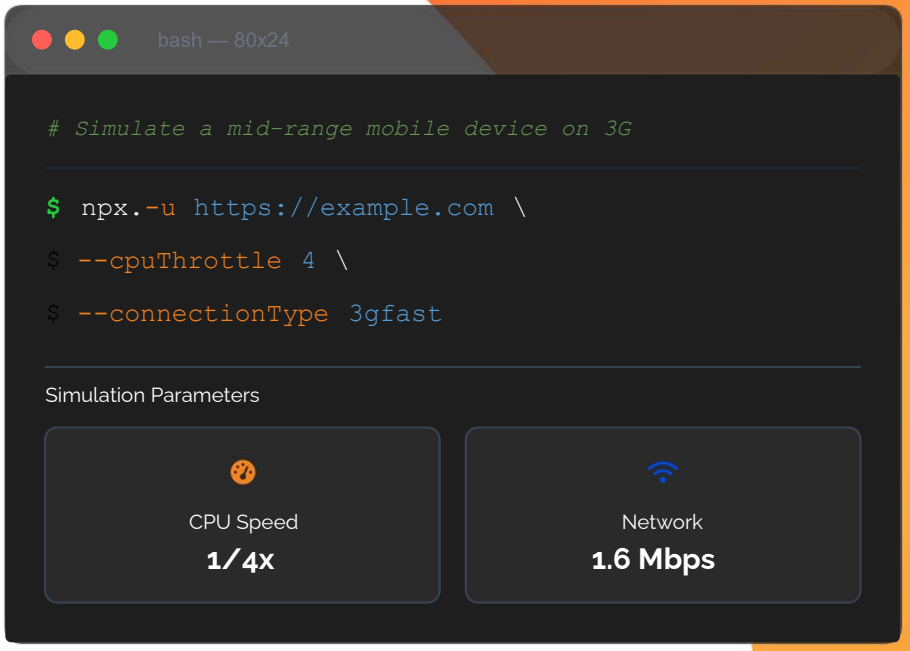
Don't test on powerful dev machines alone. Use CPU throttling to simulate lower-end hardware and expose main-thread blocking scripts.

### Focus Areas

Identify heavy costs in Script Execution, Image Decoding, and Layout Recalculation that cause jank.

### Realistic Conditions

Model real-world user environments by combining network constraints (3G/4G) with CPU throttling factors.




Higher throttle values = slower simulated device performance

# telescopetest-io

github.com/cloudflare/telescope

telescopetest.io

# How we are building it

01

### Has to be fast

Astro web framework. SSR and vanilla Javascript.
https://astro.build

02

### OSS all the way

Cross browser and open to community and all the input.

03

### Up-to-date always

Maintaining upgrades for Playwright, expanding the visuals, and keeping the bugs out.

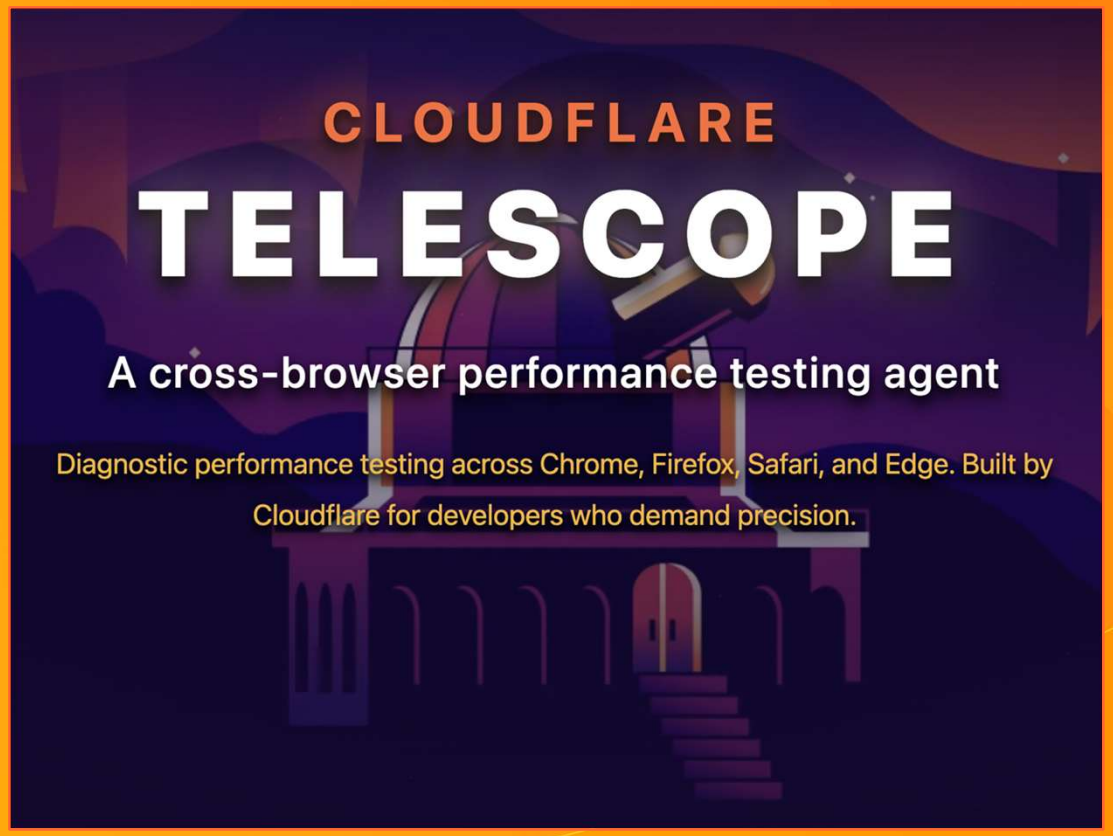
04

### Dedication

Dedicated work from the Cloudflare Observatory team to ensure progress continues.

CLOUDFLARE

# telescopetest.io




**Start it up**
Start a Basic or Advanced test from the UI. View previous test run list. Upload you zipped test data.

**Results**
Overview, Video & Filmstrip, Waterfall, Console, Resources, Metrics, Bottlenecks, and Config.

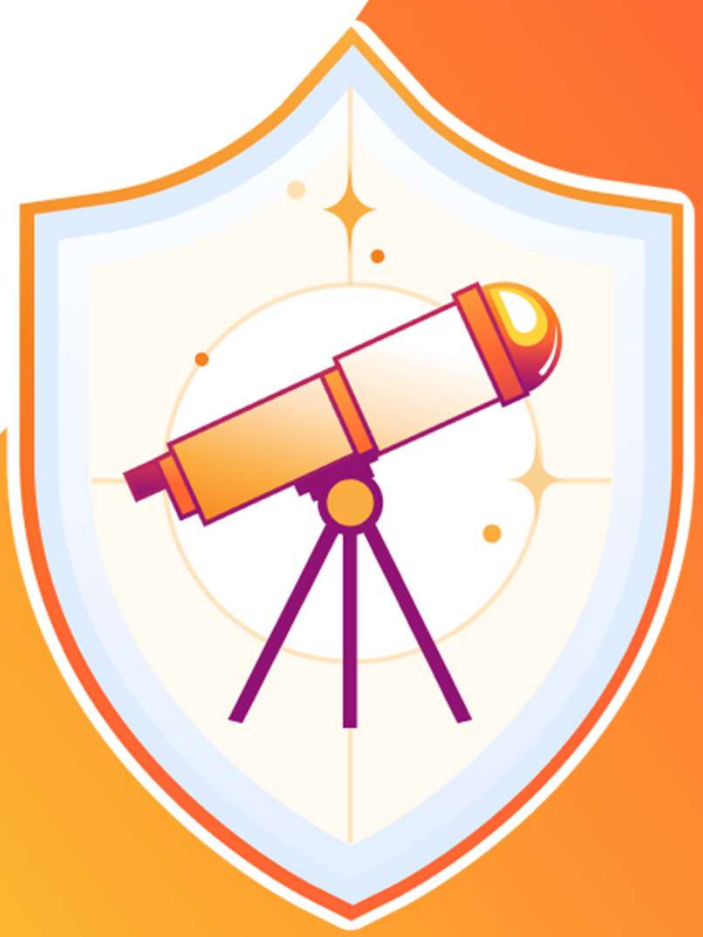
**Astro**
Fast web framework supported by Cloudflare.

# Demo time

# What can you do?

1. Use Telescope
2. Submit issues and PRs
3. Tell us what **you** need

*we're listening.*



github.com/cloudflare/telescope

telescopetest.io

CLOUDFLARE

# Thank you


1 888 99 FLARE

enterprise@cloudflare.com

cloudflare.com