

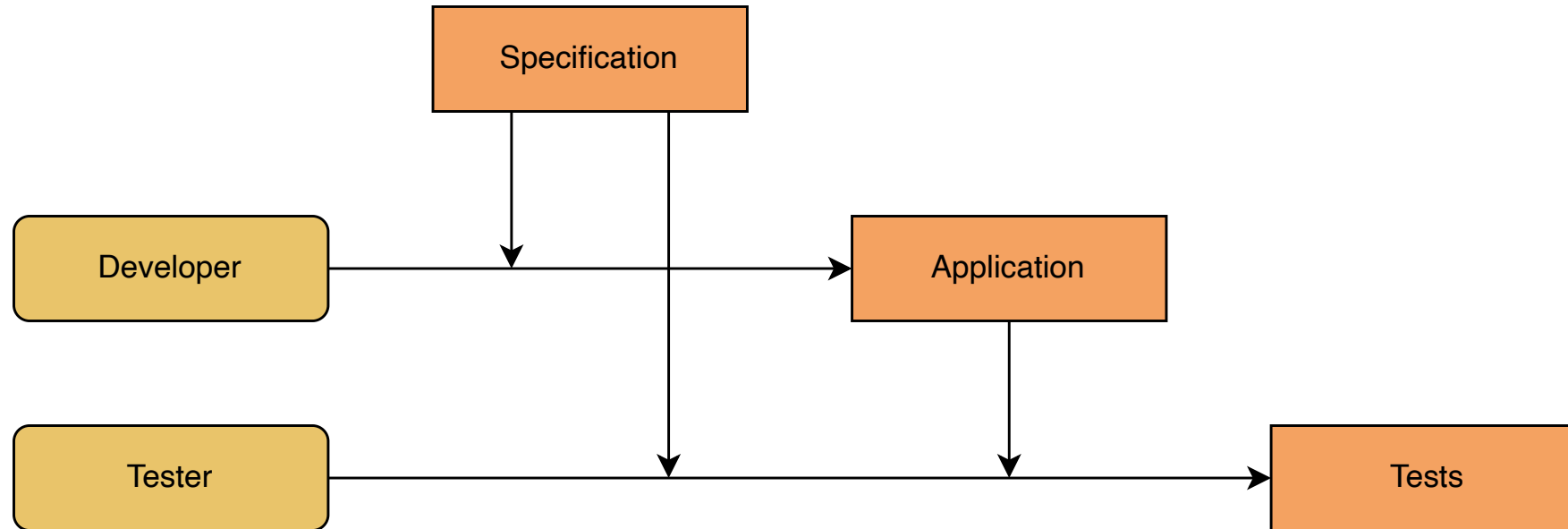
Testing unknown applications with Selenium & WebDriver BiDirectional API

Tomasz Wrona

Tech Lead & Machine Learning Engineer

Global App Testing

A story about testing web applications



Application works as intended?



#recipeoftheday

[home](#)

[recipes](#)

[about](#)

[contact](#)

[login](#)

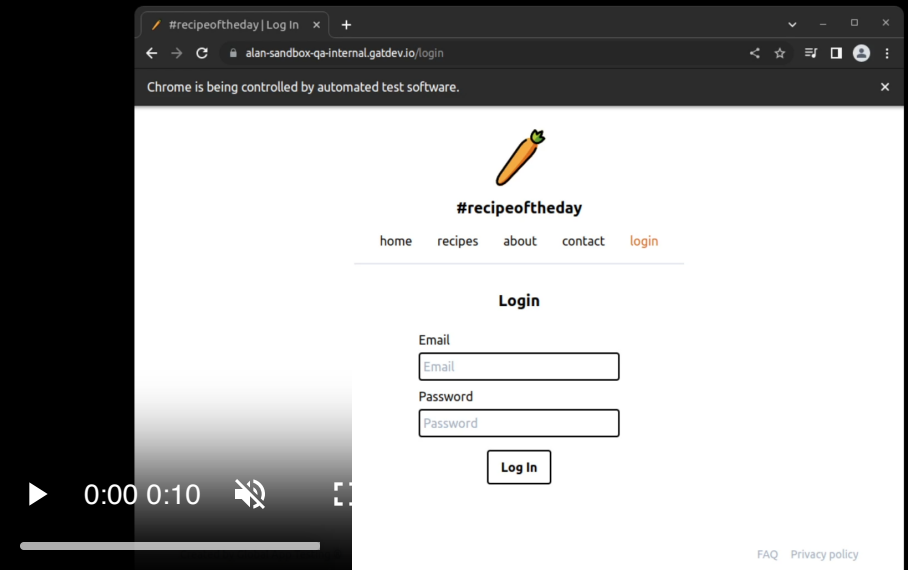
Login

Email

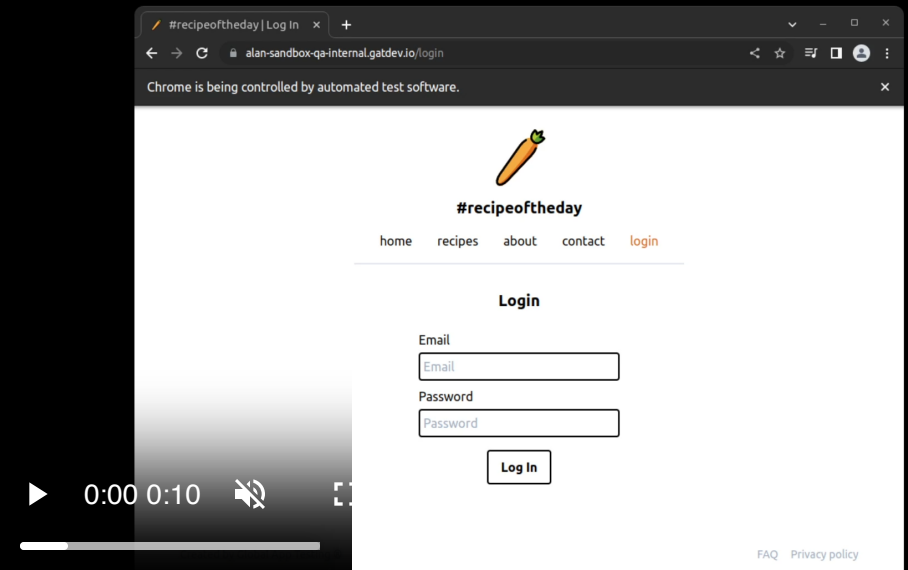
Password

Log In

```
1 # Arrange
2 def test_user_is_logged_in_after_filling_login_form(
3     driver: WebDriver, application_url: str
4 ):
5     driver.get(application_url)
6     # Act
7     email = driver.find_element(By.XPATH, "//form/div[1]/label/input")
8     password = driver.find_element(By.XPATH, "//form/div[2]/label/input")
9     submit = driver.find_element(By.XPATH, "//form/button")
10    (
11        ActionChains(driver)
12        .send_keys_to_element(email, "alan@globalapptesting.com")
13        .send_keys_to_element(password, "Alan12345!")
14        .click(submit)
15        .perform()
16    )
17    try:
18        # Assert
19        WebDriverWait(driver, timeout=5).until(
20            expected_conditions.presence_of_element_located(
21                (By.LINK_TEXT, "account")
22            ),
23            "The user hasn't been logged in.",
24        )
25    except TimeoutException as e:
26        pytest.fail(e.msg)
```



```
1 # Arrange
2 def test_user_is_logged_in_after_filling_login_form(
3     driver: WebDriver, application_url: str
4 ):
5     driver.get(application_url)
6     # Act
7     email = driver.find_element(By.XPATH, "//form/div[1]/label/input")
8     password = driver.find_element(By.XPATH, "//form/div[2]/label/input")
9     submit = driver.find_element(By.XPATH, "//form/button")
10    (
11        ActionChains(driver)
12        .send_keys_to_element(email, "alan@globalapptesting.com")
13        .send_keys_to_element(password, "Alan12345!")
14        .click(submit)
15        .perform()
16    )
17    try:
18        # Assert
19        WebDriverWait(driver, timeout=5).until(
20            expected_conditions.presence_of_element_located(
21                (By.LINK_TEXT, "account")
22            ),
23            "The user hasn't been logged in.",
24        )
25    except TimeoutException as e:
26        pytest.fail(e.msg)
```

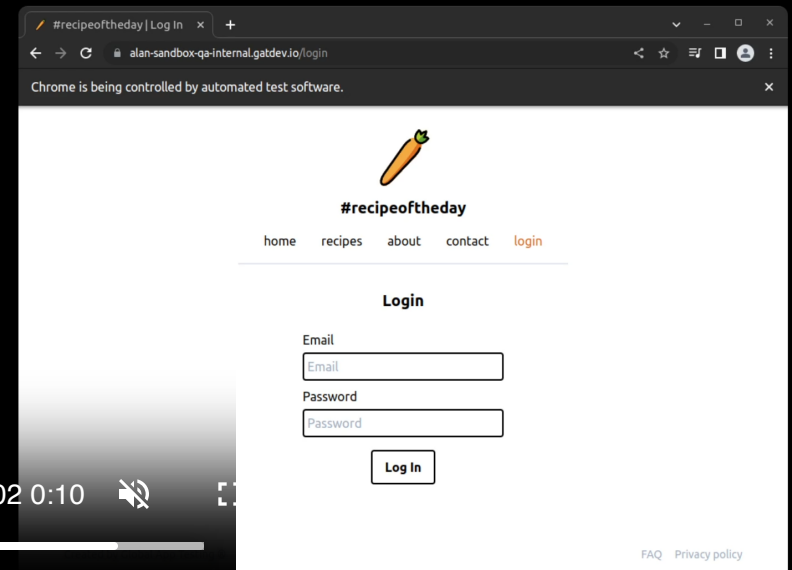


```

1 # Arrange
2 def test_user_is_logged_in_after_filling_login_form(
3     driver: WebDriver, application_url: str
4 ):
5     driver.get(application_url)
6     # Act
7     email = driver.find_element(By.XPATH, "//form/div[1]/label/input")
8     password = driver.find_element(By.XPATH, "//form/div[2]/label/input")
9     submit = driver.find_element(By.XPATH, "//form/button")
10    (
11        ActionChains(driver)
12        .send_keys_to_element(email, "alan@globalapptesting.com")
13        .send_keys_to_element(password, "Alan12345!")
14        .click(submit)
15        .perform()
16    )
17    try:
18        # Assert
19        WebDriverWait(driver, timeout=5).until(
20            expected_conditions.presence_of_element_located(
21                (By.LINK_TEXT, "account")
22            ),
23            "The user hasn't been logged in.",
24        )
25    except TimeoutException as e:
26        pytest.fail(e.msg)

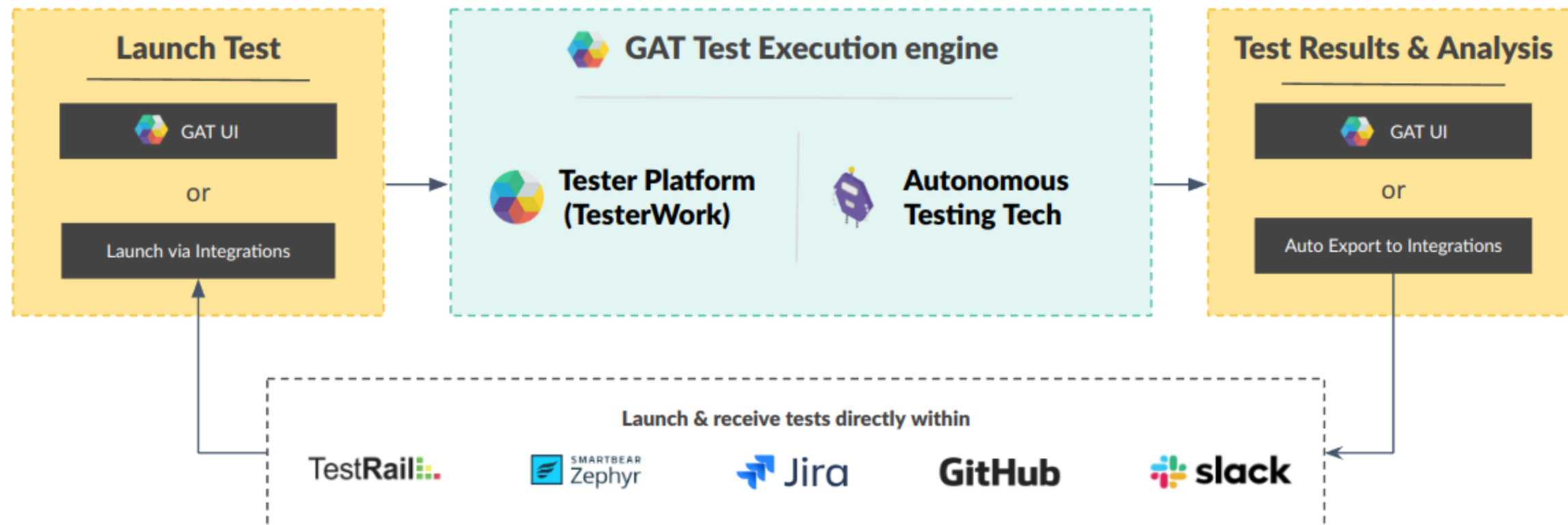
```

▶ 0:02 0:10 🔊 🔍

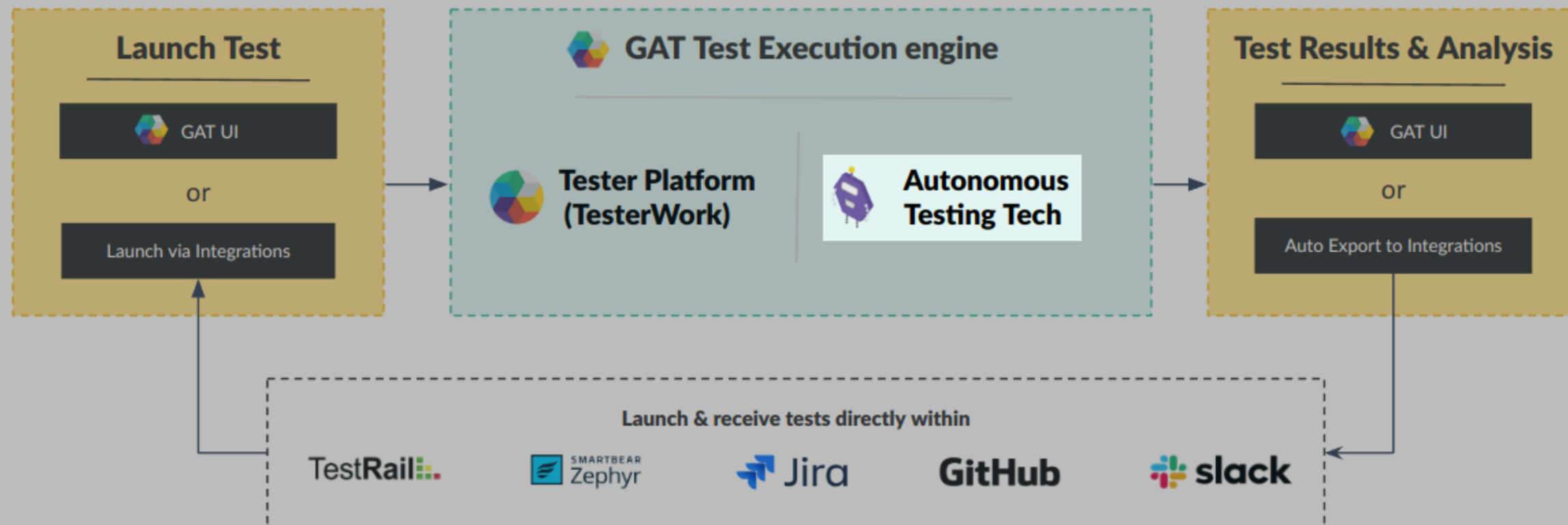


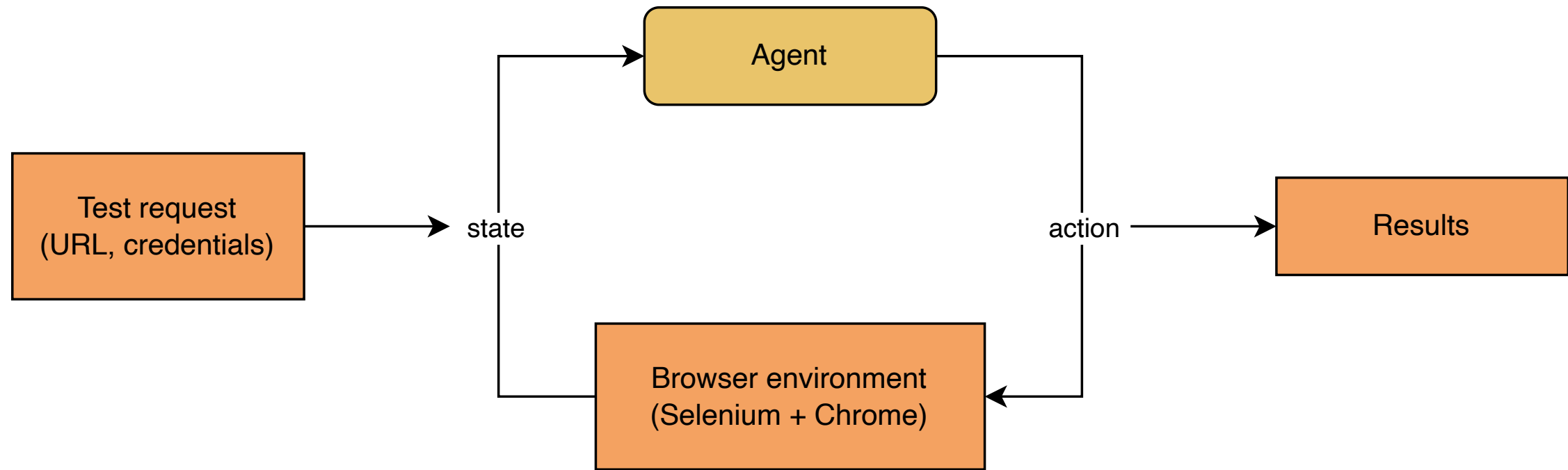
Before we move on...

How does GAT test execution engine work?

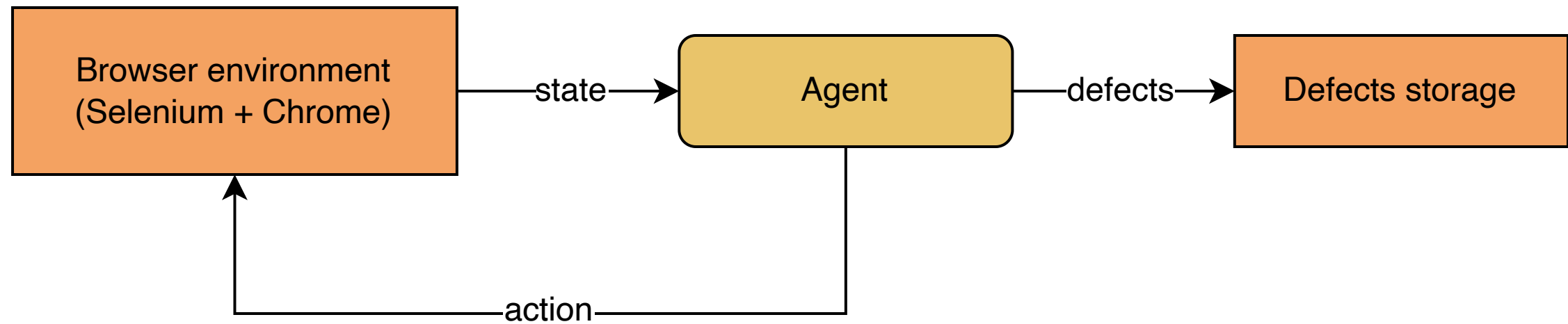


How does GAT test execution engine work?





AI folks, looks familiar?



What happens when you ask Selenium to click a button?

§ 12.5.1 *Element Click*

HTTP Method	URI Template
POST	<i>/session/{session id}/element/{element id}/click</i>

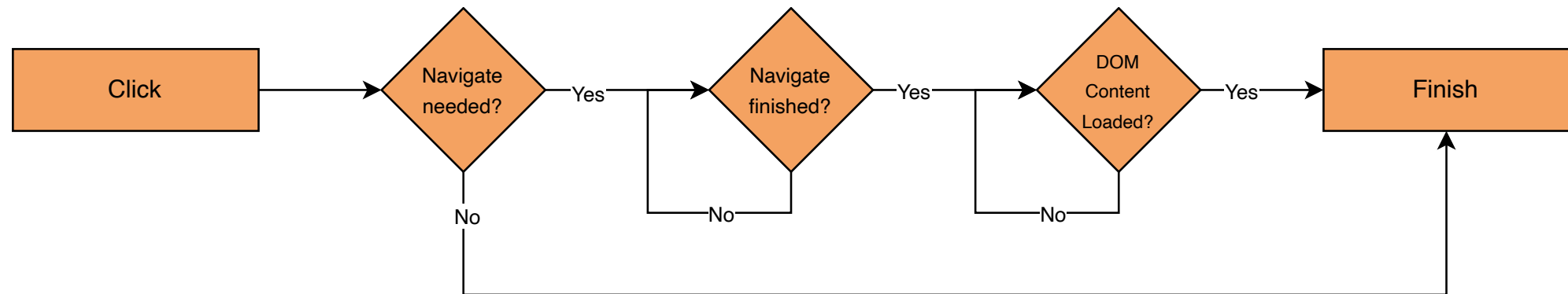
NOTE

The [Element Click](#) command [scrolls into view](#) the [element](#) if it is not already [pointer-interactable](#), and clicks its [in-view center point](#).

If the element's [center point](#) is [obscured](#) by another element, an [element click intercepted error](#) is returned. If the element is outside the [viewport](#), an [element not interactable error](#) is returned.

The [remote end steps](#) are:

1. If the [current browsing context](#) is [no longer open](#), return [error](#) with [error code](#) [no such window](#).
2. [Handle any user prompts](#) and return its value if it is an [error](#).
3. Let *element* be the result of [trying](#) to [get a known connected element](#) with argument *element id*.
4. If the *element* is an [input](#) element in the [file upload state](#) return [error](#) with [error code](#) [invalid argument](#).
5. [Scroll into view](#) the *element*'s [container](#).
6. If *element*'s [container](#) is still not [in view](#), return [error](#) with [error code](#) [element not interactable](#).



And that was just clicking...



|



Google Search

I'm Feeling Lucky

Google offered in: [polski](#)

Full page demo

1a - Infinite Scroll full page demo

Hi! This demo will show off several key features of Infinite Scroll.

- Full page scrolling
- Changing browser URL and history. Watch how the URL changes as you scroll. Try refreshing on a changed page.
- Embeds like CodePens and Tweets have their `<script>`s loaded and embeds rendered

The code looks like this:

```
<div class="article-feed">
  <!-- .articles will be added to .article-feed -->
  <article class="article">...</article>
  <article class="article">...</article>
  ...
</div>

<!-- status elements -->
<div class="scroller-status">
  <div class="infinite-scroll-request loader-ellips">
    ...
  </div>
```


What's happening under the hood?

Async requests!

#recipeoftheday | Home

alan-sandbox-qa-internal.gatdev.io


Incognito



#recipeoftheday

[home](#) [recipes](#) [about](#) [contact](#) [login](#)

Welcome to #recipeoftheday



Explore recipes

Created by Global App Testing ©

[FAQ](#) [Privacy policy](#)

Elements Console Sources Network

Filter

All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other

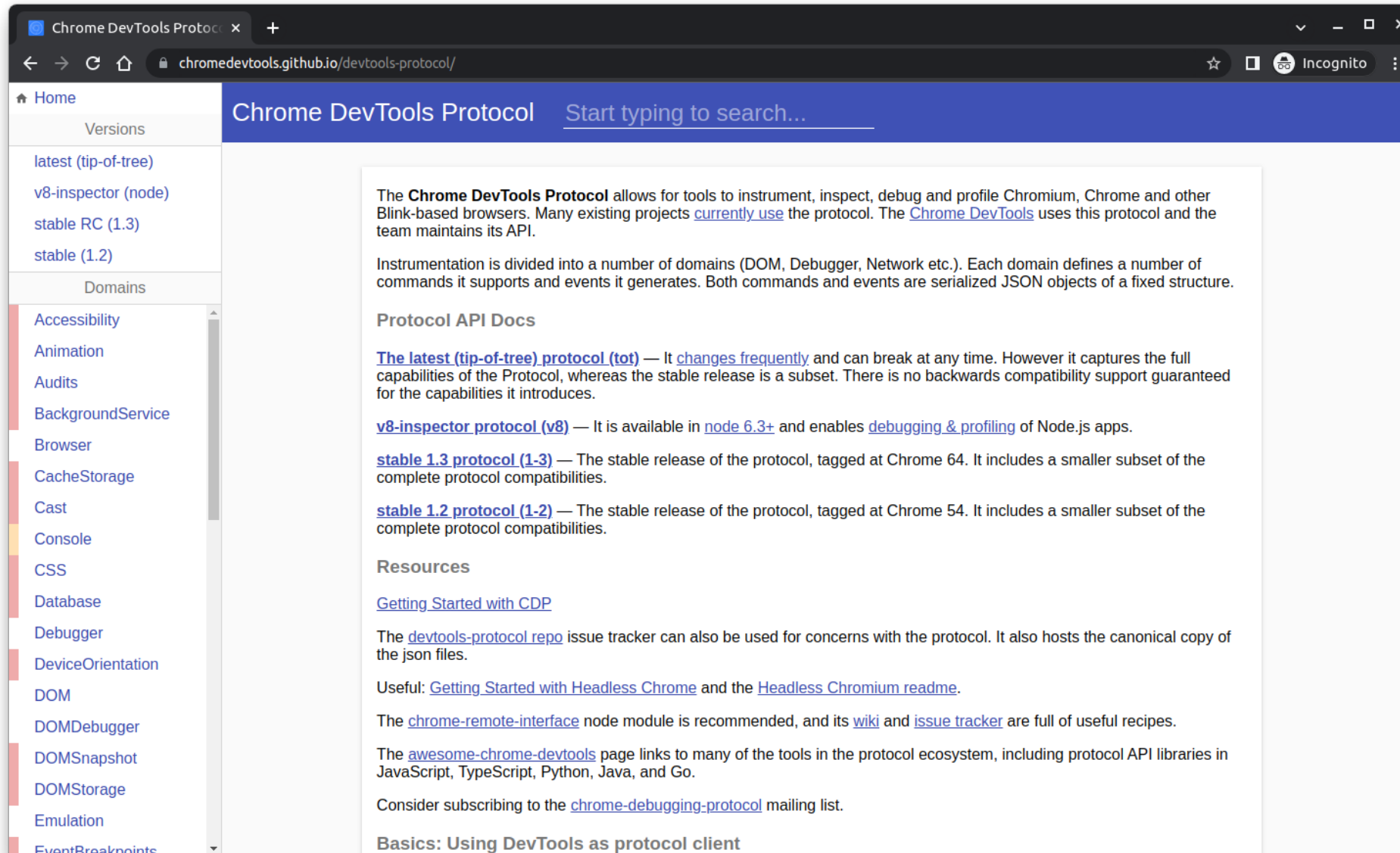
Name	Sta..	Type	Initiator	Size	Time	Waterfall
alan-sandbox-qa...	304	do...	Other	34...	61 ...	
ed547189269be...	200	sty...	(index)	(m...	0 ms	
main-1684f93a9...	200	script	(index)	(m...	0 ms	
webpack-eb080...	200	script	(index)	(m...	0 ms	
framework.03c1...	200	script	(index)	(m...	0 ms	
commons.59db...	200	script	(index)	(m...	0 ms	
fdc338935ae2ef...	200	script	(index)	(m...	0 ms	
a10e5aa71f79a...	200	script	(index)	(m...	0 ms	
025d993ffb707...	200	script	(index)	(m...	0 ms	
_app-2ee49954...	200	script	(index)	(m...	0 ms	
index-55913dd8...	200	script	(index)	(pr...	2 ms	
_buildManifest.js	200	script	(index)	(m...	0 ms	
_ssgManifest.js	200	script	(index)	(m...	0 ms	
carrot.svg	304	svg...	(index)	47...	94 ...	
cook.png	304	png	(index)	47...	61 ...	
login	200	fetch	fdc338...	31...	63 ...	
getCurrentUser	200	fetch	fdc338...	31...	59 ...	
favicon.ico	200	x-ic...	Other	(di...	2 ms	
recipes-5bf8cea...	200	jav...	main-1...	(m...	0 ms	
about-bbe9feef...	200	jav...	main-1...	(m...	0 ms	
contact-7b7fe1...	200	jav...	main-1...	(m...	0 ms	
index-55913dd8...	200	jav...	main-1...	(di...	2 ms	

23 requests | 1.9 kB transferred | 543 kB resources | Finish: 235 ms | DOMContentLoaded: 88

Solution: track and await requests

Source of requests data

1. Monkeypatching XMLHttpRequest and Fetch API
2. Custom Chrome plugin
3. Chrome DevTools Protocol



Chrome DevTools Protocol

chromedevtools.github.io/devtools-protocol/tot/Network/

Incognito (3)

Home

Versions

latest (tip-of-tree)

v8-inspector (node)

stable RC (1.3)

stable (1.2)

Domains

Accessibility

Animation

Audits

BackgroundService

Browser

CacheStorage

Cast

Console

CSS

Database

Debugger

DeviceOrientation

DOM

DOMDebugger

DOMSnapshot

DOMStorage

Emulation

EventBreakpoints

Fetch

HeadlessExperimental

HeapProfiler

Chrome DevTools Protocol

Start typing to search...

Network.setAttachDebugStack

EXPERIMENTAL

Network.setBlockedURLs

EXPERIMENTAL

Network.setBypassServiceWorker

EXPERIMENTAL

Network.takeResponseBodyForInterceptionAsStream

EXPERIMENTAL

Network.continueInterceptedRequest

EXPERIMENTAL

DEPRECATED

Network.setRequestInterception

EXPERIMENTAL

DEPRECATED

Events

Network.dataReceived

Network.eventSourceMessageReceived

Network.loadingFailed

Network.loadingFinished

Network.requestServedFromCache

Network.requestWillBeSent

Network.responseReceived

Network.webSocketClosed

Network.webSocketCreated

Network.webSocketFrameError

Network.webSocketFrameReceived

Network.webSocketFrameSent

Network.webSocketHandshakeResponseReceived

Network.webSocketWillSendHandshakeRequest

Network.webTransportClosed

Network.webTransportConnectionEstablished

Network.webTransportCreated

Network.reportingApiEndpointsChangedForOrigin

EXPERIMENTAL

Network.reportingApiReportAdded

EXPERIMENTAL

Network.reportingApiReportUpdated

EXPERIMENTAL

Network.requestWillBeSentExtraInfo

EXPERIMENTAL

Network.resourceChangedPriority

EXPERIMENTAL

Network.responseReceivedExtraInfo

EXPERIMENTAL

Network.signedExchangeReceived

EXPERIMENTAL

Network.subresourceWebBundleInnerResponseError

EXPERIMENTAL

Network.subresourceWebBundleInnerResponseParsed

EXPERIMENTAL

Network.subresourceWebBundleMetadataError

EXPERIMENTAL

Network.subresourceWebBundleMetadataReceived

EXPERIMENTAL

Network.trustTokenOperationDone

EXPERIMENTAL

Network.requestIntercepted

EXPERIMENTAL

DEPRECATED

Types

Network.BlockedReason

Network.CachedResource

Selenium built-in Chrome DevTools Protocol support

What about events? WebDriver BiDi API!

W3C Editor's Draft

TABLE OF CONTENTS

1

Introduction

2

Infrastructure

3

Protocol

3.1

Definition

3.2

Session

3.3

Modules

3.4

Commands

3.5

Events

4

Transport

4.1

Establishing a Connection

5

Sandboxed Script Execution

5.1

Sandbox Realms

5.2

Sandbox Proxy Objects

5.3

SandboxWindowProxy

6

Common Data Types

6.1

Reference

6.2

Protocol Value

6.2.1

Primitive Protocol Value

6.2.1.1

Serialization

6.2.1.2

Deserialization

6.2.2

Local Value

6.2.2.1

Deserialization

6.2.3

Remote Value

6.2.3.1

Serialization

7

Modules

7.1

The session Module

7.1.1

Definition

7.1.2

Types

7.1.2.1

The session CapabilitiesRequest Type

WebDriver BiDi

Editor's Draft, 26 October 2022



▼ More details about this document

This version:

<https://w3c.github.io/webdriver-bidi/>

Issue Tracking:

[GitHub](#)

[Inline In Spec](#)

Copyright © 2022 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and permissive document license rules apply.

Abstract

This document defines the BiDirectional WebDriver Protocol, a mechanism for remote control of user agents.

Status of this document

This is a public copy of the editors' draft. It is provided for discussion only and may change at any moment. Its publication here does not imply endorsement of its contents by W3C. Don't cite this document other than as work in progress.

[GitHub Issues](#) are preferred for discussion of this specification.

This document was produced by the [Browser Testing and Tools Working Group](#).

This document was produced by a group operating under the [W3C Patent Policy](#). W3C maintains a [public list of any patent disclosures](#) made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains [Essential Claim\(s\)](#) must disclose the information in accordance with [section 6 of the W3C Patent Policy](#).

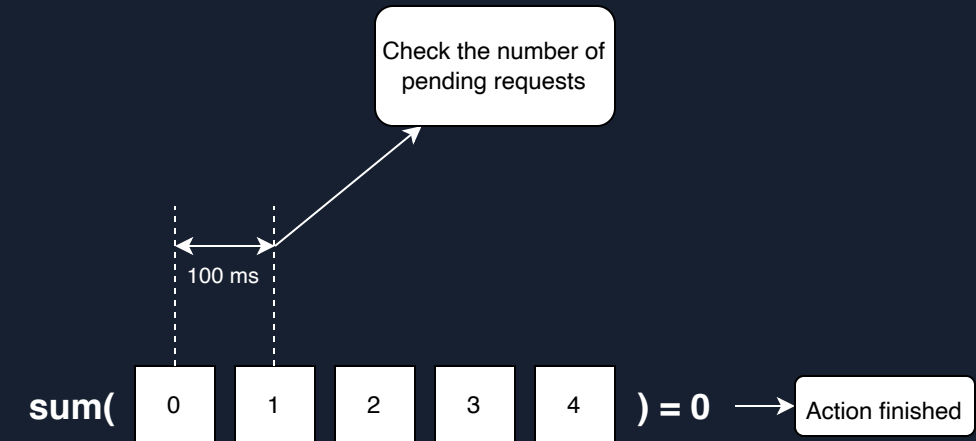
This document is governed by the [2 November 2021 W3C Process Document](#).

Selenium supports WebDriver BiDi with Trio

```

1  async def execute_and_await(
2      driver: WebDriver,
3      action: Callable[[], None],
4      timeout: float = 10.0, # [s]
5      num_checks: int = 10,
6      window_length: float = 1.0, # [s]
7  ):
8      requests: dict[str, bool] = {}
9
10     async def request_tracker(
11         session: CdpSession,
12         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
13     ):
14         ...
15
16     ready = trio.Event()
17
18     async def await_pending_requests(
19         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
20     ):
21         ...
22
23     async with driver.bidi_connection() as bidi_session:
24         ...

```



```

1  async def execute_and_await(
2      driver: WebDriver,
3      action: Callable[[], None],
4      timeout: float = 10.0, # [s]
5      num_checks: int = 10,
6      window_length: float = 1.0, # [s]
7  ):
8      requests: dict[str, bool] = {}
9
10     async def request_tracker(
11         session: CdpSession,
12         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
13     ):
14         ...
15
16     ready = trio.Event()
17
18     async def await_pending_requests(
19         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
20     ):
21         ...
22
23     async with driver.bidi_connection() as bidi_session:
24         ...

```

```

1  requests: dict[str, bool] = {}
2
3  async def request_tracker(
4      session: CdpSession,
5      task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
6  ):
7      task_status.started()
8      async for event in session.listen(
9          RequestWillBeSent, LoadingFailed, LoadingFinished
10     ):
11         match event:
12             case RequestWillBeSent(request_id=request_id, request=request):
13                 LOG.debug("New request: %s", request)
14                 requests[request_id] = True
15             case LoadingFailed(request_id=request_id) | LoadingFinished(
16                 request_id=request_id
17             ):
18                 LOG.debug("Request finalized: %s", request_id)
19                 requests[request_id] = False

```

```

1  async def execute_and_await(
2      driver: WebDriver,
3      action: Callable[[], None],
4      timeout: float = 10.0, # [s]
5      num_checks: int = 10,
6      window_length: float = 1.0, # [s]
7  ):
8      requests: dict[str, bool] = {}
9
10     async def request_tracker(
11         session: CdpSession,
12         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
13     ):
14         ...
15
16     ready = trio.Event()
17
18     async def await_pending_requests(
19         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
20     ):
21         ...
22
23     async with driver.bidi_connection() as bidi_session:
24         ...

```

```

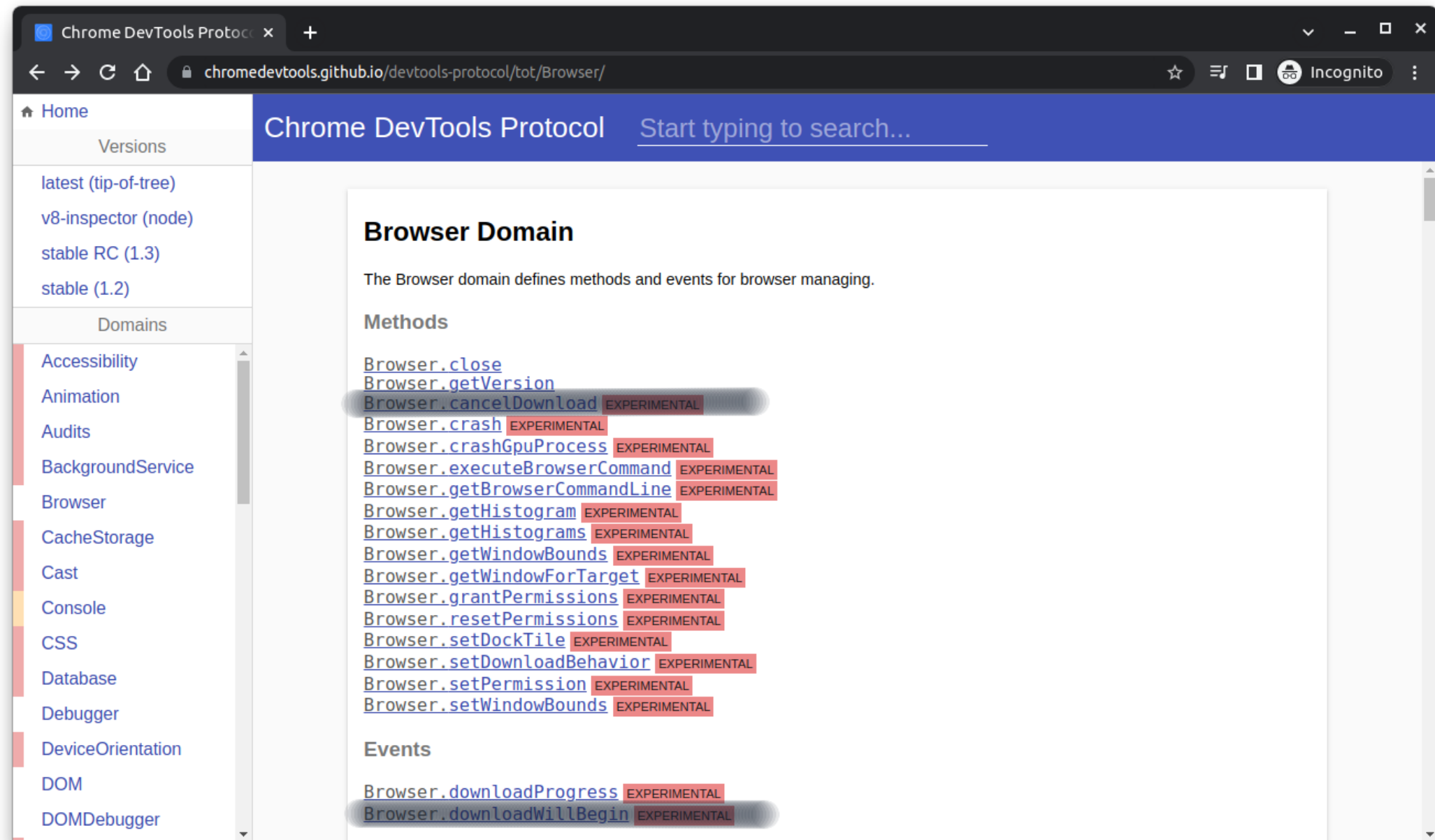
1  ready = trio.Event()
2
3  async def await_pending_requests(
4      task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,
5  ):
6      checks = deque(range(num_checks), maxlen=num_checks)
7      check_interval = window_length // max(num_checks, 1)
8      task_status.started()
9      while True:
10         num_pending_requests = sum(requests.values())
11         checks.append(num_pending_requests)
12         status = sum(checks) == 0
13         if status:
14             ready.set()
15             break
16         await trio.sleep(check_interval)

```

```
1  async def execute_and_await(  
2      driver: WebDriver,  
3      action: Callable[[], None],  
4      timeout: float = 10.0, # [s]  
5      num_checks: int = 10,  
6      window_length: float = 1.0, # [s]  
7  ):  
8      requests: dict[str, bool] = {}  
9  
10     async def request_tracker(  
11         session: CdpSession,  
12         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,  
13     ):  
14         ...  
15  
16     ready = trio.Event()  
17  
18     async def await_pending_requests(  
19         task_status: "trio._core._run._TaskStatusIgnored" = trio.TASK_STATUS_IGNORED,  
20     ):  
21         ...  
22  
23     async with driver.bidi_connection() as bidi_session:  
24         ...
```

```
1  async with driver.bidi_connection() as bidi_session:  
2      session: CdpSession = bidi_session.session  
3      await session.execute(network.enable())  
4      async with trio.open_nursery() as nursery:  
5          await nursery.start(request_tracker, session)  
6          await nursery.start(await_pending_requests)  
7          action()  
8          with trio.move_on_after(timeout) as cancel_scope:  
9              await ready.wait()  
10             if cancel_scope.cancelled_caught:  
11                 LOG.debug("Timeout waiting for pending requests.")  
12             nursery.cancel_scope.cancel()
```


What else we can find in CDP?



Summary

Need more interaction with the browser in your tests or your application itself? **Chrome DevTools Protocol** and **WebDriver BiDi API** might be the solution.

Thank you

<https://github.com/iamhatesz/pyconpl-2022>

