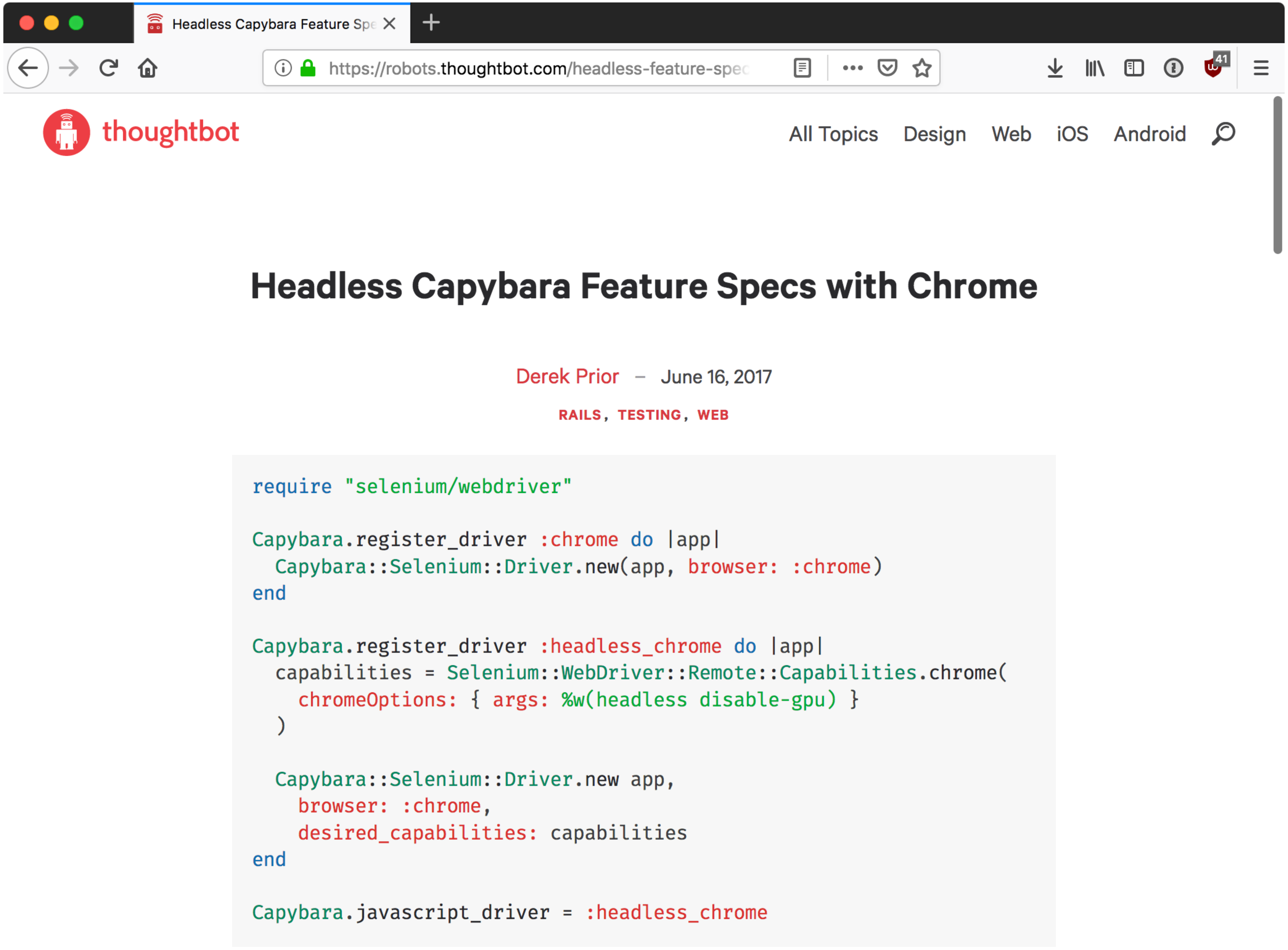
Pro Capybara Herding

Feature Testing Tips


Headless Capybara Feature Spe
https://robots.thoughtbot.com/headless-feature-spec
thoughtbot
All Topics
Design
Web
iOS
Android

Headless Capybara Feature Specs with Chrome

Derek Prior – June 16, 2017

RAILS, TESTING, WEB

```ruby
require "selenium/webdriver"

Capybara.register_driver :chrome do |app|
  Capybara::Selenium::Driver.new(app, browser: :chrome)
end

Capybara.register_driver :headless_chrome do |app|
  capabilities = Selenium::WebDriver::Remote::Capabilities.chrome(
    chromeOptions: { args: %w(headless disable-gpu) }
  )

  Capybara::Selenium::Driver.new app,
    browser: :chrome,
    desired_capabilities: capabilities
end

Capybara.javascript_driver = :headless_chrome
```

```ruby
it "is debuggable", :js do
  visit root_path

  binding.pry

  # ...
end
```

```
# spec/support/capybara.rb

+chrome_options = []
+
+if ENV.fetch("CHROME_HEADLESS", 1).to_i == 1
+  chrome_options << "headless"
+end

Capybara.register_driver :chrome do |app|
  capabilities = Selenium::WebDriver::Remote::Capabilities.chrome(
-   chromeOptions: { args: %w(headless disable-gpu) }
+   chromeOptions: { args: chrome_options }
  )

  Capybara::Selenium::Driver.new app,
    browser: :chrome,
    desired_capabilities: capabilities
end

Capybara.javascript_driver = :chrome
```

.env.test

```
CHROME_HEADLESS=1
```

.env.test

```
CHROME_HEADLESS=1
```

.env.test.local

```
#CHROME_HEADLESS=0
```

.env.test

```
CHROME_HEADLESS=1
```

.env.test.local

```
CHROME_HEADLESS=0
```

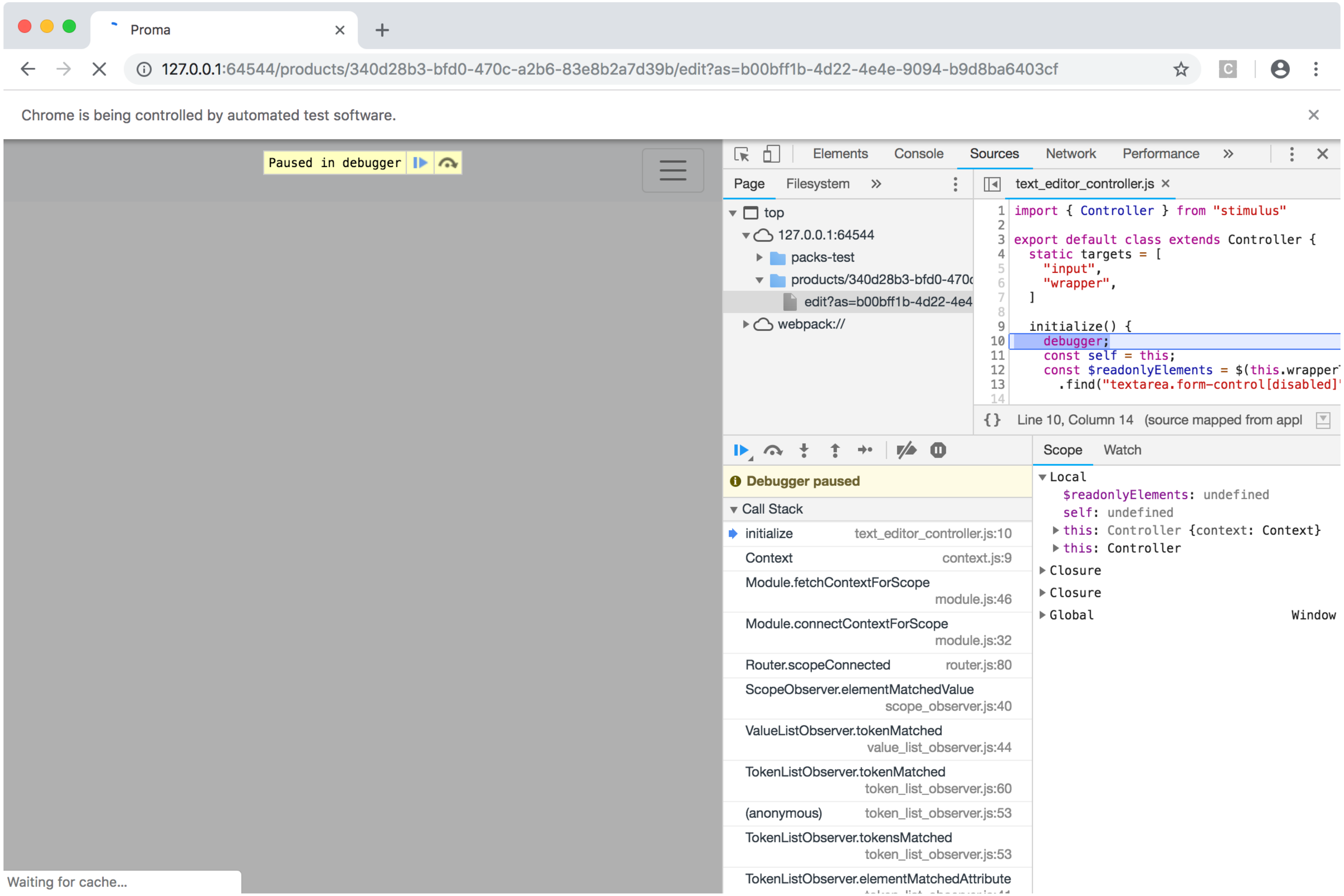
💥 `Uncaught TypeError: this.x is not a function`

```javascript
import { Controller } from "stimulus"

export default class extends Controller {
  initialize() {
    debugger;

    // Start the bug hunt...
  }
}
```

```
bundle exec rspec --only-failures

# 1 example, 1 failure
```

```ruby
# spec/support/capybara.rb

if ENV.key?("CHROME_OPEN_DEVTOOLS")
  chrome_options << "auto-open-devtools-for-tabs"
end
```

```
# .env.test.local

CHROME_OPEN_DEVTOOLS=1
```


Proma

127.0.0.1:64544/products/340d28b3-bfd0-470c-a2b6-83e8b2a7d39b/edit?as=b00bff1b-4d22-4e4e-9094-b9d8ba6403cf

Chrome is being controlled by automated test software.

Paused in debugger

text_editor_controller.js

```
import { Controller } from "stimulus"

export default class extends Controller {
  static targets = [
    "input",
    "wrapper",
  ]

  initialize() {
    debugger;
    const self = this;
    const $readonlyElements = $(this.wrapper
      .find("textarea.form-control[disabled]"
```

Line 10, Column 14 (source mapped from appl

Debugger paused

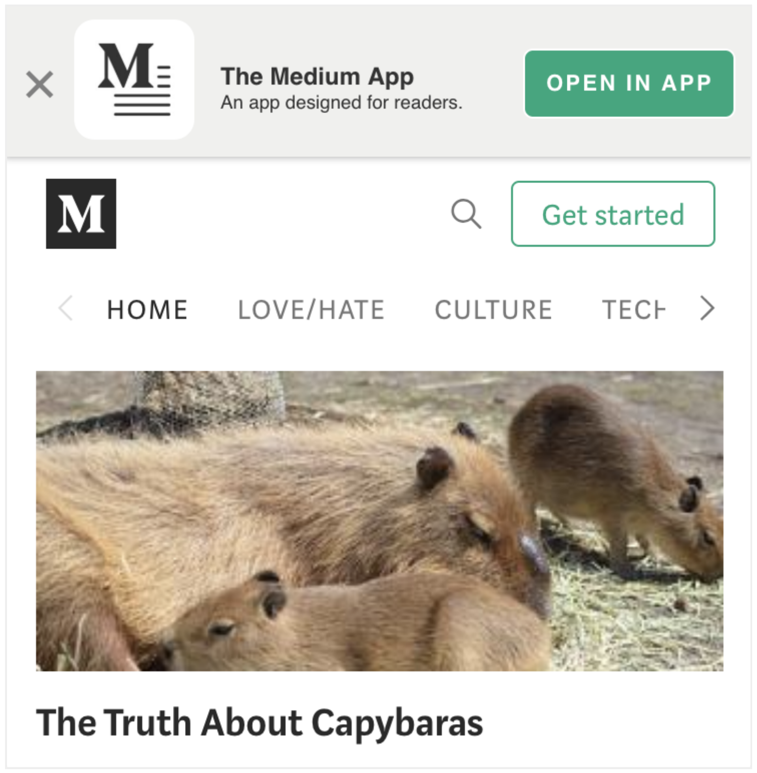
Call Stack

```
initialize                          text_editor_controller.js:10
Context                             context.js:9
Module.fetchContextForScope         module.js:46
Module.connectContextForScope       module.js:32
Router.scopeConnected               router.js:80
ScopeObserver.elementMatchedValue   scope_observer.js:40
ValueListObserver.tokenMatched      value_list_observer.js:44
TokenListObserver.tokenMatched      token_list_observer.js:60
(anonymous)                         token_list_observer.js:53
TokenListObserver.tokensMatched     token_list_observer.js:53
TokenListObserver.elementMatchedAttribute
```

Scope

```
▼ Local
    $readonlyElements: undefined
    self: undefined
  ▶ this: Controller {context: Context}
  ▶ this: Controller
▶ Closure
▶ Closure
▶ Global                                 Window
```

Waiting for cache...



The Truth About Capybaras

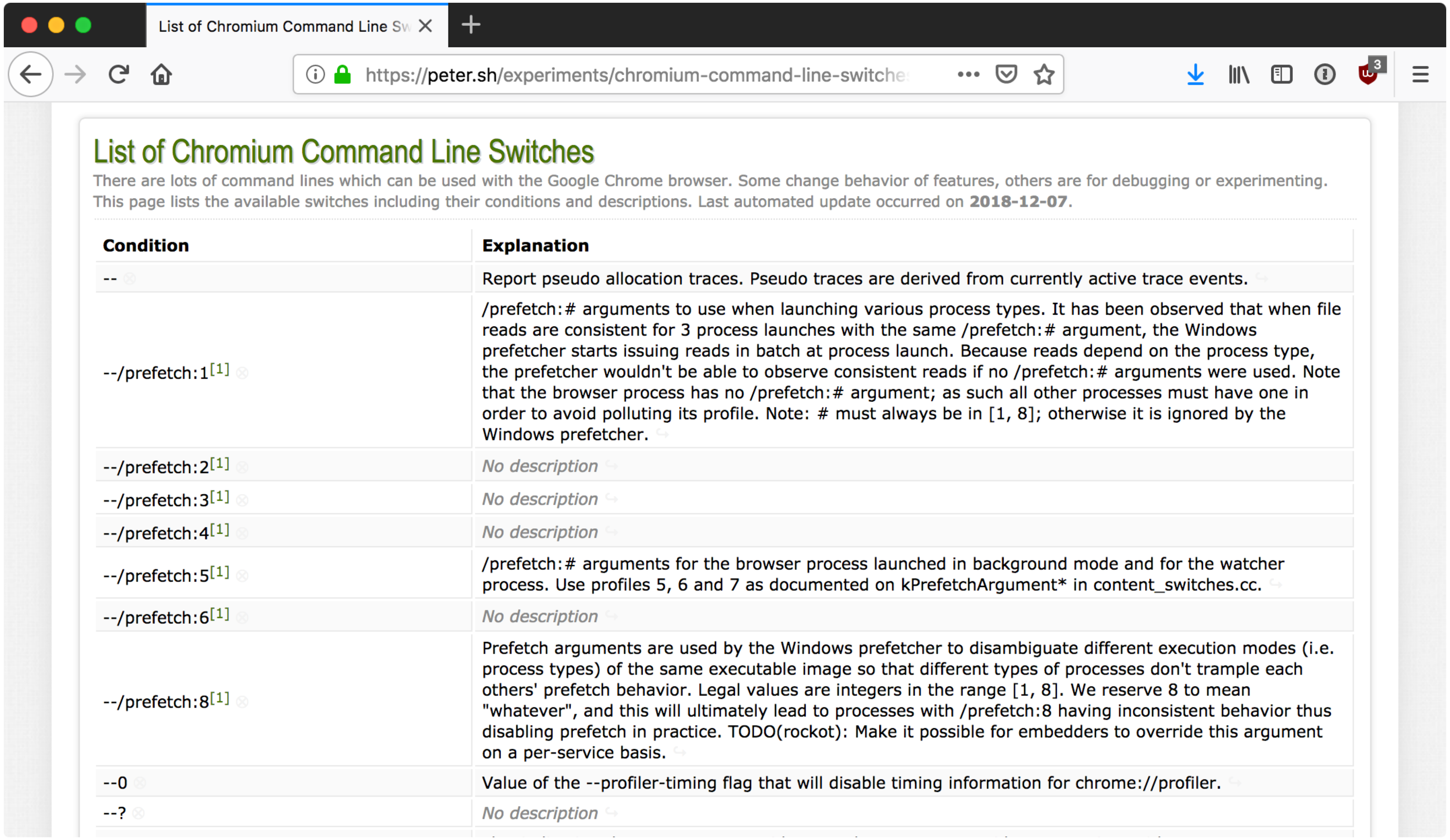
```ruby
context "on an iOS device" do
  it "displays an App Store link" do
    visit root_path

    download_href = find_link("Download App")[:href]

    expect(download_href).to start_with "https://itunes.apple.com"
  end
end
```

```ruby
context "on an iOS device" do
  it "displays an App Store link" do
    visit root_path

    download_href = find_link("Download App")[:href]

    expect(download_href).to start_with "https://itunes.apple.com"
  end
end

context "on an Android device" do
  it "displays a Play Store link" do
    visit root_path

    download_href = find_link("Download App")[:href]

    expect(download_href).to start_with "https://play.google.com"
  end
end
```

```ruby
# ... other setup

drivers_options = {
  chrome: {},
  chrome_android_phone: {
    "mobileEmulation" => { "deviceName" => "Nexus 5X" },
  },
  chrome_ios_phone: {
    "mobileEmulation" => { "deviceName" => "iPhone 6" },
  },
}

drivers_options.each do |name, driver_options|
  Capybara.register_driver(name) do |app|
    capabilities = Selenium::WebDriver::Remote::Capabilities.chrome(
      chromeOptions: { args: chrome_options }.merge(driver_options)
    )

    Capybara::Selenium::Driver.new app,
      browser: :chrome,
      desired_capabilities: capabilities
  end
end
```

```diff
-context "on an iOS device" do
+context "on an iOS device", driver: :chrome_ios_phone do
  it "displays an App Store link" do
    visit root_path

    download_href = find_link("Download App")[:href]

    expect(download_href).to start_with "https://itunes.apple.com"
  end
end

-context "on an Android device" do
+context "on an Android device", driver: :chrome_android_phone do
  it "displays a Play Store link" do
    visit root_path

    download_href = find_link("Download App")[:href]

    expect(download_href).to start_with "https://play.google.com"
  end
end
```

https://peter.sh/experiments/chromium-command-line-switches/


List of Chromium Command Line Sw
https://peter.sh/experiments/chromium-command-line-switche

List of Chromium Command Line Switches

There are lots of command lines which can be used with the Google Chrome browser. Some change behavior of features, others are for debugging or experimenting. This page lists the available switches including their conditions and descriptions. Last automated update occurred on **2018-12-07**.

Condition	Explanation
--	Report pseudo allocation traces. Pseudo traces are derived from currently active trace events.
--/prefetch:1[1]	/prefetch:# arguments to use when launching various process types. It has been observed that when file reads are consistent for 3 process launches with the same /prefetch:# argument, the Windows prefetcher starts issuing reads in batch at process launch. Because reads depend on the process type, the prefetcher wouldn't be able to observe consistent reads if no /prefetch:# arguments were used. Note that the browser process has no /prefetch:# argument; as such all other processes must have one in order to avoid polluting its profile. Note: # must always be in [1, 8]; otherwise it is ignored by the Windows prefetcher.
--/prefetch:2[1]	*No description*
--/prefetch:3[1]	*No description*
--/prefetch:4[1]	*No description*
--/prefetch:5[1]	/prefetch:# arguments for the browser process launched in background mode and for the watcher process. Use profiles 5, 6 and 7 as documented on kPrefetchArgument* in content_switches.cc.
--/prefetch:6[1]	*No description*
--/prefetch:8[1]	Prefetch arguments are used by the Windows prefetcher to disambiguate different execution modes (i.e. process types) of the same executable image so that different types of processes don't trample each others' prefetch behavior. Legal values are integers in the range [1, 8]. We reserve 8 to mean "whatever", and this will ultimately lead to processes with /prefetch:8 having inconsistent behavior thus disabling prefetch in practice. TODO(rockot): Make it possible for embedders to override this argument on a per-service basis.
--0	Value of the --profiler-timing flag that will disable timing information for chrome://profiler.
--?	*No description*

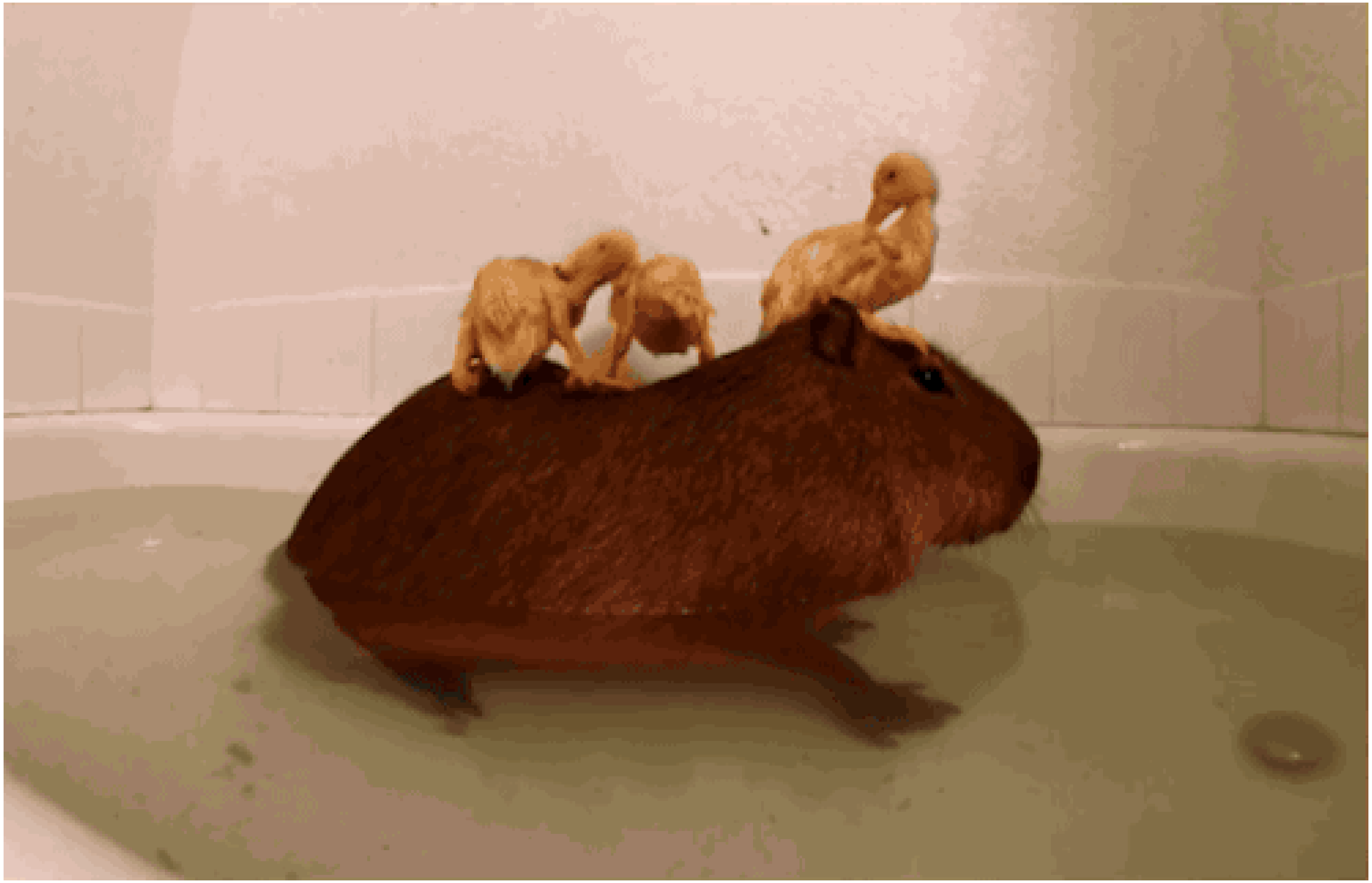
🔊

```
it "has a welcome video" do
  expect(time_played).to eq "0:00"

  find("video").click

  expect(time_played).to_not eq "0:00"
end
```

🔇

```ruby
it "has a welcome video" do
  expect(time_played).to eq "0:00"

  find("video").click

  expect(time_played).to_not eq "0:00"
end
```

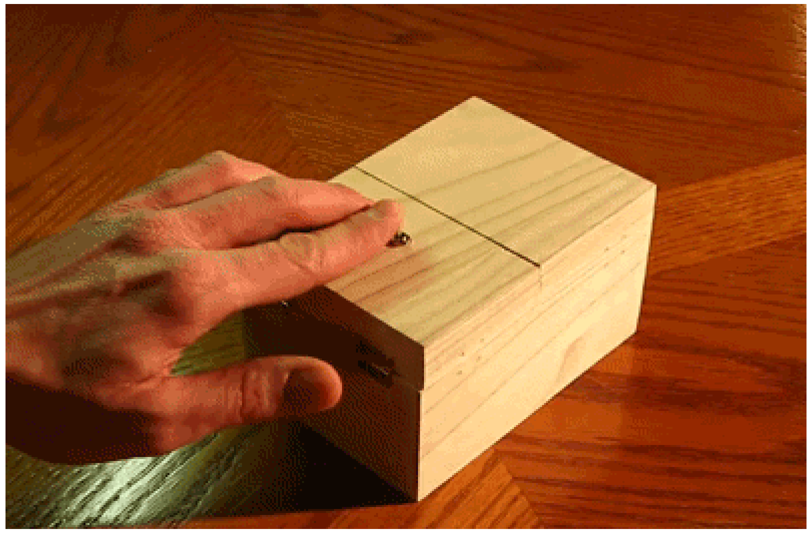
```ruby
# spec/support/capybara.rb

chrome_options << "mute-audio"
```



Flaky specs erode trust

```
bundle exec rspec

# 1663 examples, 1 failure

bundle exec rspec --only-failures

# 1 example, 0 failures
```

Burn-in new feature specs



```ruby
it "allows new users to register" do
  fill_in_date("Date of Birth", with: 25.years.ago)

  click_on "Continue"

  # ...
end
```

```
repeat 10 { bundle exec rspec spec/a_spec.rb:4 } | grep failure
repeat 10 { bundle exec rspec spec/a_spec.rb:4 } | grep failure

# 1 example, 0 failures
# 1 example, 1 failure
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 1 failure
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
```

80% pass rate...

```
 module CapybaraFormHelper
   def fill_in_date(label, date:)
     fill_in(label, with: date.strftime("%d/%m/%Y"))
+    blur_focus
   end

+  def blur_focus
+    find("body").click
+  end
 end
```

```
repeat 10 { bundle exec rspec spec/a_spec.rb:4 } | grep failure

# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
# 1 example, 0 failures
```

100% pass rate 🎉

Thanks!

@olipeate