Measuring Performance with **WebDriver**

Laba diena!

Christian Bromann

Senior Lead
Software Engineer at
Sauce Labs

christian-bromann

@bromann

53%

of mobile site visits are abandoned if pages take longer than 3 seconds to load. (Study by DoubleClick owned by Google)

3034 kb

is the average web page size in 2018, trend: increasing
(https://speedcurve.com/blog/web-performance-page-bloat/)

3.21 s

Is the average load of a webpage (Pingdom/2018)

“How fast your website loads is critical but often a completely ignored element in any online business and that includes search marketing and search engine optimisation.”

—Google


#perfmatters

“Performance stands out like a ton of diamonds. Nonperformance can always be explained away.”

—Harold S. Geneen.

Browser Performance

How **fast** does my application load ?

DevTools - saucecon.com/

Elements | Console | Network | Performance | Sources | Application | Memory | Security | Audits | JavaScript Profiler | Screener Recorder | AdBlock

top | Filter | Default levels

```
> window.performance.getEntriesByType("navigation")[0]
< PerformanceNavigationTiming {unloadEventStart: 0, unloadEventEnd: 0, domInteractive: 3169.7149999672547, domContentLoadedEventStart: 3169.804999954067, domContentLoadedEventEnd: 3272.965000011027, …}
    connectEnd: 410.75000003911555
    connectStart: 74.33999993372709
    decodedBodySize: 227007
    domComplete: 8886.48500002455
    domContentLoadedEventEnd: 3272.965000011027
    domContentLoadedEventStart: 3169.804999954067
    domInteractive: 3169.7149999672547
    domainLookupEnd: 74.33999993372709
    domainLookupStart: 74.33500001206994
    duration: 8931.220000027679
    encodedBodySize: 59913
    entryType: "navigation"
    fetchStart: 10.454999981448054
    initiatorType: "navigation"
    loadEventEnd: 8931.220000027679
    loadEventStart: 8886.509999982081
    name: "https://saucecon.com/"
    nextHopProtocol: "h2"
    redirectCount: 0
    redirectEnd: 0
    redirectStart: 0
    requestStart: 412.92000003159046
    responseEnd: 2207.320000450015
    responseStart: 2080.4449999704957
    secureConnectionStart: 0
  ▶ serverTiming: []
    startTime: 0
    transferSize: 60330
    type: "navigate"
    unloadEventEnd: 0
    unloadEventStart: 0
    workerStart: 0
  ▶ __proto__: PerformanceNavigationTiming
>
```

Console | What's New | Remote devices ×

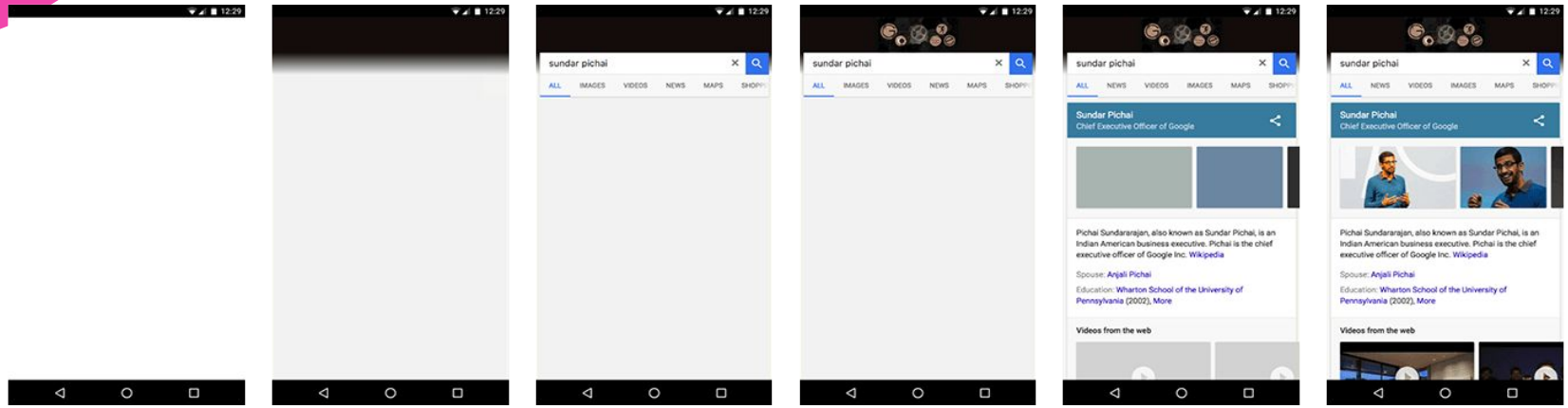
responseStart

-

fetchStart

=

time to first byte (TTFB)



First Paint (FP) | First Contentful Paint (FCP) | First Meaningful Paint (FMP) | Time To Interactive (TTI)

First Paint (FP)
first render to the screen

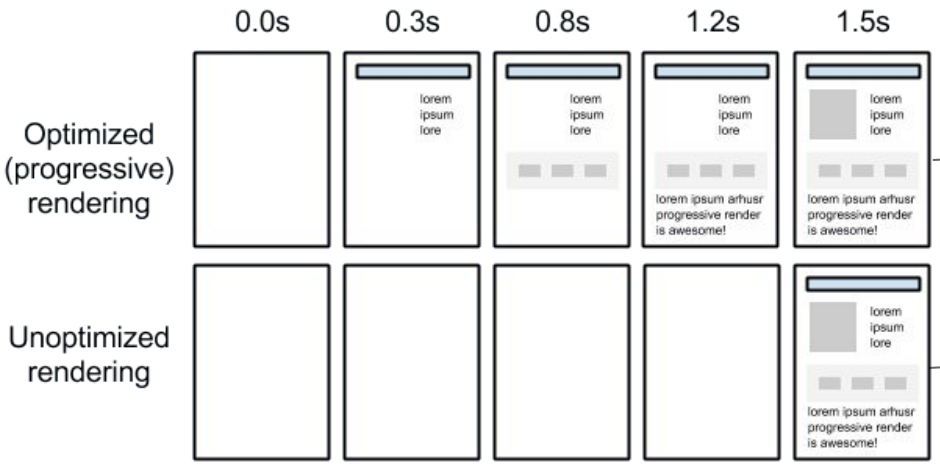
First Contentful Paint (FCP)
is triggered when any content is painted – i.e. something defined in the DOM

First Meaningful Paint (FMP)
measures how long it takes for the most meaningful content to be fully rendered on the site.

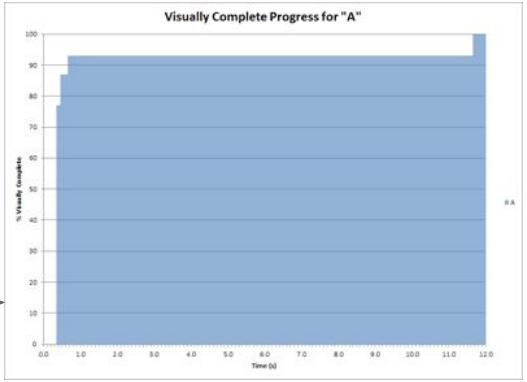
Time To Interactive (TTI)
number of seconds from the time the navigation started until the layout is stabilized

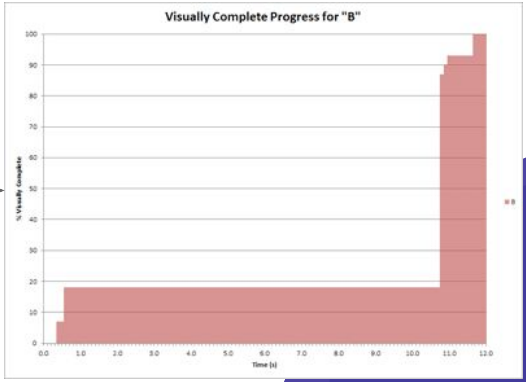
Speed Index

computes an overall score for how quickly the content painted



(Source: Google)





Other Metric Types

Milestone Based

Describing a duration between two events

Score Based

Describing performance based on a score

Resource Based

Describing certain resource limits

Are **all** these metrics important?

Yes!

Mapping Metrics to **User Experience**!

Is it happening?

Did the navigation start successfully? Has the server responded?

Is it useful?

Has enough content rendered that users can engage with it?

Is it usable?

Can users interact with the page, or is it still busy loading?

Is it delightful?

Are the interactions smooth and natural, free of lag and jank?

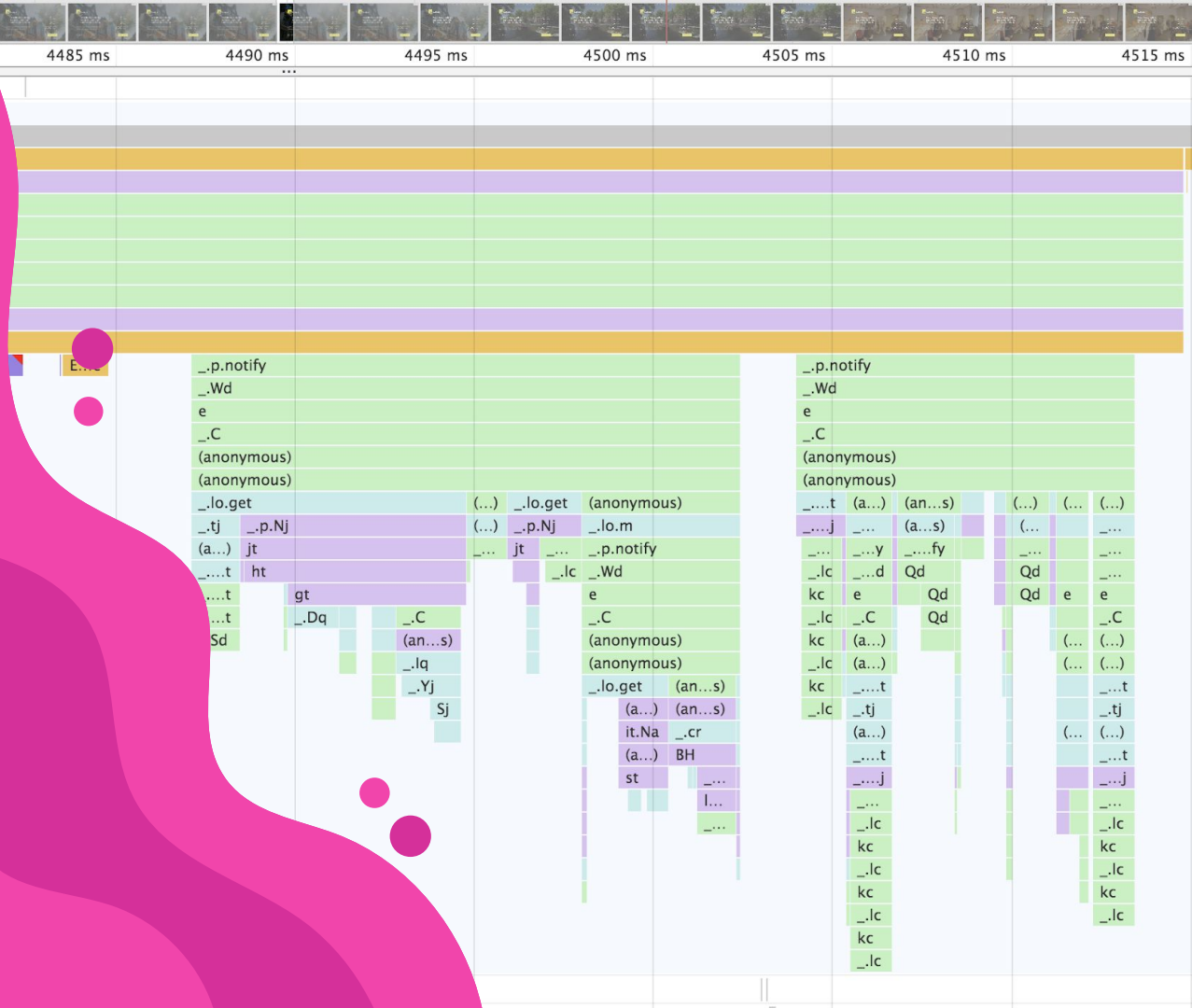
https://developers.google.com/web/fundamentals/performance/user-centric-performance-metrics

“Fast forward to today and we see that window.onload doesn’t reflect the user perception as well as it once did.”

—Steve Souders

Browser Tracing

How the **browser** knows about all this?



DevTools - events.linuxfoundation.org/events/node-js-interactive-2018/

Elements Network Performance Console Memory Application Sources Security Audits AdBlock

2 1

(no recordings) Screenshots Memory

Start profiling and reload page ⌘ ⇧ E

Click the record button ● or hit ⌘ **E** to start a new recording.

Click the reload button ↻ or hit ⌘ ⇧ **E** to record the page load.

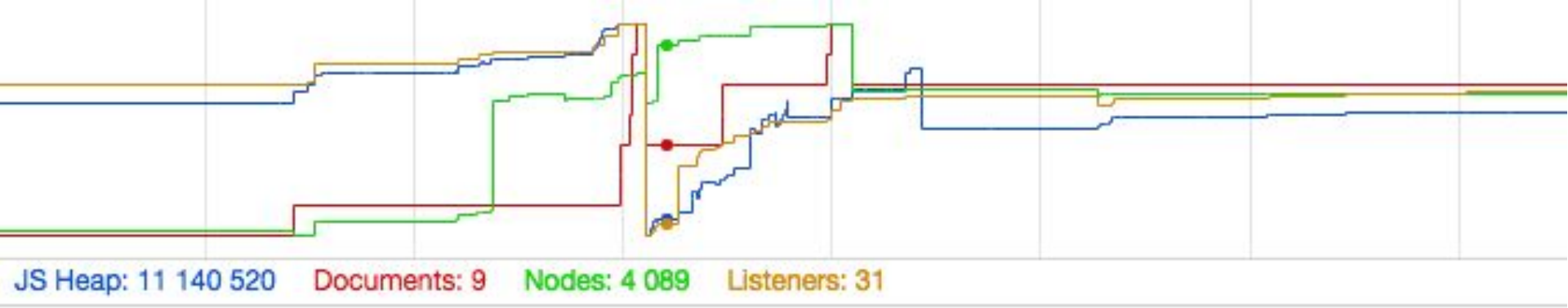
After recording, select an area of interest in the overview by dragging.
Then, zoom and pan the timeline with the mousewheel or **WASD** keys.
Learn more

- Contains a list of events from different types that happened during the capturing process, e.g.
    - Duration Events (B - begin, E - end)
    - Complete Events (x)
    - Instant Events (i)
    - Counter Events (C)
    - Sample events (P)
    - Metadata Events (M)
    - Memory Dump Events (V - global, v - process)
    - Other... (see Trace Event Format)

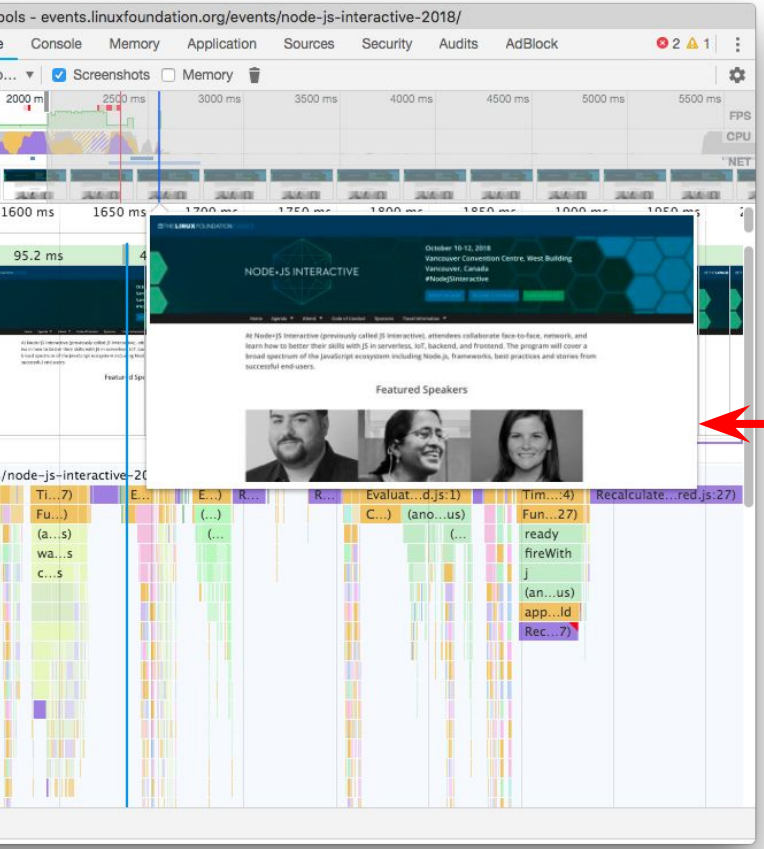
Trace data representations can be processed by a Trace Viewer tool like DevTools or Catapult

Event Descriptions:

```
{
  "name": "myName",
  "cat": "category.list",
  "ph": "B",
  "ts": 12345,
  "pid": 123,
  "tid": 456,
  "args": {
    "someArg": 1,
    "anotherArg": {
      "value": "my value"
    }
  }
}
```

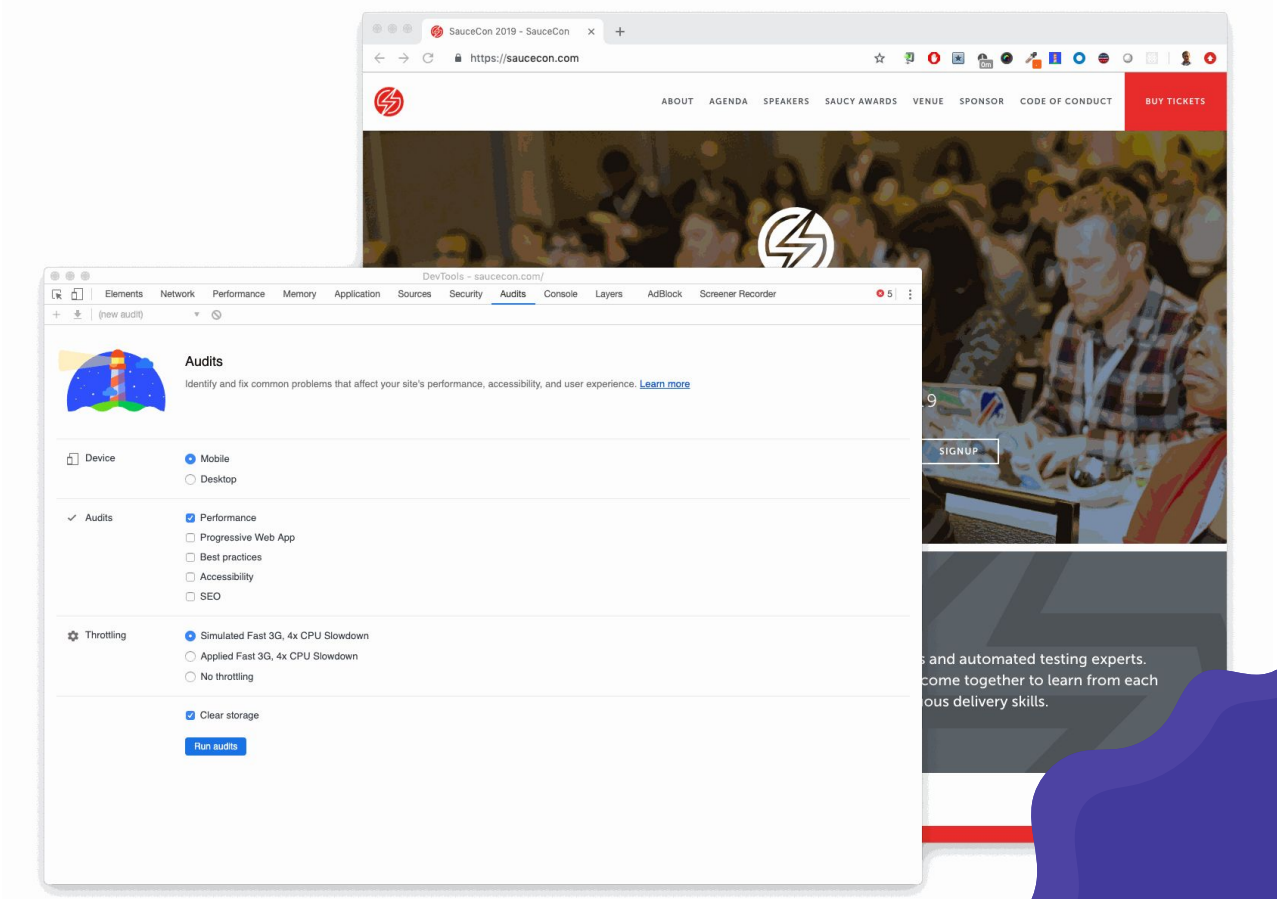



```
{
  "pid": 41316,
  "tid": 775,
  "ts": 170385299237,
  "ph": "I",
  "cat": "devtools.timeline",
  "name": "UpdateCounters",
  "args": {
    "data": {
      "jsEventListeners": 31,
      "nodes": 4089,
      "documents": 9,
      "jsHeapSizeUsed": 11140520
    }
  },
  "tts": 20811400,
  "s": "t"
}
```




```
{
  "pid": 579,
  "tid": 775,
  "ts": 170383426118,
  "ph": "O",
  "cat": "disabled-by-default-devtools.screenshot",
  "name": "Screenshot",
  "args": {
    "snapshot": "..."
  },
  "tts": 2879188825,
  "id": "0x1"
}
```

Google Lighthouse



www.webpagetest.org/result/190419_XS_9ea383e574eaeb9122ad90f5fa02c3d0/

Login | Register | Login with Google

WEBPAGETEST

HOME | TEST RESULT | TEST HISTORY | FORUMS | DOCUMENTATION | ABOUT

Need help improving?

Web Page Performance Test for

https://saucecon.com

From: Dulles, VA - **Moto G4 - Chrome - 3G**
4/19/2019, 6:13:55 PM

F	A	F	A	B	✓
First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN

Summary | Details | Performance Review | Content Breakdown | Domains | Processing Breakdown | Screenshot | Image Analysis | Request Map

Tester: MotoG4_13-192.168.1.113
First View only
Test runs: **3**
Re-run the test

Raw page data - Raw object data
Export HTTP Archive (.har)
View Test Log

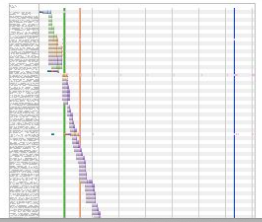
Performance Results (Median Run)

							Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Last Painted Hero	First Interactive (beta)	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 2)	70.751s	4.187s	11.888s	19.850s	71.023s	16.885s	70.751s	182	6,299 KB	77.575s	204	6,687 KB	$$$$$

Plot Full Results

Test Results

Run 1:

	Waterfall	Screenshot	Video
			

WebDriver

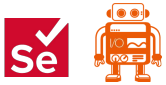
How **browser** get automated today?

6.5 List of Endpoints

The following ***table of endpoints*** lists the method and URI template for each endpoint node command. Extension commands are implicitly appended to this table.

Method	URI Template	Command
POST	/session	New Session
DELETE	/session/{*session id*}	Delete Session
GET	/status	Status
GET	/session/{*session id*}/timeouts	Get Timeouts
POST	/session/{*session id*}/timeouts	Set Timeouts
POST	/session/{*session id*}/url	Navigate To
GET	/session/{*session id*}/url	Get Current URL
POST	/session/{*session id*}/back	Back
POST	/session/{*session id*}/forward	Forward
POST	/session/{*session id*}/refresh	Refresh
GET	/session/{*session id*}/title	Get Title

```
const elem = $("#myElem")
elem.click()
```



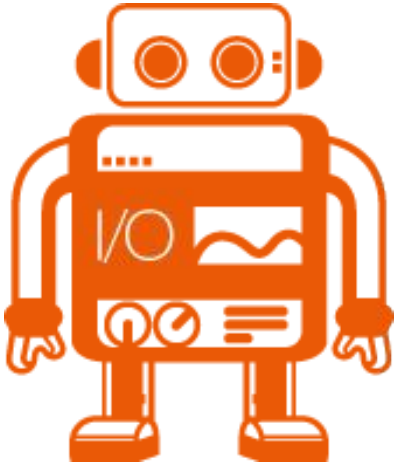
cypress.io

HTTP

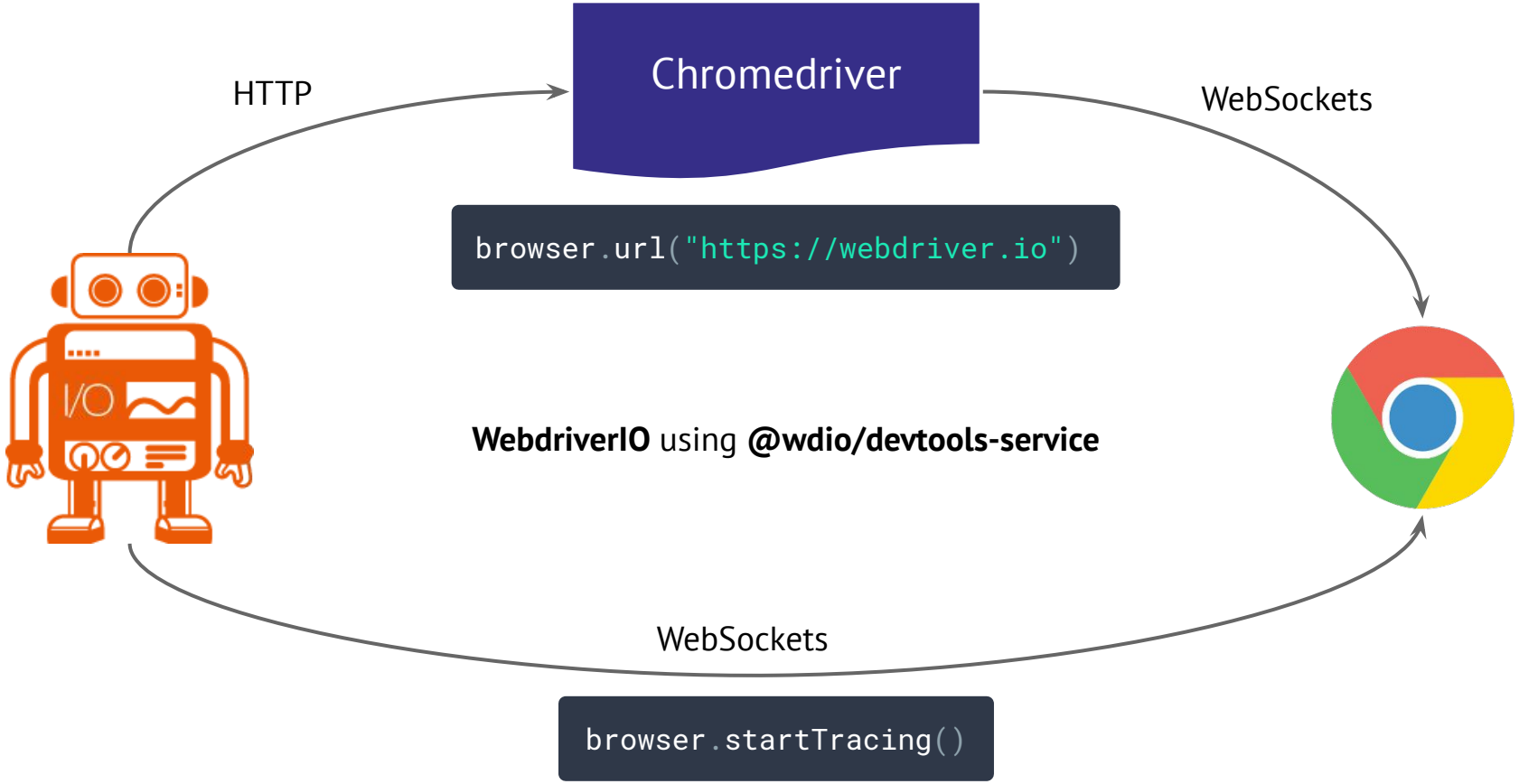
Chromedriver
Geckodriver
IEDriver
EdgeDriver
SafariDriver

Appium
Selendroid
WebDriverAgent

Selenium Grid

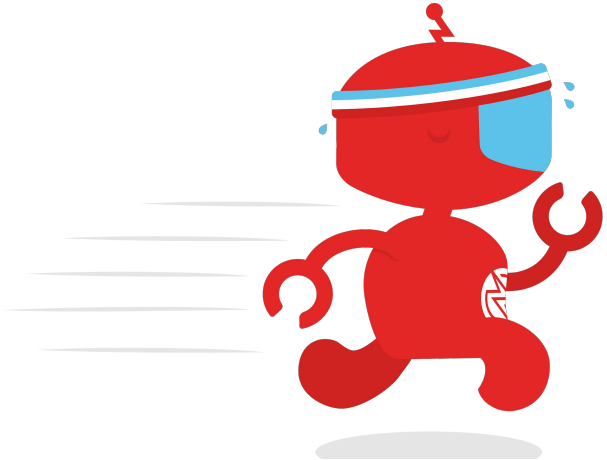


Webdriver.io


Chromedriver
HTTP
WebSockets
browser.url("https://webdriver.io")
WebdriverIO using @wdio/devtools-service
WebSockets
browser.startTracing()

DEMO

Capture Performance Data
with **WebdriverIO**



Testing Performance

Using Sauce Labs new **Performance** Feature

```
const elem = $("#myElem")
elem.click()
```

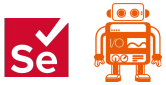
HTTP

Chromedriver
Geckodriver
IEDriver
EdgeDriver
SafariDriver

Appium
Selendroid
WebDriverAgent

Selenium Grid

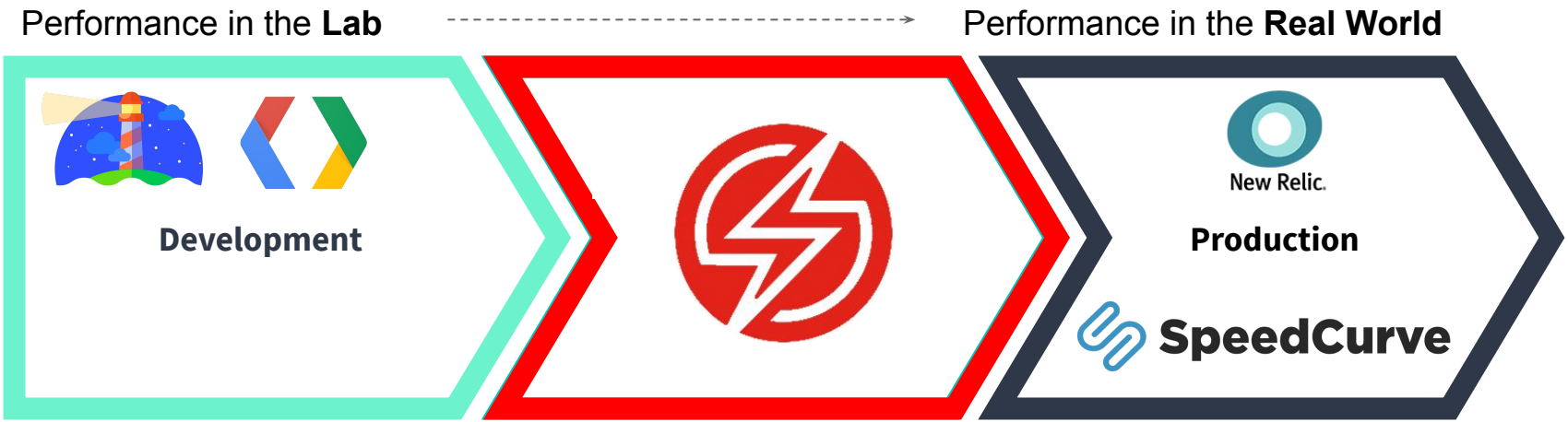
```
const elem = $("#myElem")
elem.click()
```



HTTP

Shift Testing To The Left

Existing Solutions



SPEEDO

```
2. christianbromann@SL-1122: ~ (zsh)
➜ ~ speedo run https://www.testcon.lt/
```

Install it!

```
$ npm install -g speedo
```

Run it!

```
$ speedo run https://site.com
```

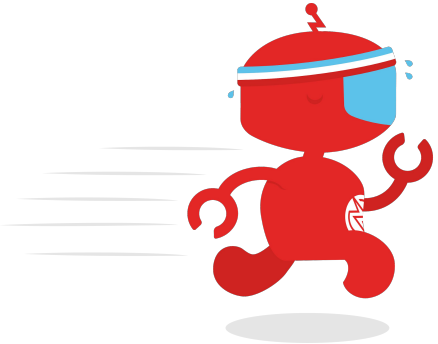
DEMO

Capture Performance Data with Speedo

```
import { remote } from 'webdriverio';

let browser

(async () => {
    browser = await remote({
        user: process.env.SAUCE_USERNAME,
        key: process.env.SAUCE_ACCESS_KEY,
        capabilities: {
            browserName: 'chrome',
            platformName: 'Windows 10',
            browserVersion: 'latest',
            'sauce:options': {
                extendedDebugging: true,
                capturePerformance: true,
                name: “Performance Test”
            }
        }
    })

    await browser.url('https://www.instagram.com/accounts/login')

    const username = await browser.$('input[name="username"]')
    await username.setValue('performancetestaccount')

    const password = await browser.$('input[name="password"]')
    await password.setValue('testpass')

    const submitBtn = await browser.$('button[type="submit"]')
    await submitBtn.click()

    await browser.deleteSession()
})().catch(async (e) => {
    console.error(e)
    await browser.deleteSession()
})
```

Check Performance for Instagram Login

```
$ speedo analyze “Performance Test” \
  -p https://www.instagram.com/ \
  --all
```



Ready For CI/CD

Speedo was build to run within your continuous integration pipeline!

```
pipeline {
    agent none
    stages {
        stage('Linting') {
            ...
        }
        stage('Unit Tests') {
            ...
        }
        stage('Functional Tests') {
            ...
        }
        stage('Performance Tests') {
            agent {
                docker { image 'saucelabs/speedo' }
            }
            steps {
                sh 'speedo run https://google.com -u
${SAUCE_USERNAME} -k ${SAUCE_ACCESS_KEY} -b ${BUILD_NUMBER}'
            }
        }
    }
}
```

```
variables:
  SPEEDO_IMAGE: saucelabs/speedo

stages:
  - lint
  - test
  - performance
  - deploy

# ...

# run performance tests
performance:
  stage: performance
  image: $SPEEDO_IMAGE
  script:
    - speedo run https://google.com -u $SAUCE_USERNAME
-k $SAUCE_ACCESS_KEY -b $BUILD_NUMBER

# ...
```

Test Performance within a WebDriver test

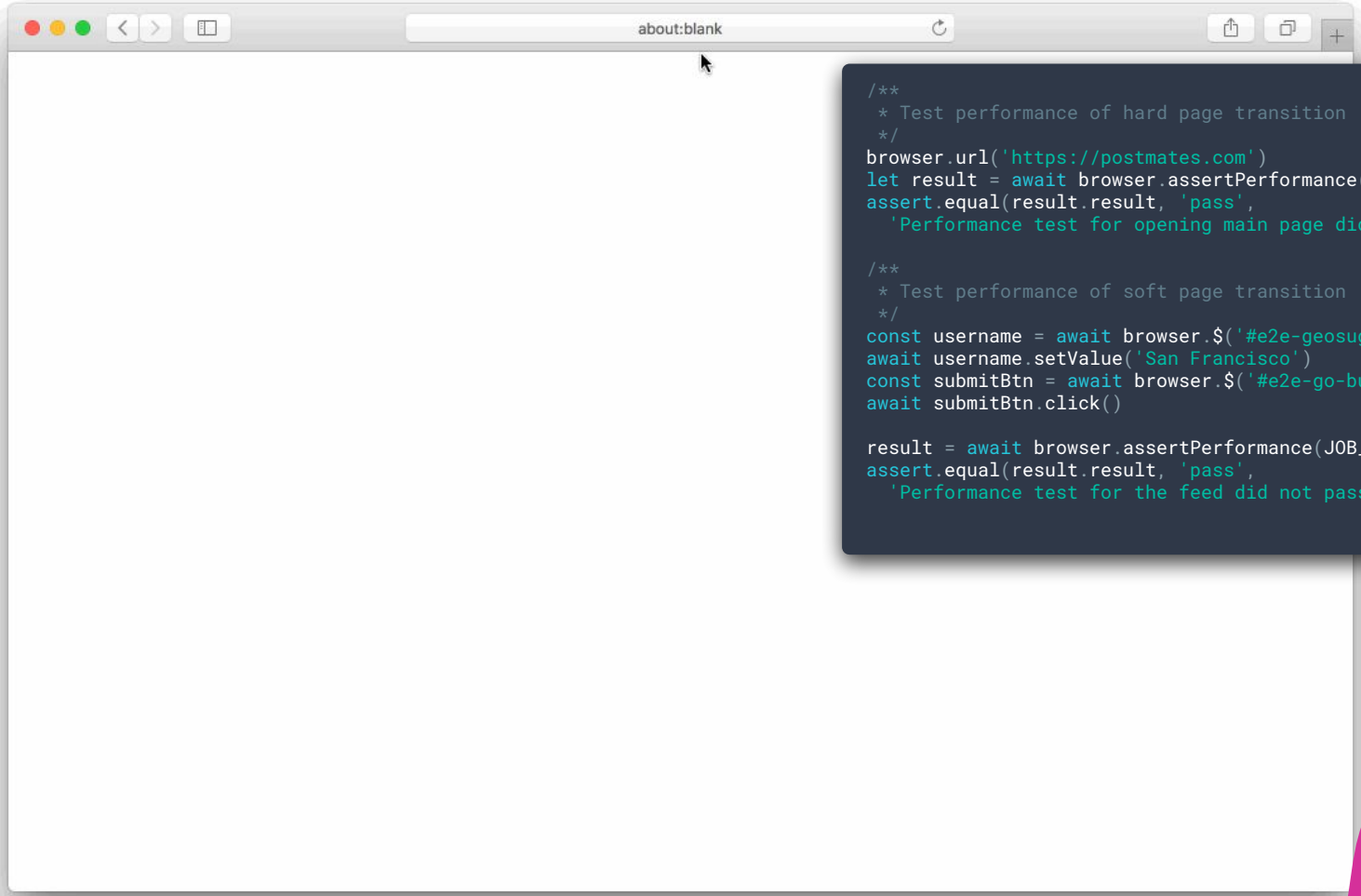
```
const submitBtn = await browser.$('button[type="submit"]')
await submitBtn.click()

const result = await browser.assertPerformance(
  'My Performance Test',
  ['speedIndex', 'timeToFirstInteractive'])

expect(result.pass).toBe(true)
```

W3C WebDriver Extension

/session/:sessionId/sauce/ondemand/performance

JS Executor (Selenium Python)

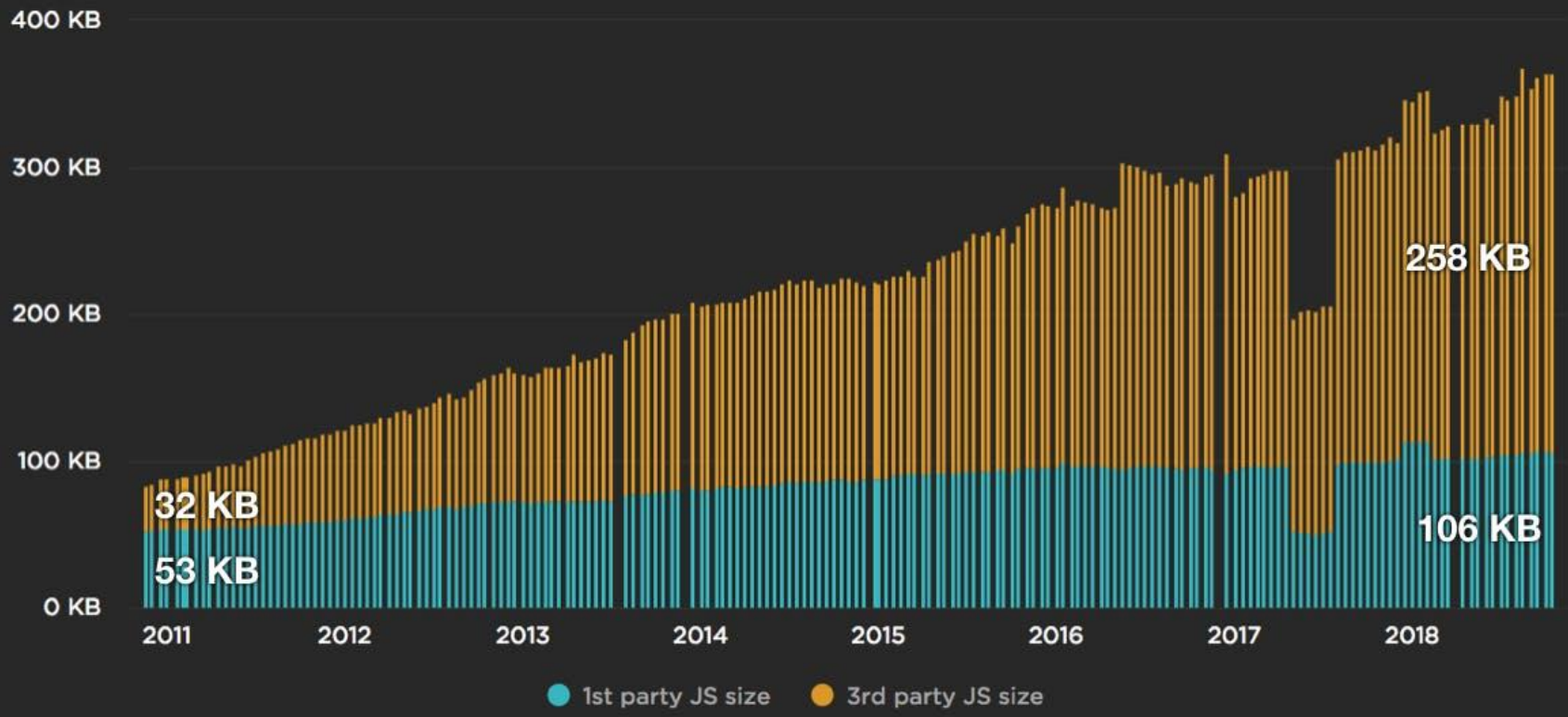
driver.execute_script('sauce:performance', {"metrics": [...]"})



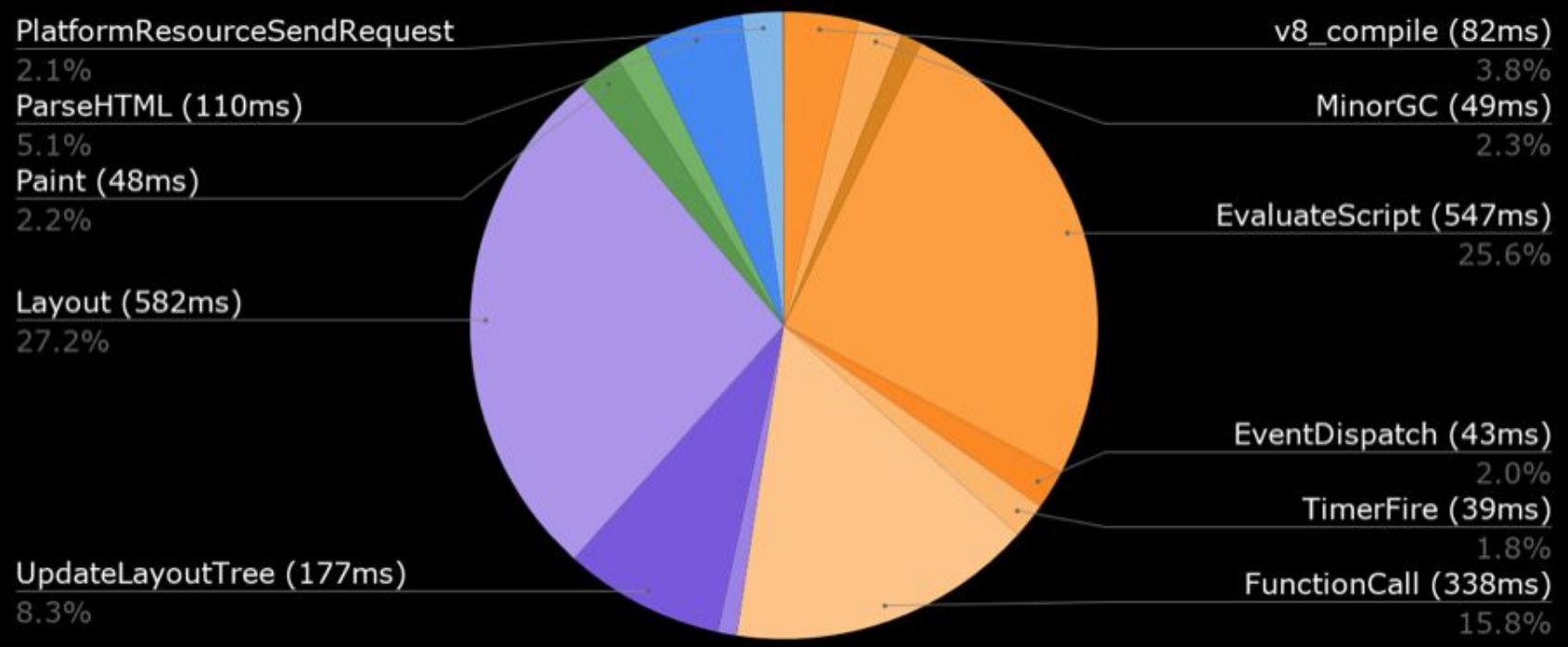

```
/**
 * Test performance of hard page transition
 */
browser.url('https://postmates.com')
let result = await browser.assertPerformance(JOB_NAME, ['score'])
assert.equal(result.result, 'pass',
  'Performance test for opening main page did not pass')

/**
 * Test performance of soft page transition
 */
const username = await browser.$('#e2e-geosuggest-input')
await username.setValue('San Francisco')
const submitBtn = await browser.$('#e2e-go-button')
await submitBtn.click()

result = await browser.assertPerformance(JOB_NAME, ['score'])
assert.equal(result.result, 'pass',
  'Performance test for the feed did not pass')
```

Jankiness

The Browser
A JavaScript Powerplant



https://speedcurve.com/blog/javascript-growth

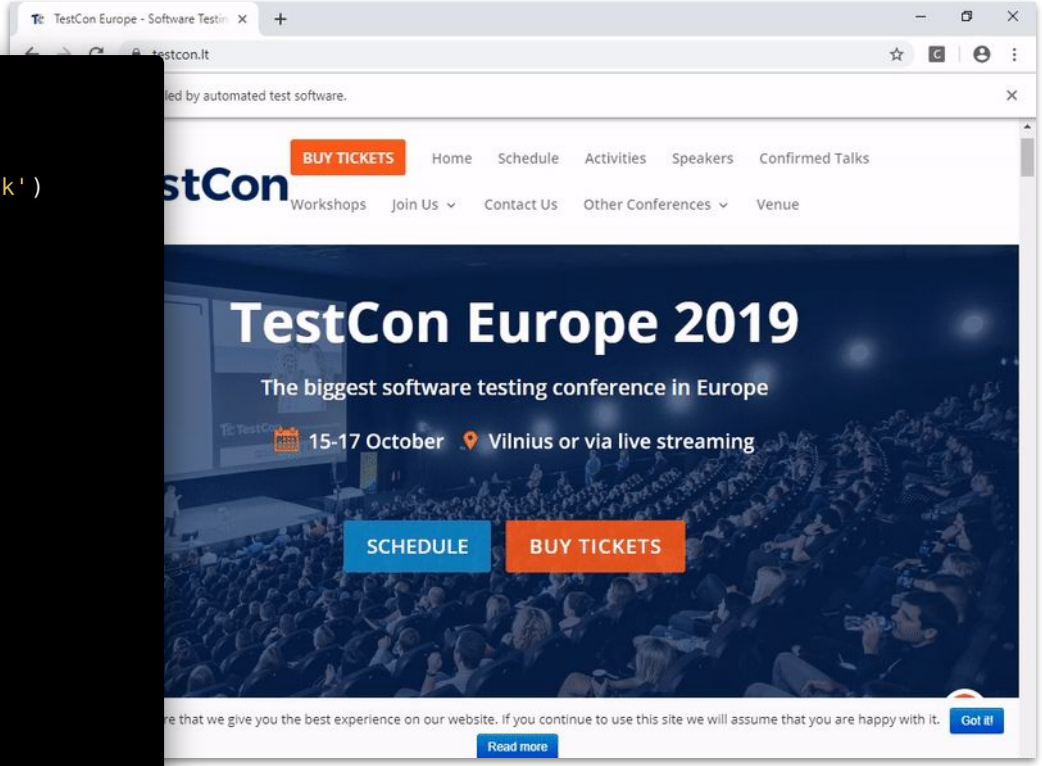


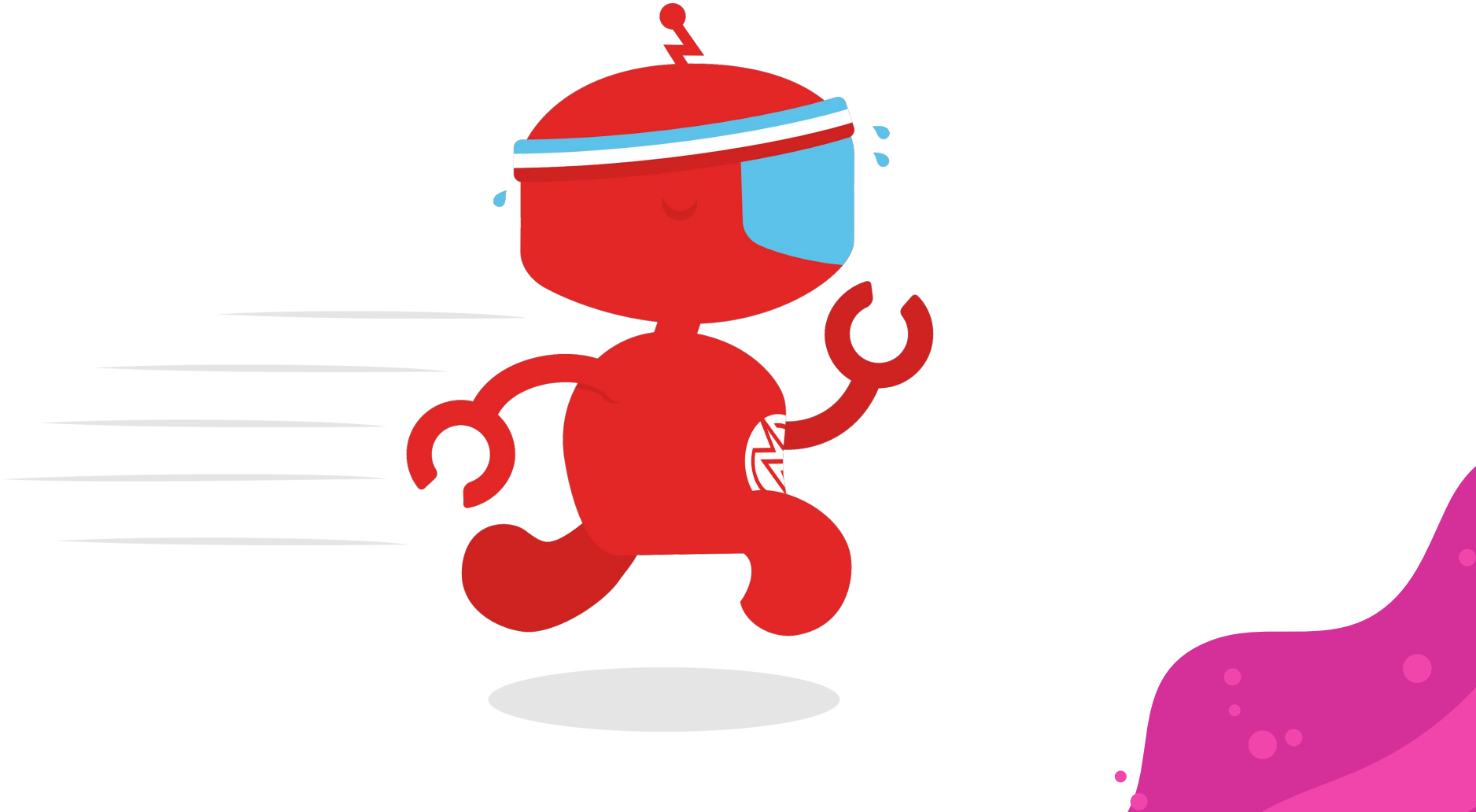
https://discuss.httparchive.org/t/cpu-time-breakdown/1510

“Jank is any stuttering, juddering or just plain halting that users see when a site or app isn't keeping up with the refresh rate. Jank is the result of frames taking too long for a browser to make, and it negatively impacts your users and how they experience your site or app.”

—jankfree.org

```
browser.url('https://www.testcon.lt/')
const result = browser.execute('sauce:jankinessCheck')
console.log(result)
/**
 * { url: 'https://www.testcon.lt/',
   timestamp: 1571130036741,
   value:
    { metrics:
       { averageFPS: 27.565962053070823,
         scriptingTime: 830,
         renderingTime: 127,
         otherTime: 188,
         idleTime: 3948,
         forcedReflowWarningCounts: 10,
         scrollTime: 5121,
         paintingTime: 94,
         memoryUsageDiff: 33128272 },
      diagnostics:
       { layoutUpdateScore: 0.9895397489539749,
         fpsScore: 0.4594327008845137,
         idleDurationScore: 0.9090146326859025,
         memoryUsageScore: 0.9847441842390949 } },
   score: 0.7391026104441131,
   loaderId: '40dc6000-ef2a-11e9-b825-57924a3e217f',
   type: 'scroll' }
 */
```





Performance Best Practices

What to do and what not to do?!

- Functional vs. Performance Testing
- Don’t worry about other browser / versions too much
- Keep it simple!
- Maintain one job name for one performance test
- Know what you want to test
  - Scoring based metrics are the best generalised metrics
  - Use others if you have more specific requirements

Thanks!

Does anyone have any questions?

https://speakerdeck.com/christianbromann/automated-performance-testing-with-webdriver

christian@saucelabs.com

christian-bromann

@bromann