Automated Web-Performance Testing with WebDriver
Hello!
I am Christian Bromann
Software Engineer for the DevTools team at SauceLabs

@bromann
christian-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/)

3.21s 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.
1. 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>URI Template</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/session</td>
<td>New Session</td>
</tr>
<tr>
<td>DELETE</td>
<td>/session/{session id}</td>
<td>Delete Session</td>
</tr>
<tr>
<td>GET</td>
<td>/status</td>
<td>Status</td>
</tr>
<tr>
<td>GET</td>
<td>/session/{session id}/timeouts</td>
<td>Get Timeouts</td>
</tr>
<tr>
<td>POST</td>
<td>/session/{session id}/timeouts</td>
<td>Set Timeouts</td>
</tr>
<tr>
<td>POST</td>
<td>/session/{session id}/url</td>
<td>Navigate To</td>
</tr>
<tr>
<td>GET</td>
<td>/session/{session id}/url</td>
<td>Get Current URL</td>
</tr>
<tr>
<td>POST</td>
<td>/session/{session id}/back</td>
<td>Back</td>
</tr>
<tr>
<td>POST</td>
<td>/session/{session id}/forward</td>
<td>Forward</td>
</tr>
<tr>
<td>POST</td>
<td>/session/{session id}/refresh</td>
<td>Refresh</td>
</tr>
<tr>
<td>GET</td>
<td>/session/{session id}/title</td>
<td>Get Title</td>
</tr>
</tbody>
</table>
```javascript
const elem = $('#myElem')
 elem.click()
```
WebDriver Limitations

- Only a limited set of commands
- Stateless
- No debugging capabilities
- Slow
- Stagnant innovation
Simple Browser Automation

Limitations:

- Only one browser (Chrome)
- Manual Test Setup (Testrunner, Assertion Library, etc)
Test Status Menu
See how many tests passed or failed and how fast the tests took to run.

Url Preview
The url of your app updates as you test, so you have more visibility on testing routes.

Viewport Sizing
Set your app's viewport size to test responsive layouts.

Command Log
Every command logs as it executes. Hover to highlight the affected element, or click to log more info to the console.

App preview
See your app while test commands execute in real-time. Use your devtools to inspect or debug each command.
Cypress.io Limitations

- Only one browser (Chrome)
- Only in JavaScript
- Vendor bound
- No mobile support
What to use?

Pick a test automation tool that suits your team requirements!
2. Browser Performance

And how to make the web fast again
Browser Performance

responseStart - navigationStart = time to first byte (TTFB)
Browser Performance

```javascript
// Start with one mark.
performance.mark("mySetTimeout-start");

// Wait some time.
setTimeout(function() {
    // Mark the end of the time.
    performance.mark("mySetTimeout-end");

    // Measure between the two different markers.
    performance.measure(
        "mySetTimeout",
        "mySetTimeout-start",
        "mySetTimeout-end"
    );

    // Get all of the measures out.
    // In this case there is only one.
    var measures = performance.getEntriesByName("mySetTimeout");
    var measure = measures[0];
    console.log("setTimeout milliseconds:", measure.duration)

    // Clean up the stored markers.
    performance.clearMarks();
    performance.clearMeasures();
}, 1000);
```

setTimeout milliseconds: 1376.199999987893
Browser Performance

```javascript
> window.performance.getEntriesByType("paint")
```

```
(2) [PerformancePaintTiming, PerformancePaintTiming]

  0: PerformancePaintTiming
      duration: 0
      entryType: "paint"
      name: "first-paint"
      startTime: 3997.000000003027
      __proto__: PerformancePaintTiming

  1: PerformancePaintTiming
      duration: 0
      entryType: "paint"
      name: "first-contentful-paint"
      startTime: 3997.000000003027
      __proto__: PerformancePaintTiming

__proto__: Array(0)
```

Chrome/Opera ONLY!
## Paint Metrics

<table>
<thead>
<tr>
<th>First Pain (FP)</th>
<th>First Contentful Paint (FCP)</th>
<th>First Meaningful Paint (FMP)</th>
<th>Time To Interactive (TTI)</th>
</tr>
</thead>
</table>

**First Pain**
- first render to the screen

**First Contentful Paint**
- is triggered when any content is painted – i.e. something defined in the DOM

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

**Time To Interactive**
- number of seconds from the time the navigation started until the layout is stabilized
computes an overall score for how quickly the content painted
The Question Is ...

Are all these metrics important?

YES!
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
The Experiment

Script Flaw

```javascript
function foo (i) {
    return i * Math.random()
}

for (var i = 0; i < 10e3; ++i) {
    foo(i)
    document.onclick = () => console.log('foo')
}
```

Render Flaw

```javascript
for (var i = 0; i < 10e3; ++i) {
    const elem = document.createElement('div')
    elem.style.backgroundColor = 'red'
    elem.style.width = '10px'
    elem.style.hei = '10px'
    elem.style.height = '10px'
    document.querySelector('body').appendChild(elem)
    setTimeout(() => elem.remove(), 0)
}
```
Performance Flaws Influence SpeedIndex
Impact On `DOMContentLoaded`
Performance Flaws Influence SpeedIndex
Fast forward to today and we see that window.onload doesn’t reflect the user perception as well as it once did.

Steve Souders.
3.

Browser Tracing

How the browser knows about all this
Tracing Browser

Start profiling and reload page to begin.

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

Click the reload button or hit ⌘ R ⌘ 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
Tracing Event Format

- 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:

```json
{
    "name": "myName",
    "cat": "category,list",
    "ph": "B",
    "ts": 12345,
    "pid": 123,
    "tid": 456,
    "args": {
        "someArg": 1,
        "anotherArg": {
            "value": "my value"
        }
    }
}
```
Tracing Format Viewers - DevTools Performance Tab
Tracing Format Viewers
Tracing Event Format

```
{
  "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"
}
```
Tracing Event Format

```json
{
    "pid": 579,
    "tid": 775,
    "ts": 170383426118,
    "ph": "O",
    "cat": "disabled-by-default-devtools.screenshot",
    "name": "Screenshot",
    "args": {
        "snapshot": "...
    },
    "tts": 2879188825,
    "id": "0x1"
}
```
4.

Performance Testing

How to keep track of our application performance
Shift Testing To The Left
Current Process

- Test in Lab
- Write Code
- Validate
- Release to user
Existing Solutions

Performance in the **Lab**

Development

------------------------

Performance in the **Real World**

Staging

New Relic

Production

WEBPAGETEST
WebdriverIO DevTools Service

A WebdriverIO service that allows you to run Chrome DevTools commands in your tests

With Chrome v63 and up the browser started to support multi clients allowing arbitrary clients to access the Chrome DevTools Protocol. This provides interesting opportunities to automate Chrome beyond the WebDriver protocol. With this service you can enhance the wdio browser object to leverage that access and call Chrome DevTools commands within your tests to e.g. intercept requests, throttle network capabilities or take CSS/JS coverage.

Note: this service currently only supports Chrome v63 and up!

Installation

The easiest way is to keep wdio-devtools-service as a devDependency in your package.json.
WebDriver Architecture

Open Chrome
/Applications/.../Google Chrome
--remote-debugging-port=0

Chromedriver

POST /wd/hub/session

HTTP

Creates File
/var/folders/ns/.../DevToolsActivePort
WebDriver Architecture

Chromedriver

HTTP

POST

/wd/hub/<sessionId>/url

Sends WebSocket Command

Page.navigate

Opens

chrome://version

Get User Data Dir

Including DevToolsActivePort

its Content
WebDriver Architecture

Chromedriver

browser.url("https://webdriver.io")

browser.startTracing()
White is the color of milk and fresh snow, the color produced by the combination of all the colors of the visible spectrum.

```javascript
// wdio.conf.js
const path = require('path')
exports.config = {
  hostname: '0.0.0.0',
  port: 4444,

  specs: [
    './tests/performance/speedIndex.test.js'
  ],

  capabilities: [{
    browserName: 'chrome'
  }],

  services: ['devtools'],

  logLevel: 'trace',
  coloredLogs: true,
  framework: 'mocha',
  logDir: __dirname,
  reporters: ['spec'],
  mochaOpts: {
    ui: 'bdd',
    timeout: 60000
  }
}
```

```javascript
// speedIndex.test.js
describe('App Performance Tests', () => {
  before(() => {
    browser.startTracing()
    browser.url('http://json.org')
    browser.endTracing()
  })

  it('Capture Speedindex', () => {
    console.log(browser.getSpeedIndex())
    // outputs:
    // { 
    //   speedIndex: 689.663480006,
    //   perceptualSpeedIndex: 785.090186023
    // }
  })
})
```
White is the color of milk and fresh snow, the color produced by the combination of all the colors of the visible spectrum.
const path = require('path')

exports.config = {
  hostname: '0.0.0.0',
  port: 4444,

  specs: ['./tests/performance/pageWeight.test.js'],

  capabilities: [{
    browserName: 'chrome'
  }],

  services: ['devtools'],

  logLevel: 'trace',
  coloredLogs: true,
  framework: 'mocha',
  logDir: __dirname,
  reporters: ['spec'],

  mochaOpts: {
    ui: 'bdd',
    timeout: 60000
  }
}

// pageWeight.test.js

describe('App Performance Tests', () => {
  before(() => {
    // trace page
    browser.startTracing()
    browser.url('http://saucelabs.com')
    browser.endTracing()
  })

  it('Get Page Weights', () => {
    console.log(browser.getPageWeight())
    // outputs:
    // {
    //   pageWeight: 2438485,
    //   transferred: 1139136,
    //   requestCount: 72,
    //   details: {
    //     Document: {
    //       size: 221705,
    //       encoded: 85386,
    //       count: 11
    //     },
    //     Stylesheet: { ... },
    //     Image: { ... },
    //     Script: { ... },
    //     Font: { ... },
    //     Other: { ... },
    //     XHR: { ... }
    //   }
    // }
  })
})
const path = require('path')

exports.config = {
    /**
     * server configurations
     */
    hostname: '0.0.0.0',
    port: 4444,

    /**
     * specify test files
     */
    specs: ['./tests/performance/tracelog.test.js'],

    /**
     * capabilities
     */
    capabilities: [{}
        browserName: 'chrome'
    ],

    services: ['devtools'],

    /**
     * test configurations
     */
    logLevel: 'trace',
    coloredLogs: true,
    framework: 'mocha',
    logDir: __dirname,

    reporters: ['spec'],

    mochaOpts: {
        ui: 'bdd',
        timeout: 60000
    }
}
Goal of tutorial

This tutorial teaches you how to use Chrome DevTools to find ways to make your websites load faster.

Read on, or watch the video version of this tutorial:

https://developers.google.com/web/tools/chrome-devtools/speed/get-started
https://github.com/christian-bromann/webdriverio-performance-testing
Thanks!

Any questions?

You can find me at:

@bromann

christian@saucelabs.com