WebDriver BiDi

Three Ways to Automate Your Browser, and Why We Are Adding a Fourth

Michael Hablich @mhablich
Product Manager on the Google Chrome team

Michael Hablich

Product Manager @ Google Chrome

~ 20 years in tech

Spent a significant chunk of this building test automation solutions

Why am I here?

- Developer satisfaction of testing in web development is looooow
- Testing your application is a significant chunk of QA
- Automating tests is a good way to reduce this cost

Browser automation
in a nutshell

example.js | Order Mocca.js | coffee.js

src > Order Mocca.js > <function>

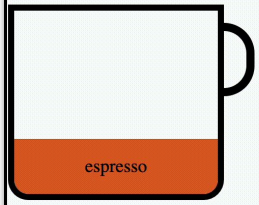
```
{
    const targetPage = page;
    const element = await waitForSelectors([["aria/Mocha"],["[data-test=Mocha]"]], targetPage, { timeout, visible: true });
    await scrollIntoViewIfNeeded(element, timeout);
    await element.click({
      offset: {
        x: 96.40188598632812,
        y: 53.254608154296875,
      },
    });
}
{
    const targetPage = page;
    const element = await waitForSelectors([["aria/Cart page"],["#app > ul > li:nth-child(2) > a"]], targetPage, { timeout, vis
    await scrollIntoViewIfNeeded(element, timeout);
    await element.click({
      offset: {
        x: 19.859375,
        y: 10.5,
      },
    });
}
{
    const targetPage = page;
    const element = await waitForSelectors([["aria/Add one Mocha"],["#app > div.list > div > ul > li.list-item > div:nth-child(2
    await scrollIntoViewIfNeeded(element, timeout);
    await element.click({
      offset: {
        x: 14,
        y: 9.78125,
      },
    });
}
{
    const targetPage = page;
    const element = await waitForSelectors([["aria/Add one Mocha"],["#app > div.list > div > ul > li.list-item > div:nth-child(2) >
    await scrollIntoViewIfNeeded(element, timeout);
    await element.click({
      offset: {
```

Code that simulates a user

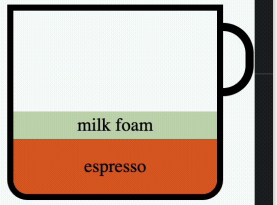
Chrome is being controlled by automated test software.

menu | cart (0) | github | 526px × 1001px

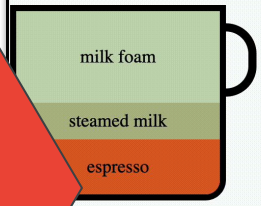
Espresso $10.00



Espresso Macchiato $12.00

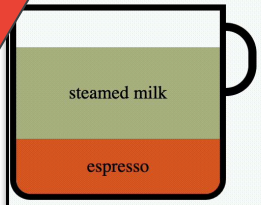


Cappucino $19.00

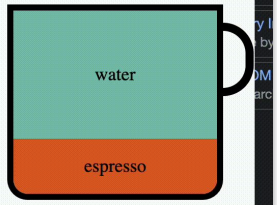


Mocha $8.00

White .00



Americano $7.00



Cafe Latte $16.00

Espresso Con Panna $14.0

Total: $0.00

90s

1993

1994

1995

1996

Birth of the web.
Static web pages.
Interactivity with plugins.

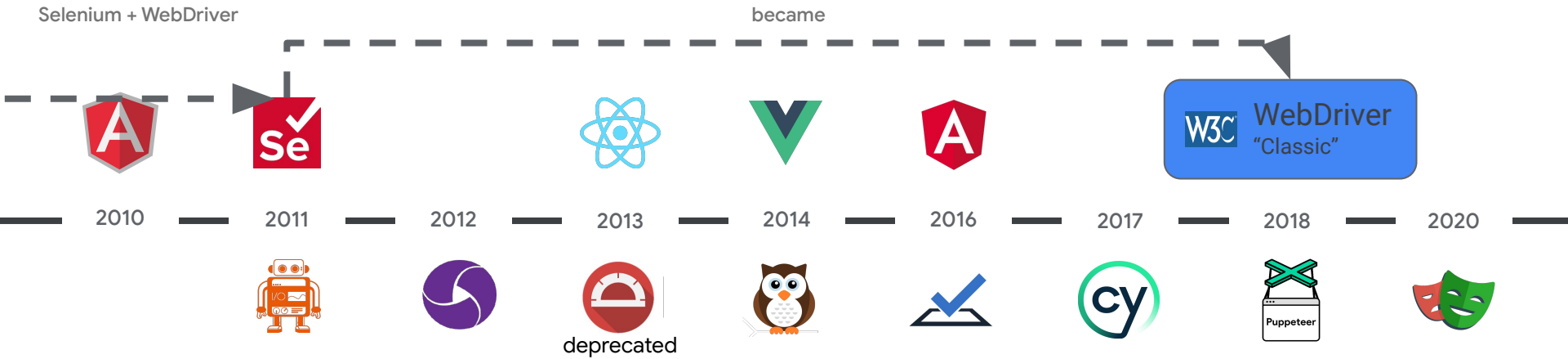
2000

2002 — 2003 — 2004 — 2005 — 2006 — 2008 — 2009

WebDriver

More browsers.
Interactive & responsive.
Browser automation.

2010



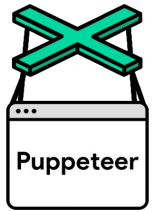
WebDriver became a W3C standard.
Popular JavaScript frameworks.
JavaScript automation tools.

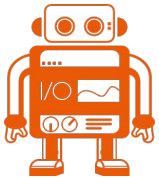
There are three ways to automate a browser

WebDriver Protocol

ChromeDevTools Protocol (CDP)

Web APIs + Browser Extension





How tools automate browsers

High level

Execute JavaScript injected into the browser

Low level

Execute remote commands outside of the browser

Web APIs & Browser Extension

bit.ly/testcafe-architecture
bit.ly/cypress-architecture

Low level

Execute remote commands outside of the browser

Web APIs & Browser Extension

cypress

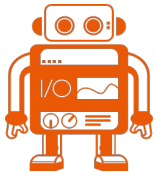
TestCafé®

bit.ly/testcafe-architecture
bit.ly/cypress-architecture

WebDriver “Classic” Protocol

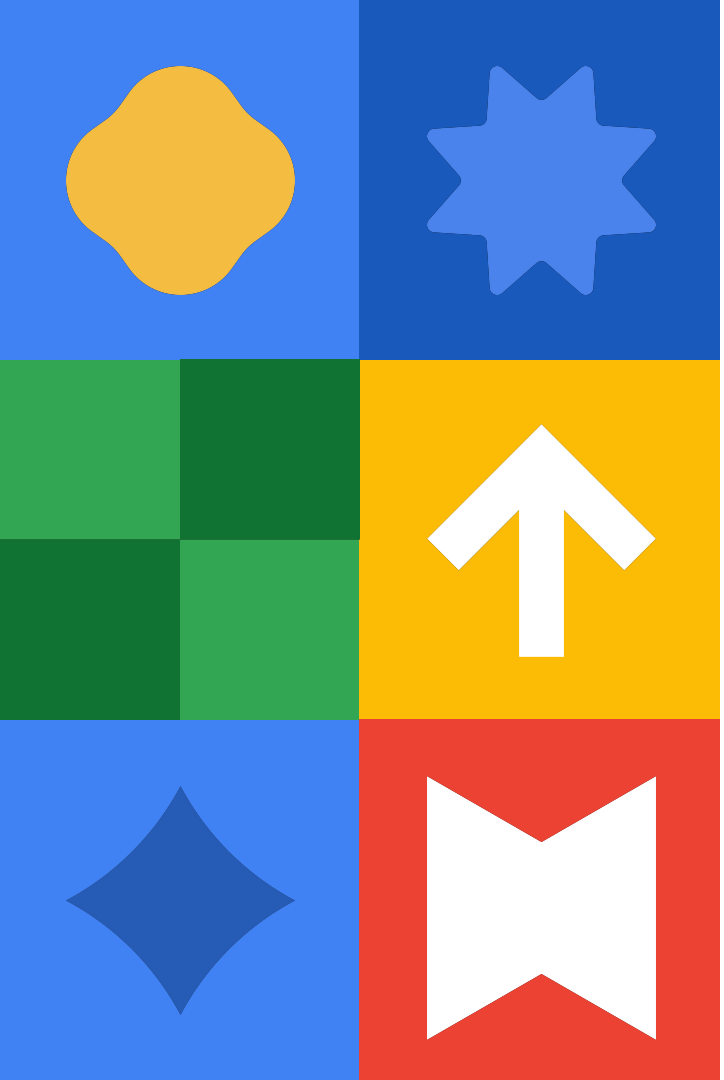
Selenium

WebdriverIO



Nighwatch

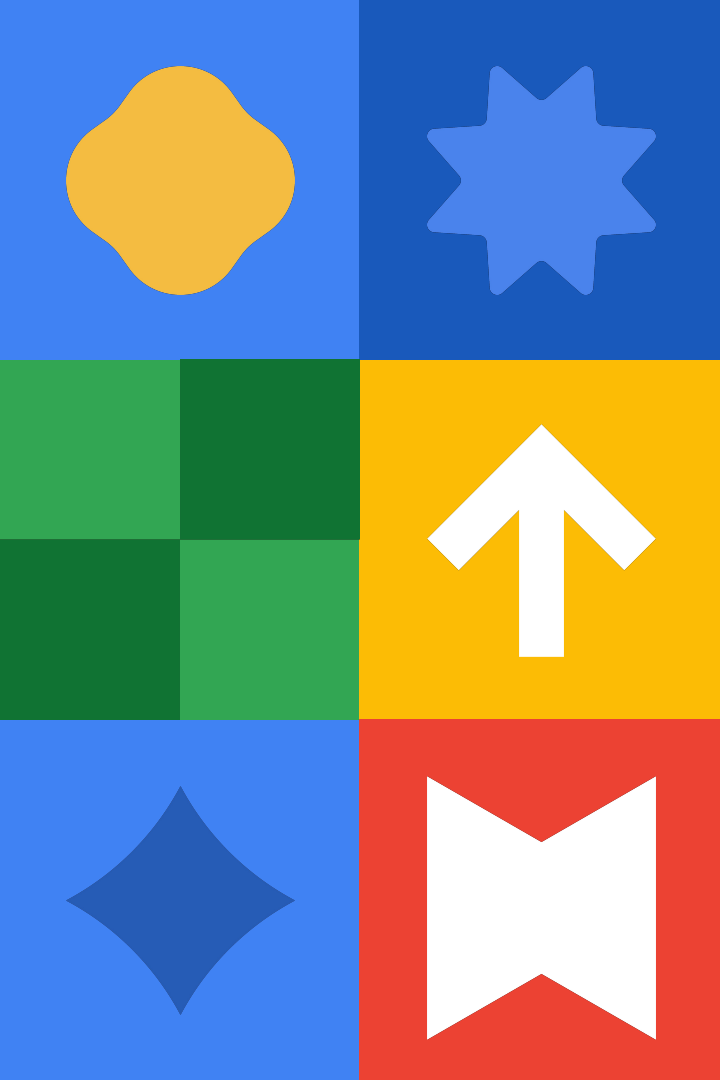
Chrome DevTools Protocol (CDP)



Web APIs & Chrome Extensions

The built-your-own automation approach

First on my list of browser automation techniques



WebDriver

The Standard Technique

Second on my list of browser automation techniques

Web APIs & Browser Extension

cypress

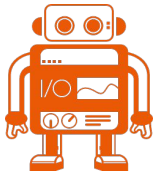
TestCafé®

bit.ly/testcafe-architecture
bit.ly/cypress-architecture

WebDriver “Classic” Protocol

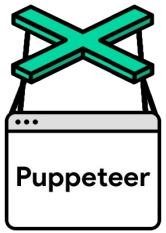
Selenium

WebdriverIO



Nighwatch

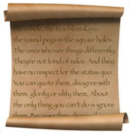
Chrome DevTools Protocol (CDP)



What you do

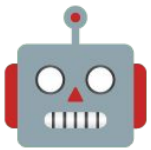
Developers

Pick an **automation tool**
- library or framework



Test scripts

Behind the scenes

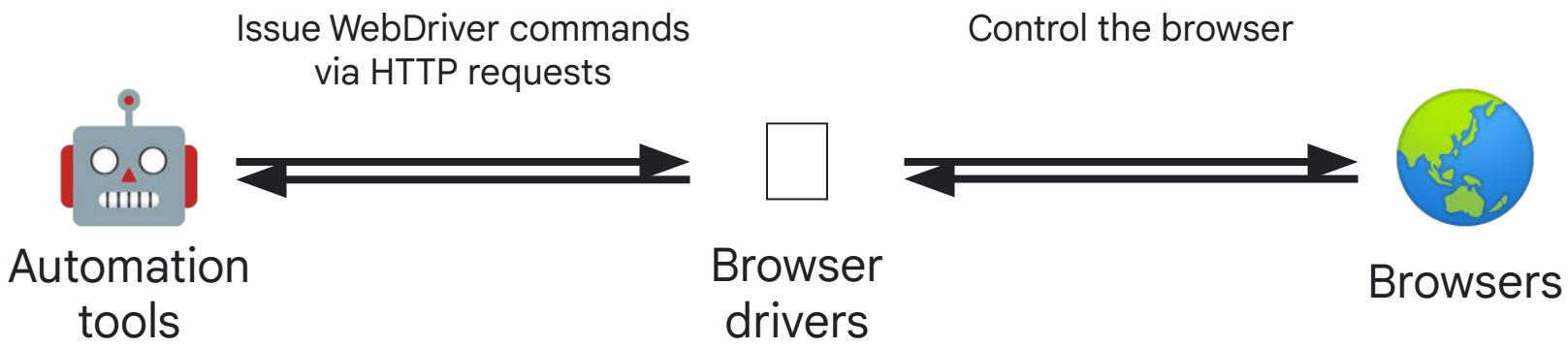


Automation tools

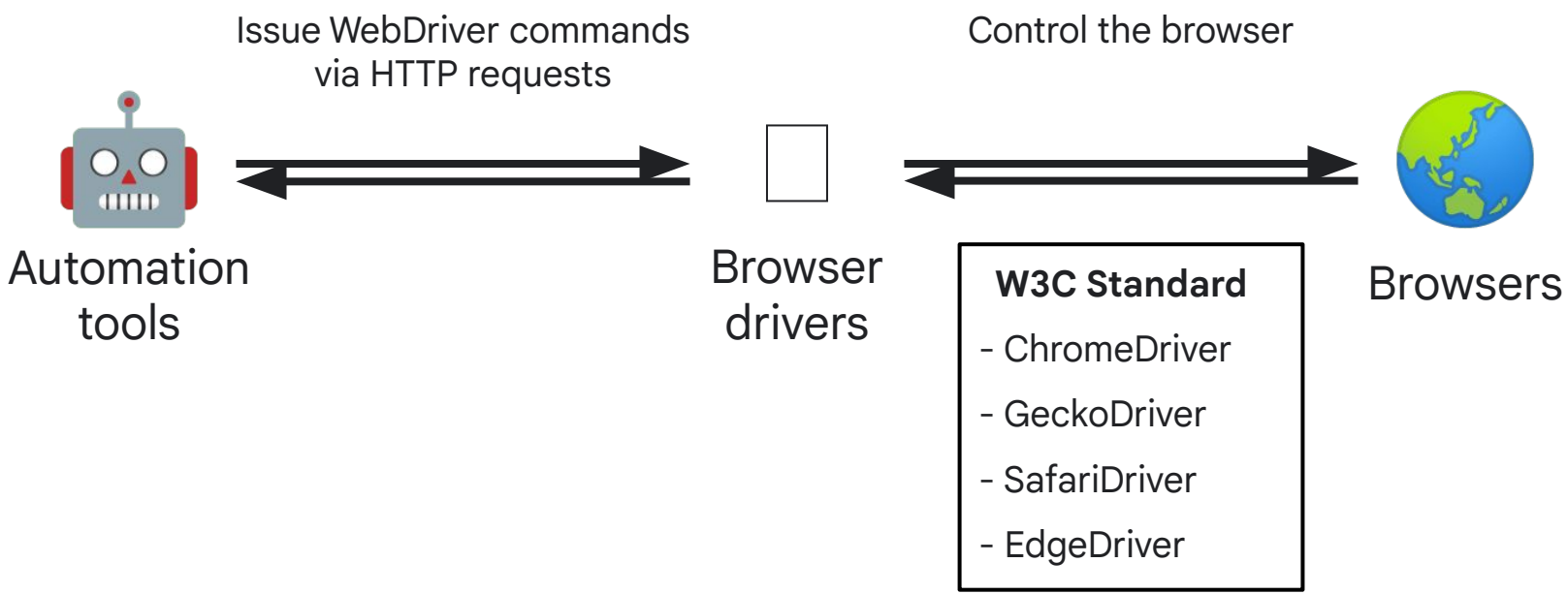
Run your scripts in browser through **protocols**

Browsers

WebDriver
“Classic”

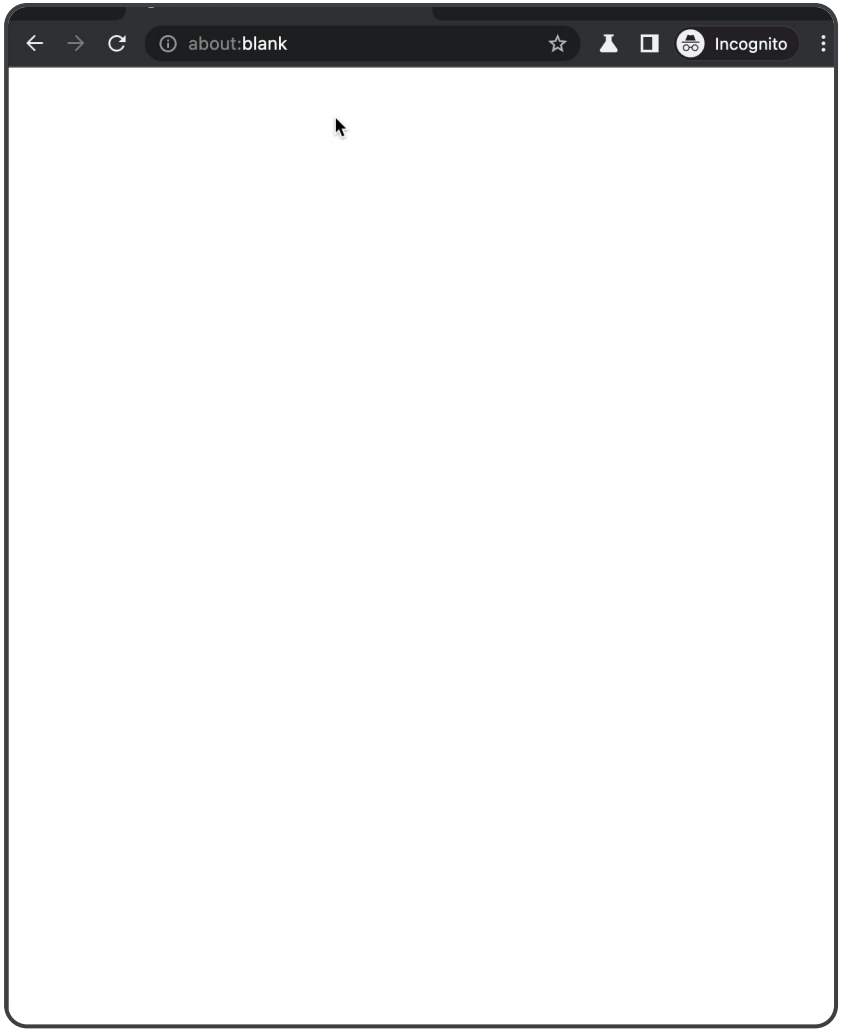


WebDriver
“Classic”



WebDriver
“Classic”

1. Navigate to the page
2. Click on Espresso



Demos by @jecfish

WebdriverIO

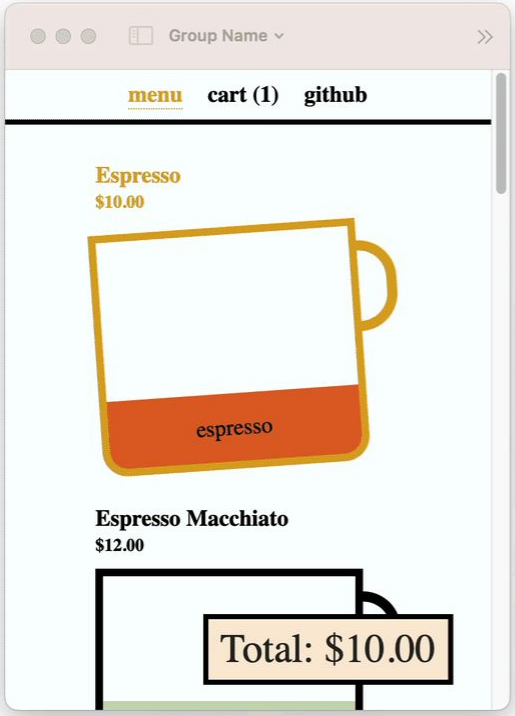
```
/* WebdriverIO */

// Setup browser
const browser = ...;

// Action
await browser.setWindowSize(600, 1041);
await browser.url('https://coffee-cart.app');
await browser.$('[data-test="Espresso"]').click();
```

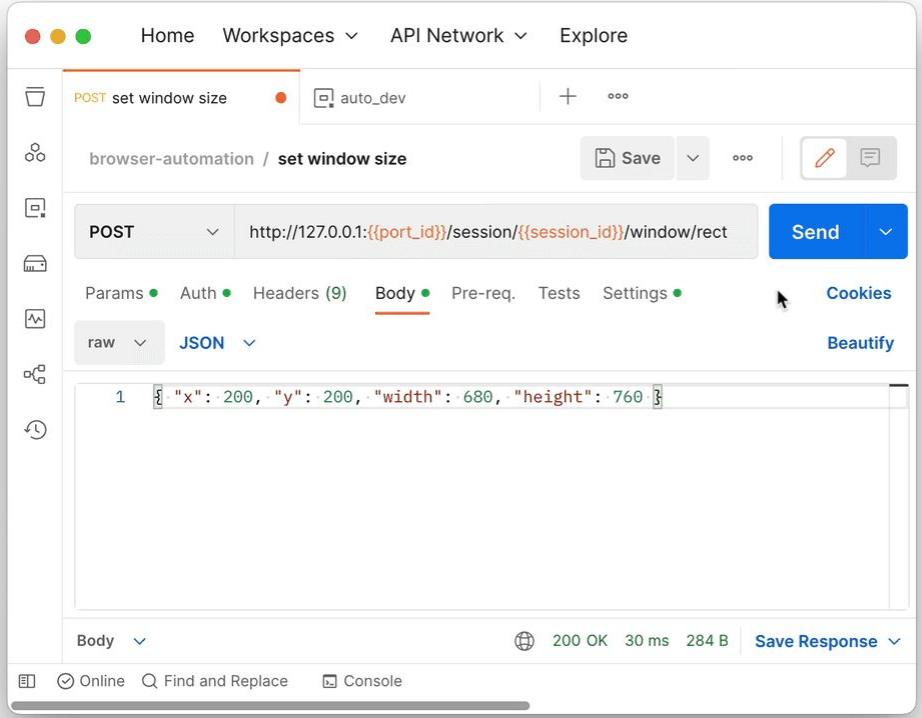
Full demo: bit.ly/wbidi-demo

WebdriverIO

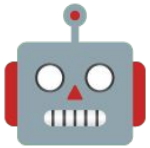
```
/* WebdriverIO */

// Setup browser
const browser = ...;

// Action
await browser.setWindowSize(600, 1041);
await browser.url('https://coffee-cart.app');
await browser.$('[data-test="Espresso"]').click();
```

Full demo: bit.ly/wbidi-demo


Group Name
menu
cart (1)
github
Espresso
$10.00
espresso
Espresso Macchiato
$12.00
Total: $10.00


Home
Workspaces
API Network
Explore
POST set window size
auto_dev
browser-automation / set window size
Save
POST
http://127.0.0.1:{{port_id}}/session/{{session_id}}/window/rect
Send
Params
Auth
Headers (9)
Body
Pre-req.
Tests
Settings
Cookies
raw
JSON
Beautify
1 { "x": 200, "y": 200, "width": 680, "height": 760 }
Body
200 OK
30 ms
284 B
Save Response
Online
Find and Replace
Console

WebDriver
“Classic”



Automation tools

1. Navigate to the page

[POST] /session/:id/url

```
{ url: 'https://coffee-cart.app' }
```

Browser drivers

2. Find the coffee

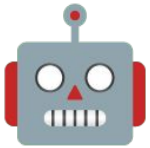
[POST] /session/:id/element

```
{ using: 'css selector', value: '[data-test="Espresso"]' }
```

3. Click the coffee

[POST] /session/:id/element/:element-id/click

WebDriver
“Classic”



Automation tools

1. Navigate to the page

[POST] /session/:id/url

```
{ url: 'https://coffee-cart.app' }
```

Browser drivers

2. Find the coffee

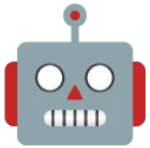
[POST] /session/:id/element

```
{ using: 'css selector', value: '[data-test="Espresso"]' }
```

3. Click the coffee

[POST] /session/:id/element/:element-id/click

WebDriver
“Classic”



Automation tools

1. Navigate to the page

[POST] /session/:id/url

```
{ url: 'https://coffee-cart.app' }
```

2. Find the coffee

[POST] /session/:id/element

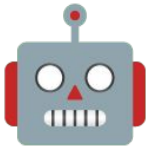
```
{ using: 'css selector', value: '[data-test="Espresso"]' }
```

3. Click the coffee

[POST] /session/:id/element/:element-id/click

Browser drivers

WebDriver
“Classic”



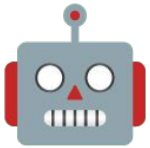
Automation tools

Wait for the Espresso through
HTTP long polling

Establish a session

Browser drivers

WebDriver
“Classic”



Automation tools

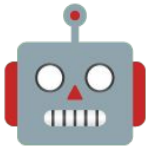
Wait for the Espresso through HTTP long polling

Establish a session

Is the Espresso loaded?

Browser drivers

WebDriver
“Classic”



Automation tools

Wait for the Espresso through HTTP long polling

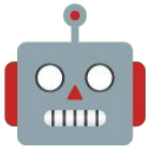
Establish a session

Is the Espresso loaded?

Is the Espresso loaded?

Browser drivers

WebDriver
“Classic”



Automation tools

Wait for the Espresso through HTTP long polling

Establish a session →

Is the Espresso loaded? →

← Is the Espresso loaded? →

Slow!

Browser drivers

WebDriver
“Classic”

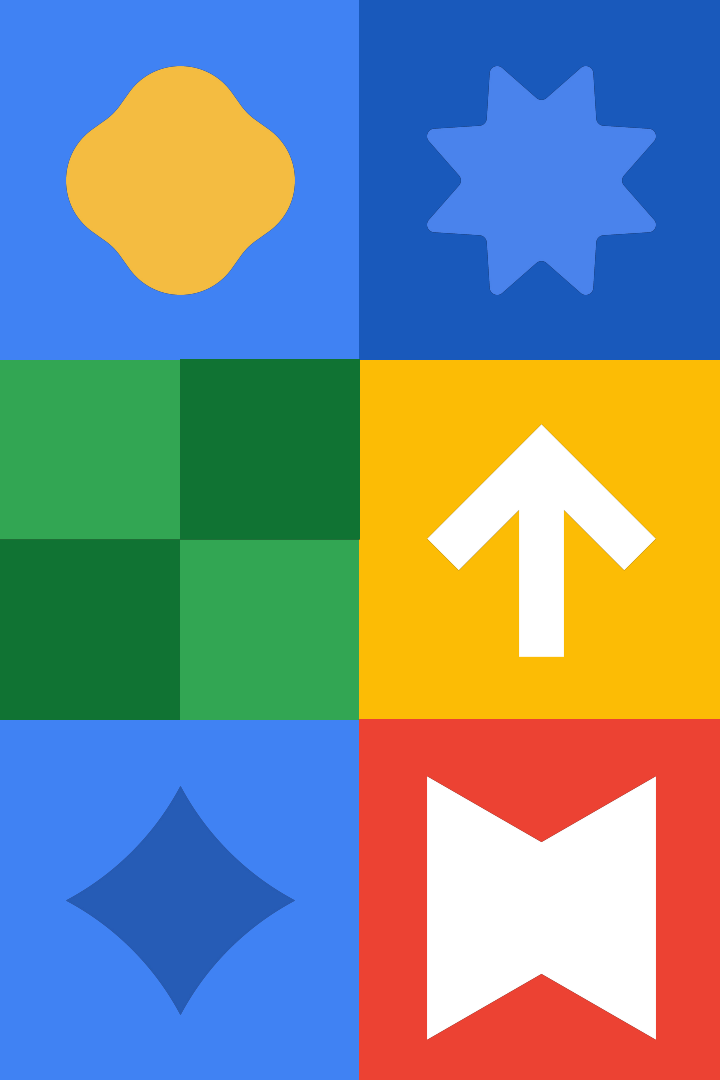
Best **cross-browser** support

W3C standard

Built for **testing**

Slower, HTTP-based

Provide **only high-level** control



CDP

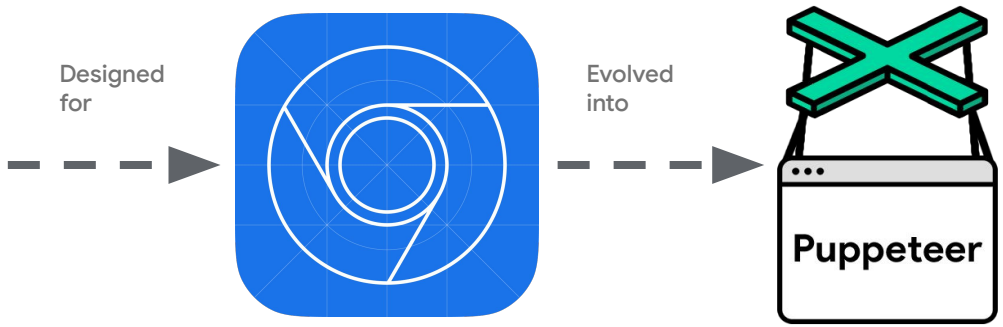
The Chrome Devtools Protocol

Third on my list of browser automation techniques

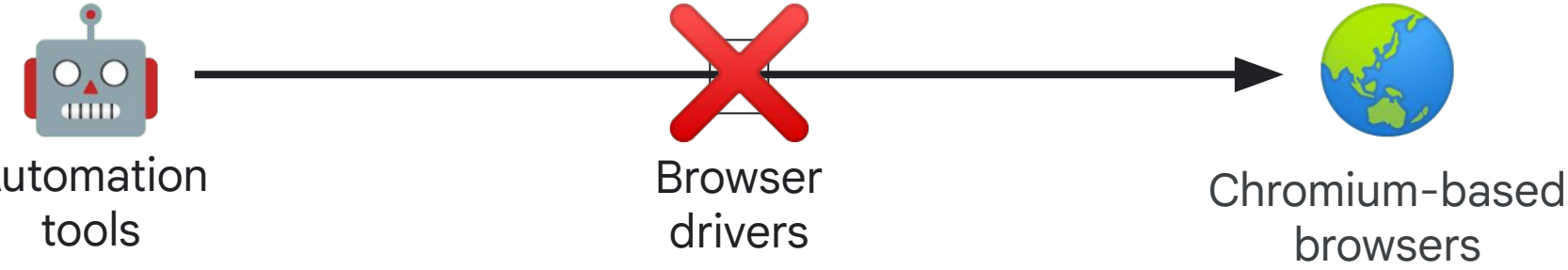
CDP

Chrome DevTools Protocol

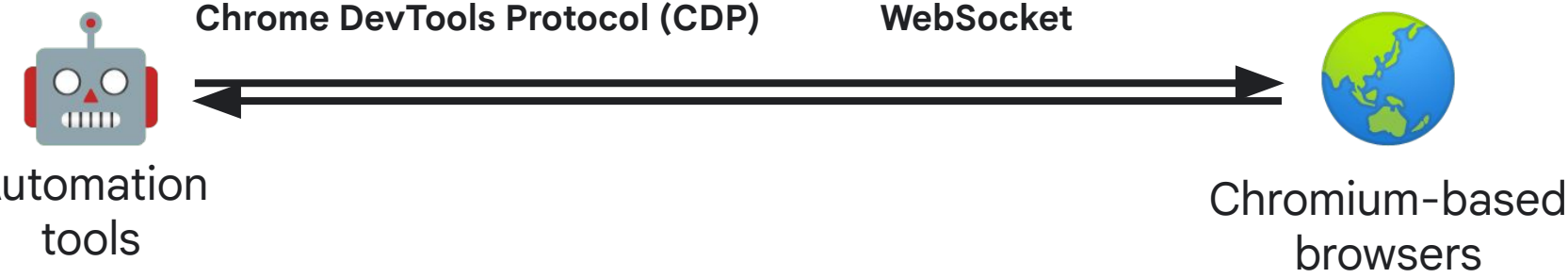
goo.gle/cdp-docs



CDP

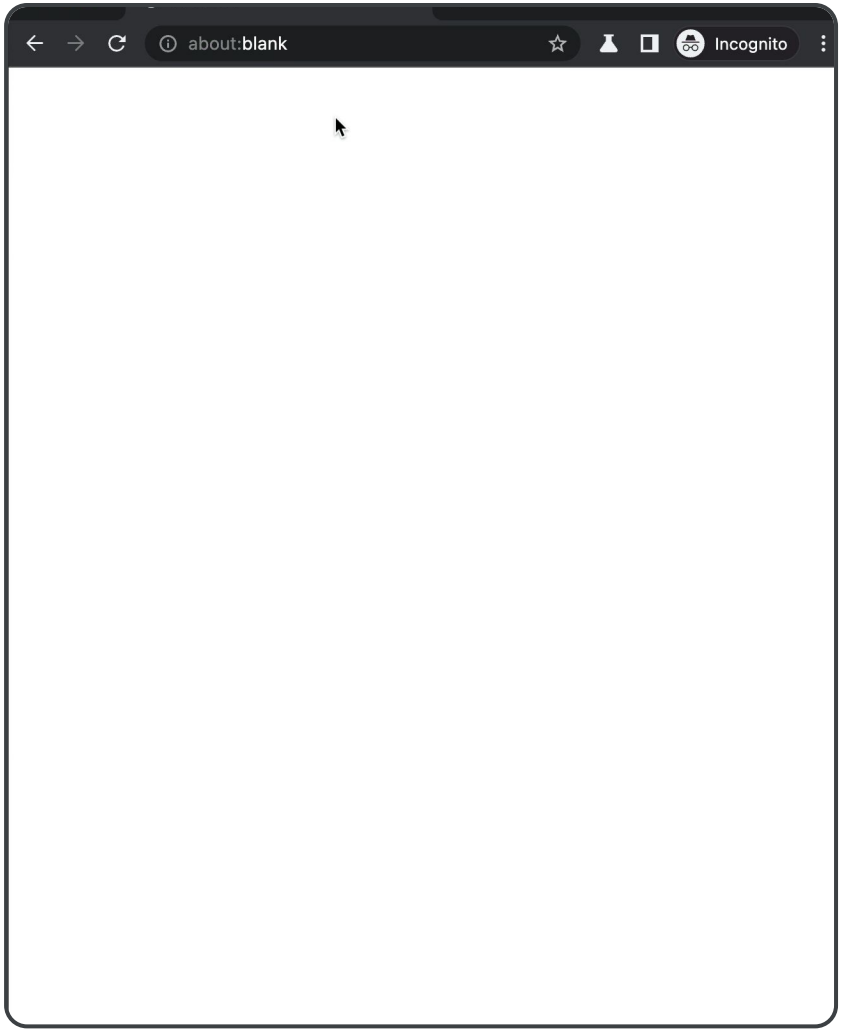

Automation tools
Browser drivers
Chromium-based browsers

CDP


Issue commands through
Chrome DevTools Protocol (CDP)
Control the browser via
WebSocket
Automation tools
Chromium-based browsers

CDP

1. Navigate to the page
2. Click on Espresso



Puppeteer

```
/* Puppeteer */

// Setup browser
const browser = ...
const page = await browser.newPage();

// Action
await page.setViewport({width: 600, height: 1041});
await page.goto('https://coffee-cart.app');

const coffee = await page.$('[data-test="Espresso"]');
await coffee.click();
```

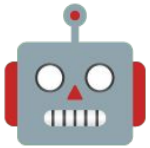
Full demo: bit.ly/wbidi-demo

Puppeteer

```
/* Puppeteer */

// Setup browser
const browser = ...
const page = await browser.newPage();

// Action
await page.setViewport({width: 600, height: 1041});
await page.goto('https://coffee-cart.app');

const coffee = await page.$('[data-test="Espresso"]');
await coffee.click();
```

Full demo: bit.ly/wbidi-demo

CDP



Automation tools

1. Navigate to the page

```
{ command: 'Page.navigate',
  parameters: { url: 'https://coffee-cart.app/' }}
```

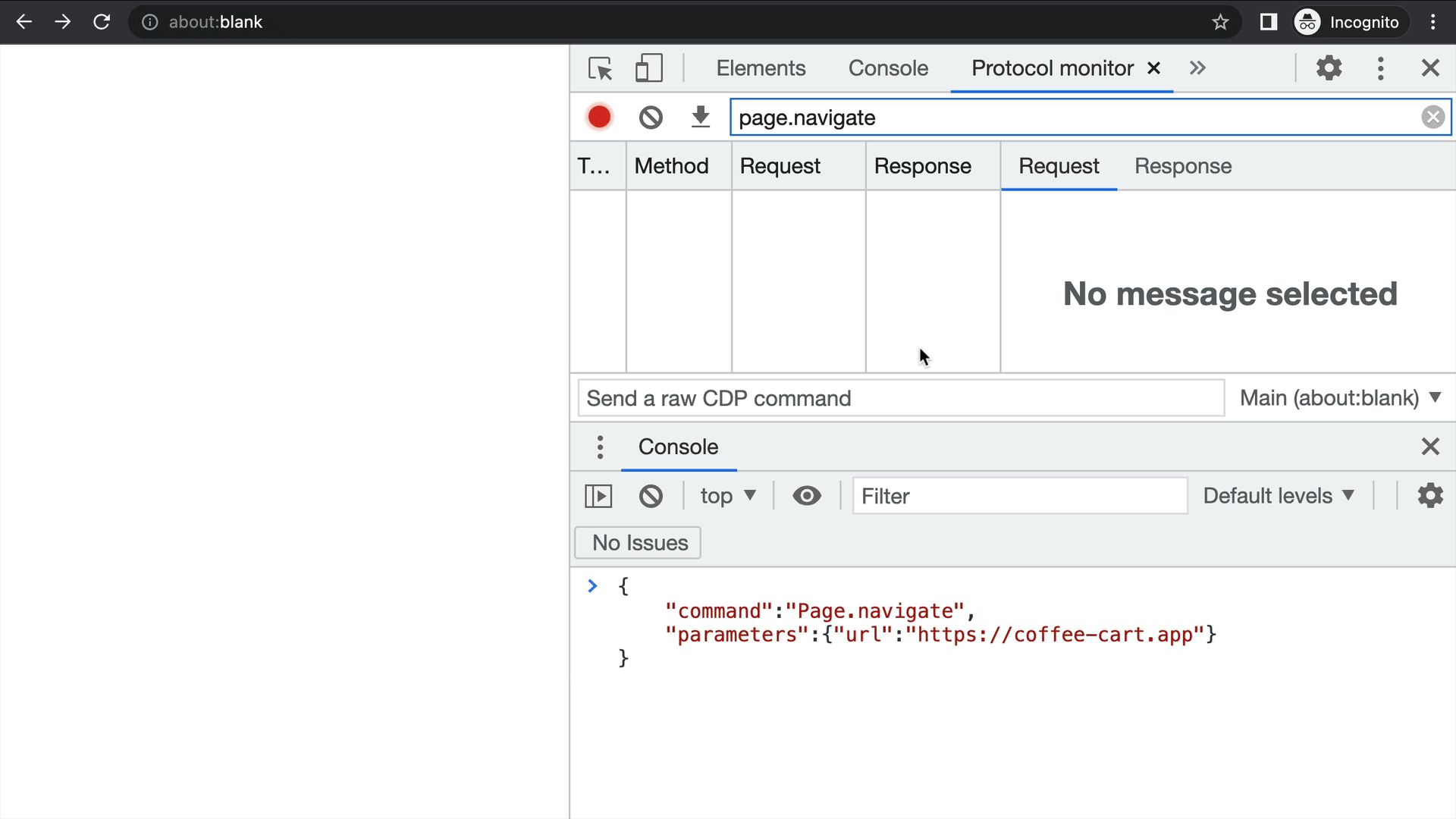
Chromium-based browsers

2. Find the coffee

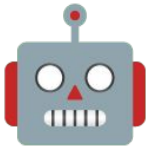
```
{ command: 'Runtime.evaluate',
  parameters: { expression:
    `document.querySelector('[data-test="Espresso"]')` }}
```

3. Click the coffee

```
{ command: 'Input.dispatchMouseEvent', parameters: {
    type: 'mousePressed', x: 10.34, y: 27.1, clickCount: 1 }}

{ command: 'Input.dispatchMouseEvent', parameters: {
    type: 'mouseReleased', x: 10.34, y: 27.1, clickCount: 1 }}
```


about:blank
Incognito
Elements
Console
Protocol monitor
page.navigate
T...
Method
Request
Response
Request
Response
No message selected
Send a raw CDP command
Main (about:blank)
Console
top
Filter
Default levels
No Issues
{
"command":"Page.navigate",
"parameters":{"url":"https://coffee-cart.app"}
}

CDP



Automation tools

1. Navigate to the page

```
{ command: 'Page.navigate',
  parameters: { url: 'https://coffee-cart.app/' }}
```

Chromium-based browsers

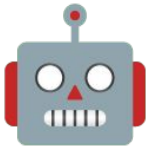
2. Find the coffee

```
{ command: 'Runtime.evaluate',
  parameters: { expression:
    `document.querySelector('[data-test="Espresso"]')` }}
```

3. Click the coffee

```
{ command: 'Input.dispatchMouseEvent', parameters: {
    type: 'mousePressed', x: 10.34, y: 27.1, clickCount: 1 }}

{ command: 'Input.dispatchMouseEvent', parameters: {
    type: 'mouseReleased', x: 10.34, y: 27.1, clickCount: 1 }}
```

CDP



Automation tools

1. Navigate to the page

```
{ command: 'Page.navigate',
  parameters: { url: 'https://coffee-cart.app/' }}
```

Chromium-based browsers

2. Find the coffee

```
{ command: 'Runtime.evaluate',
  parameters: { expression:
    `document.querySelector('[data-test="Espresso"]')` }}
```

3. Click the coffee

```
{ command: 'Input.dispatchMouseEvent', parameters:
    type: 'mousePressed', x: 10.34, y: 27.1, clickC

{ command: 'Input.dispatchMouseEvent', parameters:
    type: 'mouseReleased', x: 10.34, y: 27.1, click
```

Fast, bi-directional

CDP

Low level control

Intercept network requests

Simulate device mode

Simulate geolocation

Get console messages

CDP

Fast, bi-directional messaging

Provide **low level** control

Chromium-based browsers

Non-standard

Complex, built for debugging

CDP

Fast, bi-directional messaging

Provide **low level** control

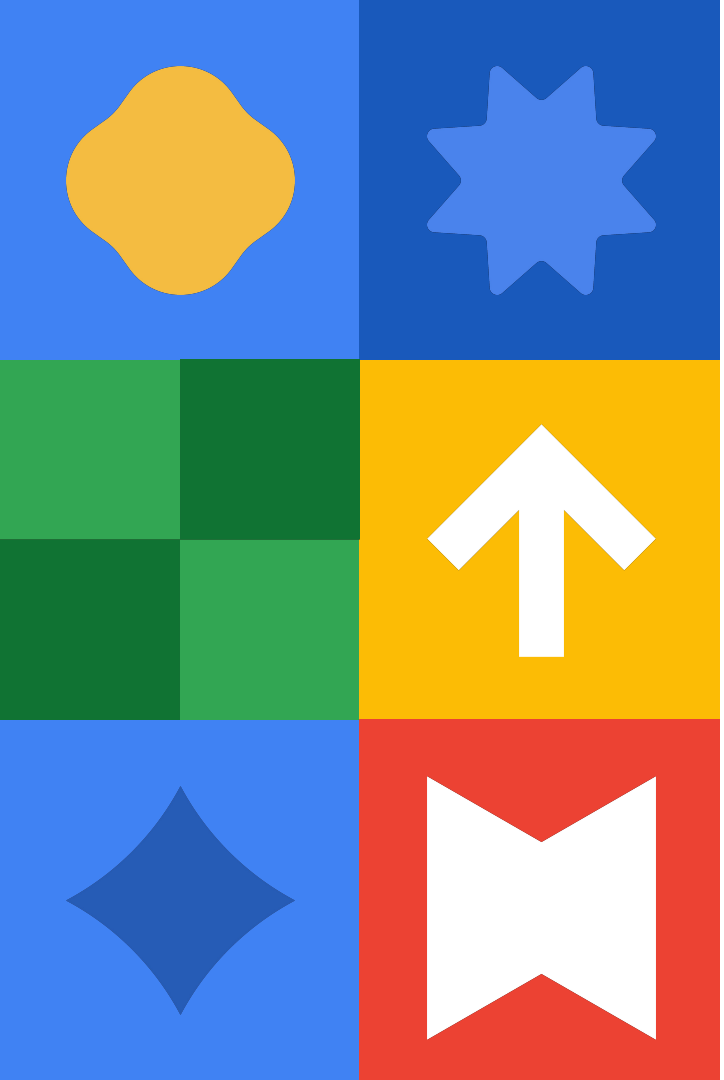
WebDriver
“Classic”

Best **cross-browser** support

W3C standard

Built for **testing**

What if we only take the good parts from both protocols?



WebDriver BiDi

WebDriver+CDP=Yes

Fourth on my list of browser automation techniques

WebDriver BiDi

Fast, bi-directional messaging

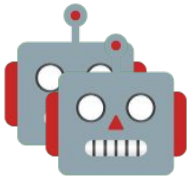
Provide **low level** control

Best **cross-browser** support

W3C standard

Built for **testing**

WebDriver BiDi



Any automation tools

Issue commands through **WebDriver BiDi protocol**

Fast connection via **WebSocket**

Any browsers or drivers

WebDriver BiDi

Complements WebDriver

WebDriver BiDi will function as an addition to WebDriver (Classic). There is no need for a big migration!

Succeeds CDP

WebDriver BiDi should succeed - with some caveats - the direct usage of CDP in automation libraries.

Combines benefits

WebDriver BiDi combines the benefits of CDP and WebDriver, to provide a stronger browser automation package.

WebDriver BiDi

Collaboration

Browsers

Test automation frameworks

Test infrastructure providers

BrowserStack

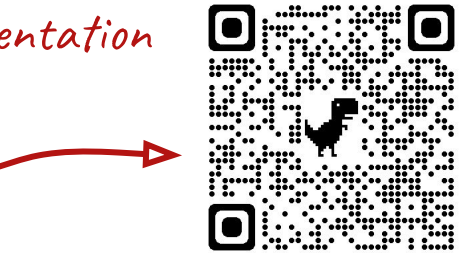
SauceLabs

Now the bad news

WebDriver BiDi

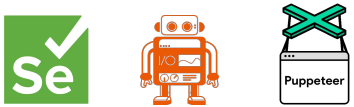
🚧 Work in progress

Track browser implementation progress here



Now the good news

WebDriver BiDi



🎉 Initial support in automation tools

bit.ly/wbidi-demo

bit.ly/selenium-initial-bidi-support
bit.ly/webdriverio-initial-bidi-support
bit.ly/pptr-initial-bidi-support

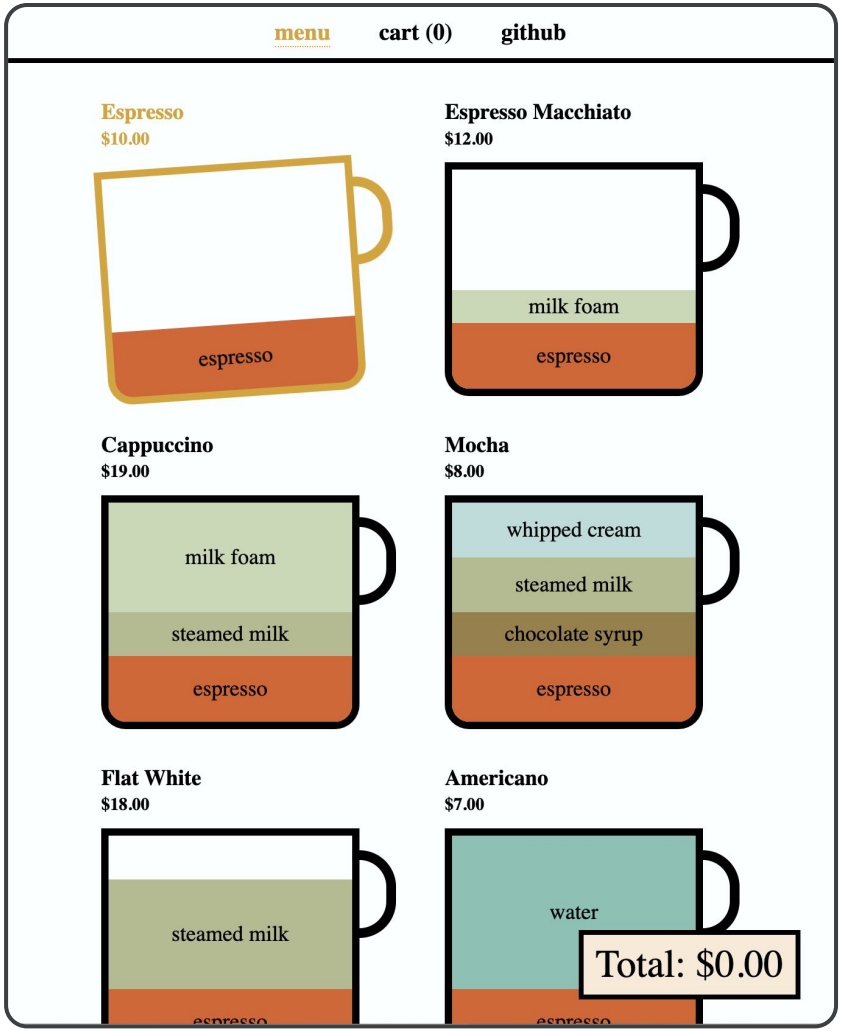
WebDriver BiDi

Path	Chrome 114 Linux 20.04 c56aec6 May 4, 2023
browsing_context/	267 / 290
errors/	2 / 2
input/	0 / 99
log/	59 / 65
network/	53 / 109
script/	520 / 656
session/	84 / 84
Subtest Total	985 / 1305

As of May 04, 2023: bit.ly/wbidi-dashboard

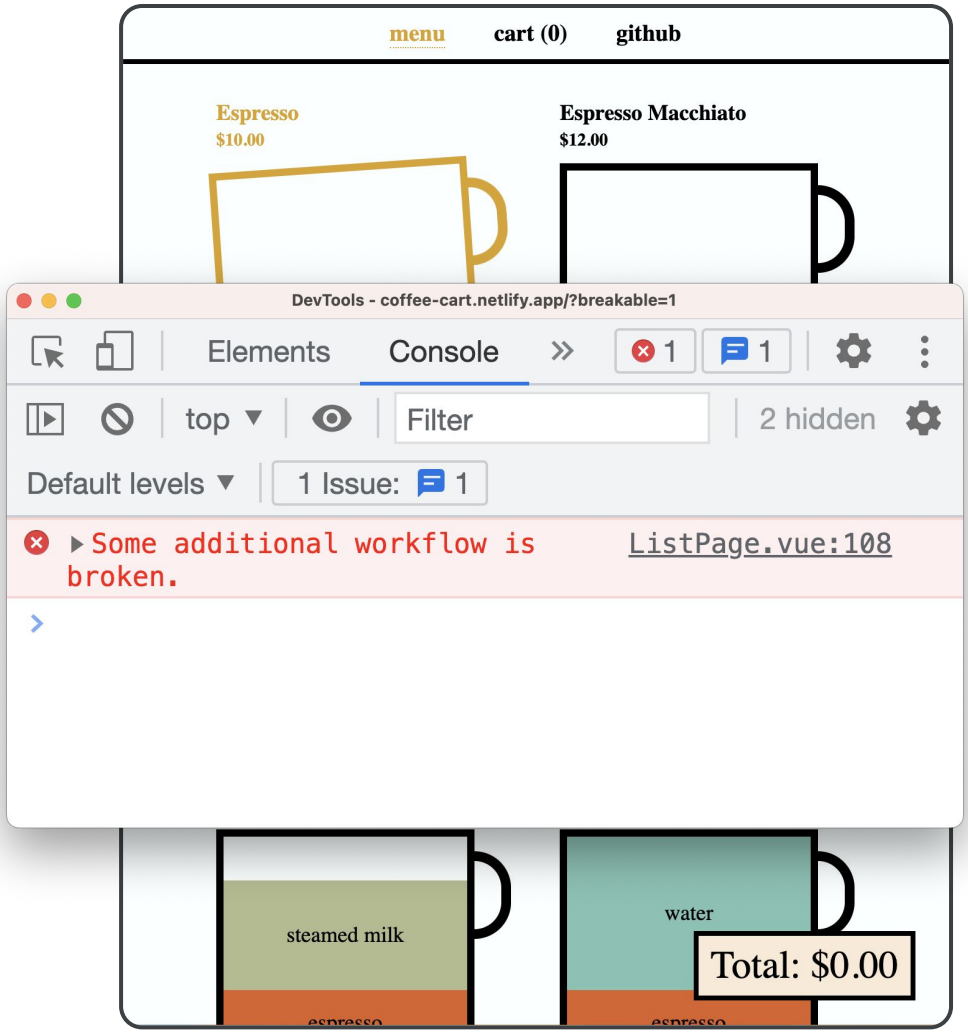
WebDriver BiDi

1. Navigate to the page
2. Click on Espresso



WebDriver BiDi

1. Navigate to the page
2. Click on Espresso
3. Catch console errors



WebdriverIO

```
/* WebdriverIO */

// Setup browser
const browser = await remote({
  capabilities: { browserName: 'chrome', webSocketUrl: true }
});

// TODO: Monitor log messages
...

// Action
...
```

Full demo: bit.ly/wbidi-demo

WebdriverIO

```
/* WebdriverIO */

// Setup browser
const browser = await remote({
  capabilities: { browserName: 'chrome', webSocketUrl: true }
});

// TODO: Monitor log messages
...

// Action
...
```

Full demo: bit.ly/wbidi-demo

WebdriverIO

```
/* WebdriverIO */

// Setup browser
const browser = await remote({
  capabilities: { browserName: 'firefox', webSocketUrl: true }
});

// TODO: Monitor log messages
...

// Action
...
```

Full demo: bit.ly/wbidi-demo

WebdriverIO

```
/* WebdriverIO */

// Setup browser
const browser = await remote({
  capabilities: { browserName: 'firefox', webSocketUrl: true }
});

// TODO: Monitor log messages
...

// Action
...
```

Full demo: bit.ly/wbidi-demo

WebdriverIO

```
// Monitor log messages
await browser.send({
  method: 'session.subscribe',
  params: { events: ['log.entryAdded'] }
});

browser.on('message', (data) => {
  const txt = JSON.parse(data).params.text;
  assert.fail(`Unexpected console message received: ${txt}`);
});

// Action
...
```

Full demo: bit.ly/wbidi-demo

WebdriverIO

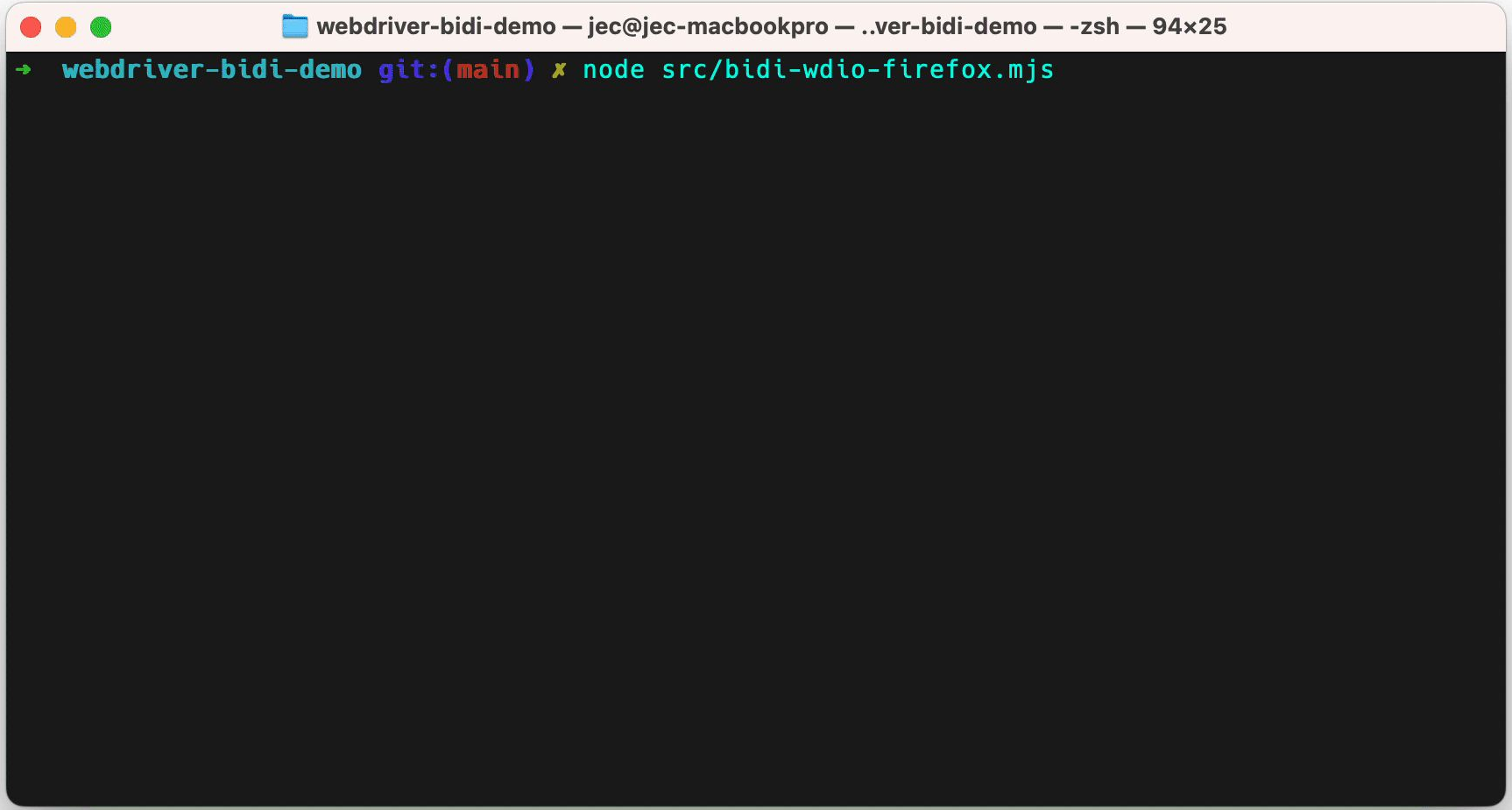
```
// Monitor log messages
await browser.send({
  method: 'session.subscribe',
  params: { events: ['log.entryAdded'] }
});

browser.on('message', (data) => {
  const txt = JSON.parse(data).params.text;
  assert.fail(`Unexpected console message received: ${txt}`);
});

// Action
...
```

Full demo: bit.ly/wbidi-demo

WebdriverIO

```
// Monitor log messages
await browser.send({
  method: 'session.subscribe',
  params: { events: ['log.entryAdded'] }
});

browser.on('message', (data) => {
  const txt = JSON.parse(data).params.text;
  assert.fail(`Unexpected console message received: ${txt}`);
});

// Action
...
```

Full demo: bit.ly/wbidi-demo


webdriver-bidi-demo — jec@jec-macbookpro — ..ver-bidi-demo — -zsh — 94×25
➜ webdriver-bidi-demo git:(main) ✗ node src/bidi-wdio-firefox.mjs

Puppeteer

```
/* Puppeteer */

// Setup browser
const browser = await puppeteer.launch({ protocol: 'webDriverBiDi' });
const page = ...

// Monitor log messages
page.on('console', message => {
  const txt = message.text();
  assert.fail(`Unexpected console message received: ${txt}`);
});

// Action
...
```

Full demo: bit.ly/wbidi-demo

Puppeteer

```
/* Puppeteer */

// Setup browser
const browser = await puppeteer.launch({ protocol: 'webDriverBiDi' });
const page = ...

// Monitor log messages
page.on('console', message => {
  const txt = message.text();
  assert.fail(`Unexpected console message received: ${txt}`);
});

// Action
...
```

Full demo: bit.ly/wbidi-demo

Puppeteer

```
/* Puppeteer */

// Setup browser
const browser = await puppeteer.launch({ protocol: 'webDriverBiDi' });
const page = ...

// Monitor log messages
page.on('console', message => {
  const txt = message.text();
  assert.fail(`Unexpected console message received: ${txt}`);
});

// Action
...
```

Full demo: bit.ly/wbidi-demo

The wrap

The wrap

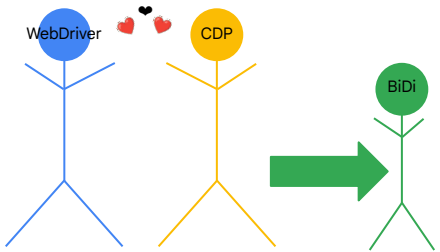
Browser automation is hard

There are multiple ways to automate a browser, each with its own benefits and drawbacks.

WebDriver BiDi to the rescue

WebDriver BiDi combines the benefits of the two most stable browser automation techniques: WebDriver and CDP.

Learn more at goo.gle/webdriver-bidi.



Try it out

WebDriver BiDi capabilities are shipped incrementally, you can try them out already if your browser and test automation library already supports it.

Super important!!1!!11!

The End

Three Ways to Automate Your Browser, and Why We Are Adding a Fourth: WebDriver BiDi

Michael Hablich @mhablich
Product Manager on the Google Chrome team