Advanced Web Scraping: Selenium and Requests-HTML

Clarke Bishop
BIG DATA ENGINEER

@ClarkeBishop www.clarkebishop.com

Overview

Scraping JavaScript generated websites

Requests-HTML

Selenium

JavaScript Scraping Problem

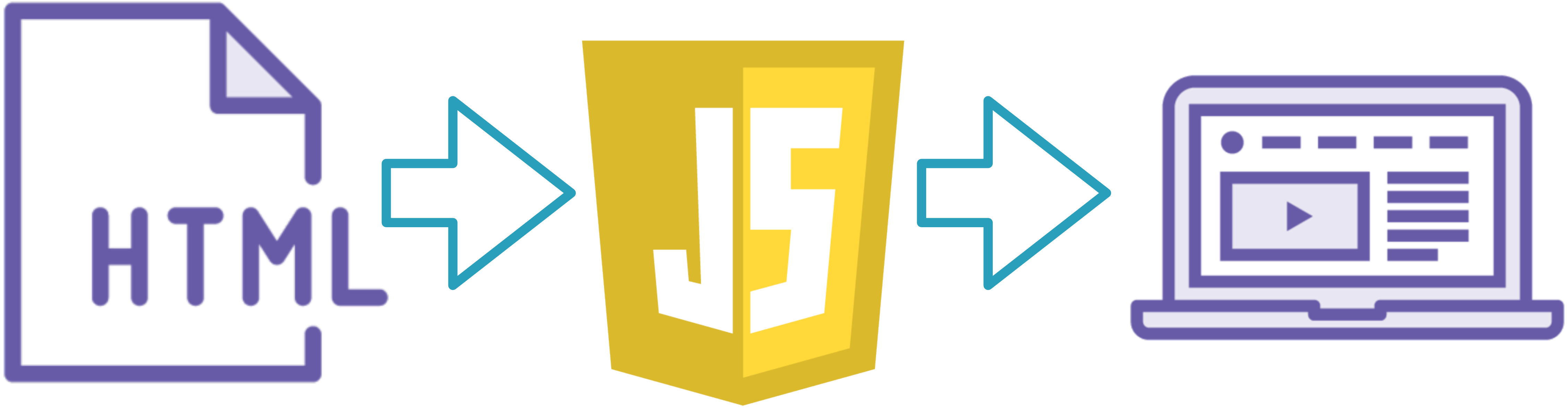
Demo

iSeeCars uses JavaScript

Chrome DevTools - slow rendering

Chrome DevTools - disable JavaScript

JavaScript Page Rendering



JavaScript Page Not Rendering

Requests-HTML

Search projects

Help Donate Log in Register

requests-html 0.10.0

```
pip install requests-html
```

Latest version

Released: Feb 17, 2019

HTML Parsing for Humans.

Navigation

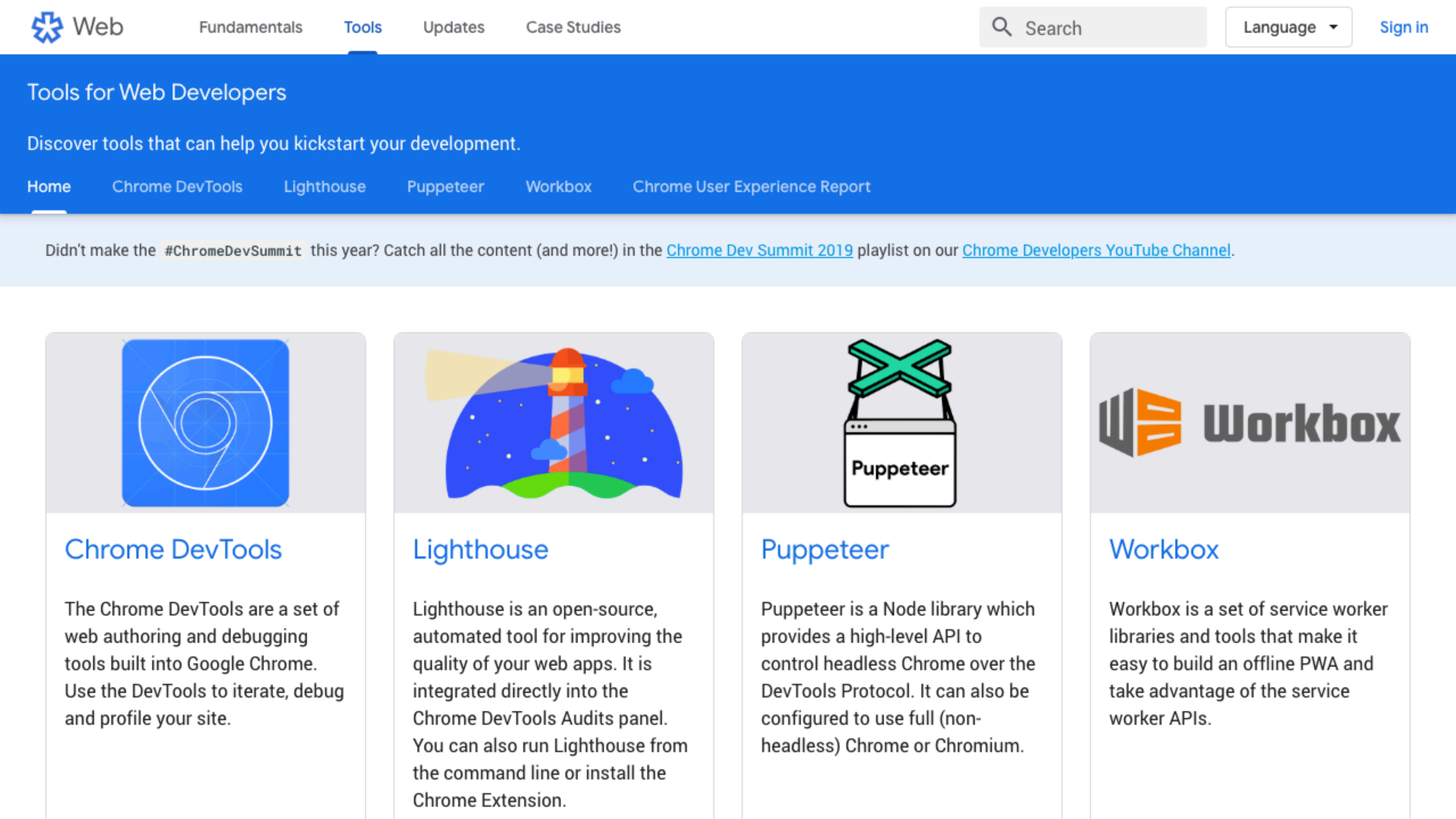
- Project description
- Release history
- Download files

Project links

- Homepage

Project description

Requests-HTML: HTML Parsing for Humans™



Web Fundamentals Tools Updates Case Studies Search Language Sign in

Tools for Web Developers

Discover tools that can help you kickstart your development.

Home Chrome DevTools Lighthouse Puppeteer Workbox Chrome User Experience Report

Didn't make the `#ChromeDevSummit` this year? Catch all the content (and more!) in the Chrome Dev Summit 2019 playlist on our Chrome Developers YouTube Channel.

Chrome DevTools

The Chrome DevTools are a set of web authoring and debugging tools built into Google Chrome. Use the DevTools to iterate, debug and profile your site.

Lighthouse

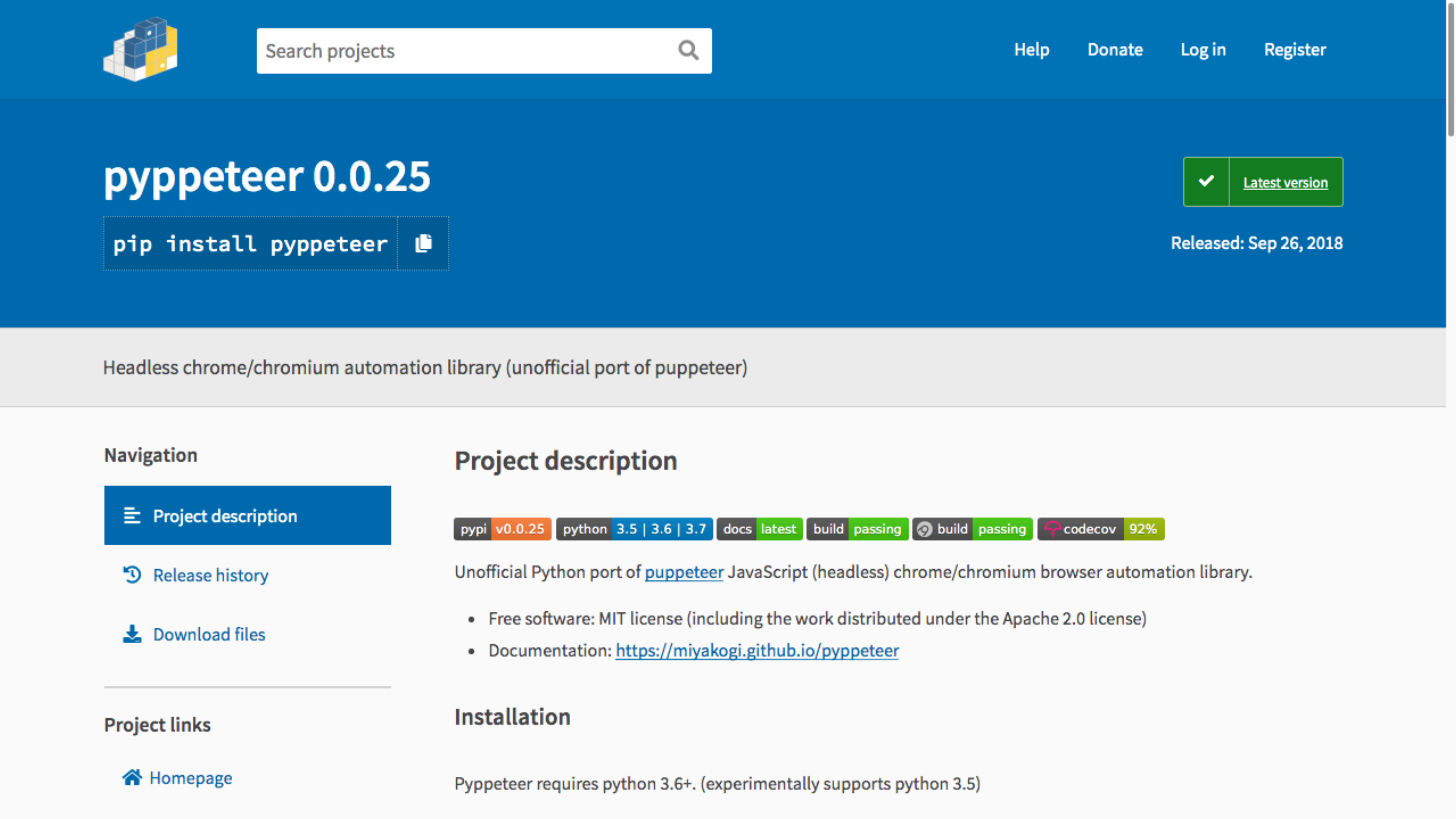
Lighthouse is an open-source, automated tool for improving the quality of your web apps. It is integrated directly into the Chrome DevTools Audits panel. You can also run Lighthouse from the command line or install the Chrome Extension.

Puppeteer

Puppeteer is a Node library which provides a high-level API to control headless Chrome over the DevTools Protocol. It can also be configured to use full (non-headless) Chrome or Chromium.

Workbox

Workbox is a set of service worker libraries and tools that make it easy to build an offline PWA and take advantage of the service worker APIs.


Search projects

Help Donate Log in Register

pyppeteer 0.0.25

```
pip install pyppeteer
```

Latest version

Released: Sep 26, 2018

Headless chrome/chromium automation library (unofficial port of puppeteer)

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Project description

pypi v0.0.25 | python 3.5 | 3.6 | 3.7 | docs latest | build passing | build passing | codecov 92%

Unofficial Python port of puppeteer JavaScript (headless) chrome/chromium browser automation library.

- Free software: MIT license (including the work distributed under the Apache 2.0 license)
- Documentation: https://miyakogi.github.io/pyppeteer

Installation

Pyppeteer requires python 3.6+. (experimentally supports python 3.5)

Headless Browsers

Pyppeteer (Puppeteer) - Requests-HTML

Selenium

Automated Testing

Development Environment

```
> cd my-dev
> mkdir r-html-project
> cd r-html-project
```

◀ Change to your dev directory. Make a new directory, and cd into the new directory.

```
> pyenv install 3.6.9
> pyenv local 3.6.9
```

◀ Set the local Python with pyenv.

```
> pipenv --python 3.6.9
> pipenv install requests-html
> pipenv shell
```

◀ Install requests-html and make sure to launch the pipenv shell.

```python
from requests_html import HTMLSession


session = HTMLSession()

start_url = 'https://www.iseecars.com/used-cars/used-tesla-for-sale#Location=66952' + \
            '&Radius=all&Make=Tesla&Model=Model+3&Condition=used&_t=a&maxResults=15' + \
            '&sort=BestDeal&sortOrder=desc&lfc_t0=MTU2Nzk2NzkzNDc2NQ%3D%3D'

r = session.get(start_url)
```

Retrieve a page with requests-html

```
tesla = r.html.find('div#cars_v2-result-list article', first=True)

model = tesla.find('h3', first=True).text

print(model)    // Returns: 2017 Tesla Model S 60D - 17,181 mi
```

Extract Data for a Tesla

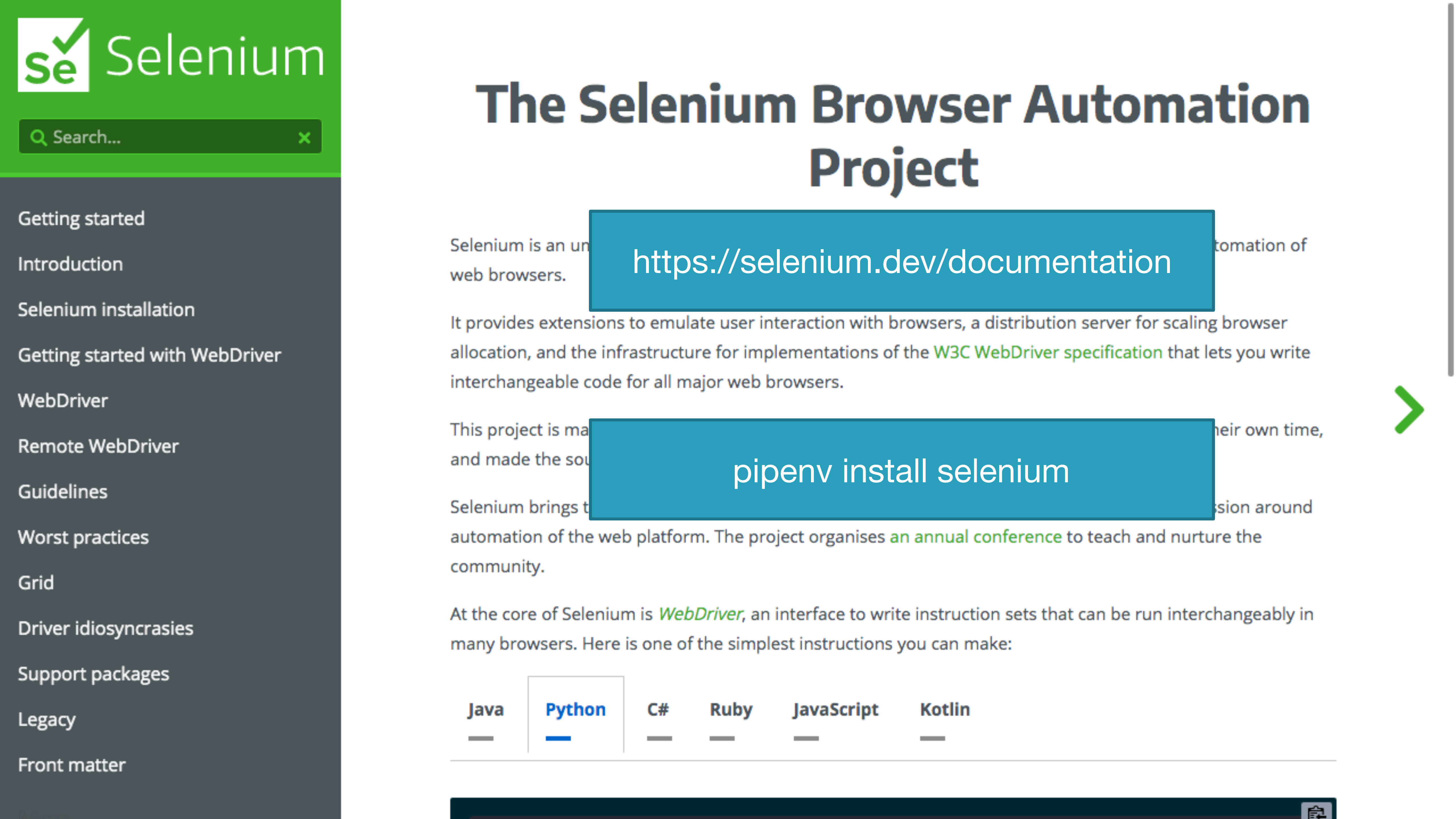
```
r.html.render(sleep=5)

tesla = r.html.find('div#cars_v2-result-list article', first=True)

model = tesla.find('h3', first=True).text

print(model)    // Returns: 2018 Tesla Model 3 Mid range battery - 5,818 mi
```

Extract the Right Tesla Data

Selenium

Requests-HTML or Selenium

Requests-HTML	Selenium
Pyppeteer (Puppeteer) - Based on Chrome	Firefox, Safari, Opera, Chrome, or Edge
XPath or CSS Selectors	XPath or CSS Selectors
Easier to get started	Very Powerful



The Selenium Browser Automation Project

Selenium is an un tomation of web browsers.

https://selenium.dev/documentation

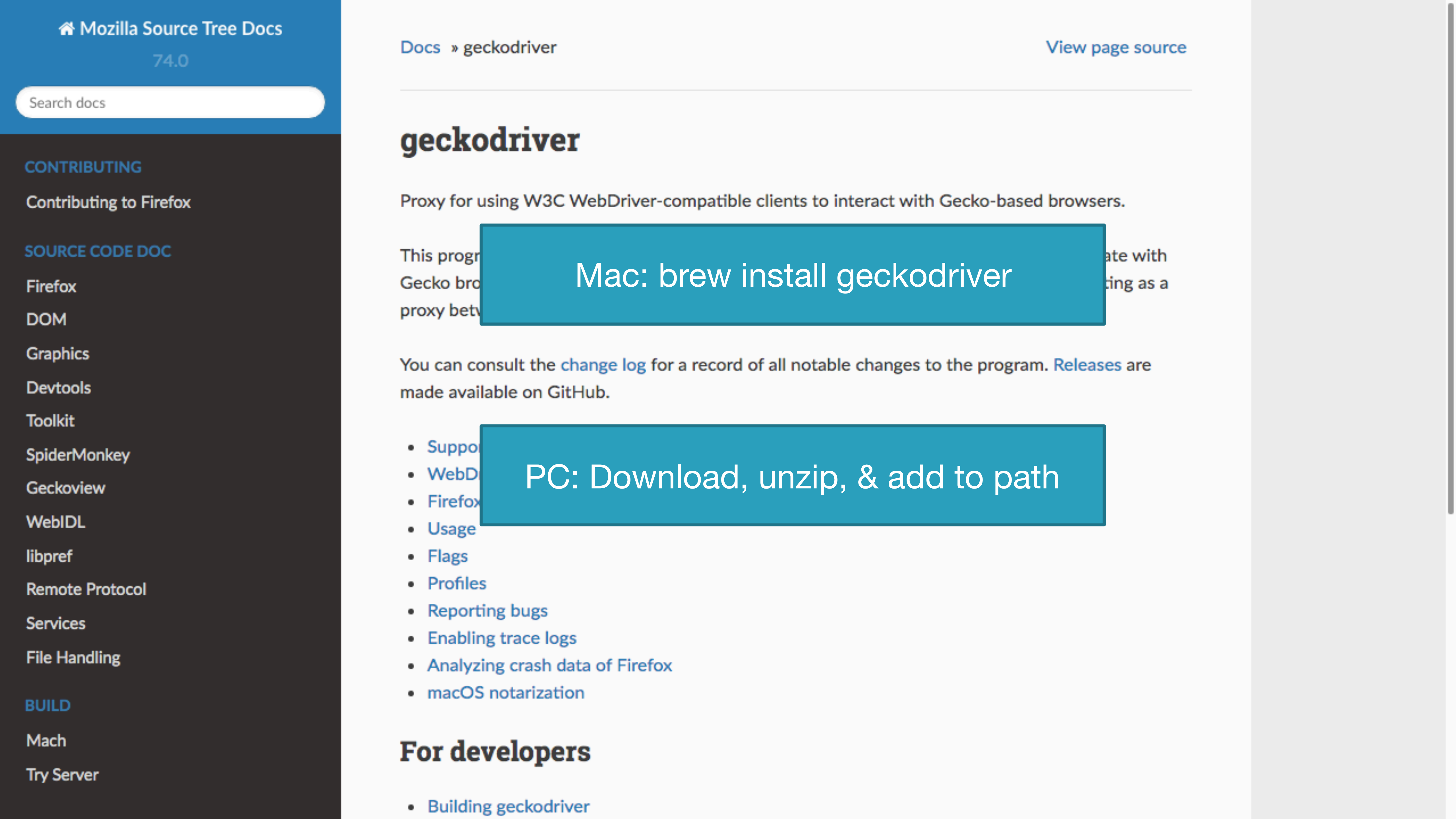
It provides extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the W3C WebDriver specification that lets you write interchangeable code for all major web browsers.

This project is ma heir own time, and made the sou

pipenv install selenium

Selenium brings t ssion around automation of the web platform. The project organises an annual conference to teach and nurture the community.

At the core of Selenium is *WebDriver*, an interface to write instruction sets that can be run interchangeably in many browsers. Here is one of the simplest instructions you can make:



Mozilla Source Tree Docs

74.0

Search docs

CONTRIBUTING

Contributing to Firefox

SOURCE CODE DOC

Firefox

DOM

Graphics

Devtools

Toolkit

SpiderMonkey

Geckoview

WebIDL

libpref

Remote Protocol

Services

File Handling

BUILD

Mach

Try Server

Docs » geckodriver

View page source

geckodriver

Proxy for using W3C WebDriver-compatible clients to interact with Gecko-based browsers.

This progr ate with Gecko bro ting as a proxy betv

Mac: brew install geckodriver

You can consult the change log for a record of all notable changes to the program. Releases are made available on GitHub.

- Suppo
- WebD
- Firefox
- Usage
- Flags
- Profiles
- Reporting bugs
- Enabling trace logs
- Analyzing crash data of Firefox
- macOS notarization

PC: Download, unzip, & add to path

For developers

- Building geckodriver

```python
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
import time


start_url = 'https://www.iseecars.com/used-cars/used-tesla-for-sale#Location=66952' + \
            '&Radius=all&Make=Tesla&Model=Model+3&Condition=used&_t=a&maxResults=15' + \
            '&sort=BestDeal&sortOrder=desc&lfc_t0=MTU2Nzk2NzkzNDc2NQ%3D%3D'

with webdriver.Firefox() as driver:
    . . .
```

Selenium Imports & Setup

```
with webdriver.Firefox() as driver:
    wait = WebDriverWait(driver, 10)
    driver.get(start_url)

    time.sleep(10)

    teslas = driver.find_element_by_css_selector('div#cars_v2-result-list article')
    model = teslas.find_element_by_css_selector('h3')

    print(model.text)   // Returns: 2018 Tesla Model 3 Mid range battery - 5,818 mi
```

Selenium Scraping

```
with webdriver.Firefox() as driver:
    wait = WebDriverWait(driver, 10)
    driver.get(start_url)

    wait.until ( . . . )

    teslas = driver.find_element_by_css_selector('div#cars_v2-result-list article')
    model = teslas.find_element_by_css_selector('h3')

    print(model.text)   // Returns: 2018 Tesla Model 3 Mid range battery - 5,818 mi
```

Waiting for an element

Summary

Scraping JavaScript generated websites

Requests-HTML

- Code example
- Sleep to give JavaScript time

Selenium

- Code example
- Install a WebDriver
- wait.until (. . .)