Quick Start

Install

```
pip install ichrome -U
```

> Uninstall & Clear the user data folder

```
    $ python3 -m ichrome --clean
    $ pip uninstall ichrome
```

47M

Download & unzip the latest version of Chromium browser

> python3 -m ichrome –install="/home/root/chrome_bin"

WARNING:

install the missing packages yourself

a. use `ldd chrome | grep not` on linux to check and install them, or view the link: Checking out and building Chromium on Linux

add `--no-sandbox` to extra_configs to avoid `No usable sandbox` errors, unless you really need `sandbox`: Chromium: "Running without the SUID sandbox!" error - Ask Ubuntu

Have a Try?

> Interactive Debugging (REPL Mode)

There are two ways to enter the `REPL` mode

`python3 -m ichrome -t`

or run `await tab.repl()` in your code

```
λ python3 -m ichrome -t
>>> await tab.goto('https://github.com/ClericPy')
True
>>> title = await tab.title
>>> title
'ClericPy (ClericPy) · GitHub'
>>> await tab.click('.pinned-item-list-item-content
```

```
[href="/ClericPy/ichrome"]')
Tag(a)
>>> await tab.wait_loading(2)
True
>>> await tab.wait_loading(2)
False
>>> await tab.js('document.body.innerHTML="Updated"')
{'type': 'string', 'value': 'Updated'}
>>> await tab.history_back()
True
>>> await tab.set_html('hello world')
{'id': 21, 'result': {}}
>>> await tab.set_ua('no UA')
{'id': 22, 'result': {}}
>>> await tab.goto('http://httpbin.org/user-agent')
True
>>> await tab.html
'<html><head></head><body><pre style="word-wrap: break-word; white-space: pre-
wrap;">{\n  "user-agent": "no UA"\n}\n</pre></body></html>'
```

Quick Start

Start a new chrome daemon process with headless=False

`python -m ichrome`

Then connect to an exist chrome instance

```
        async with AsyncChrome() as chrome:
            async with chrome.connect_tab() as tab:
```

or launching the chrome daemon in code may be a better choice

```
        async with AsyncChromeDaemon() as cd:
            async with cd.connect_tab() as tab:
```

Operations on the tabs: new tab, wait loading, run javascript, get html, close tab

Close the browser GRACEFULLY instead of killing process

```
from ichrome import AsyncChromeDaemon
import asyncio


async def main():
    async with AsyncChromeDaemon(headless=0, disable_image=False) as cd:
        # [index] 0: current activate tab, 1: tab 1, None: new tab, $URL: new
tab for url
        async with cd.connect_tab(index=0, auto_close=True) as tab:
            # tab: AsyncTab
            await tab.alert(
                'Now using the default tab and goto the url, click to
continue.'
```

```
            )
            print(await tab.goto('https://github.com/ClericPy/ichrome',
                                 timeout=5))
            # wait tag appeared
            await tab.wait_tag('[data-content="Issues"]', max_wait_time=5)
            await tab.alert(
                'Here the Issues tag appeared, I will click that button.')
            # click the issues tag
            await tab.click('[data-content="Issues"]')
            await tab.wait_tag('#js-issues-search')
            await tab.alert('Now will input some text and search the issues.')
            await tab.mouse_click_element_rect('#js-issues-search')
            await tab.keyboard_send(string='chromium')
            await tab.js(
                r'''document.querySelector('[role="search"]').submit()''')
            await tab.wait_loading(5)
            await asyncio.sleep(2)
            await tab.alert('demo finished.')

            # start REPL mode
            # await tab.repl()

            # no need to close tab for auto_close=True
            # await tab.close()

        # # close browser gracefully
        # await cd.close_browser()
        print('clearing the user data cache.')
        await cd.clear_user_data_dir()


if __name__ == "__main__":
    asyncio.run(main())
```

Listen to the network traffic, or abort some requests.

```
import asyncio
import json

from ichrome import AsyncChromeDaemon


async def main():
    async with AsyncChromeDaemon() as cd:
        async with cd.connect_tab() as tab:
            url = 'http://httpbin.org/ip'
            # 1. listen request/response network
            RequestPatternList = [{
                'urlPattern': '*httpbin.org/ip*',
                'requestStage': 'Response'
            }]
            async with tab.iter_fetch(RequestPatternList) as f:
                await tab.goto(url, timeout=0)
                # only one request could be catched
                event = await f
```

```
                print('request event:', json.dumps(event), flush=True)
                response = await f.get_response(event, timeout=5)
                print('response body:', response['data'])

            # 2. disable image requests
            url = 'https://www.bing.com'
            RequestPatternList = [
                {
                    'urlPattern': '*',
                    'resourceType': 'Image',  # could be other types
                    'requestStage': 'Request'
                },
                {
                    'urlPattern': '*',
                    'resourceType': 'Stylesheet',
                    'requestStage': 'Request'
                },
                {
                    'urlPattern': '*',
                    'resourceType': 'Script',
                    'requestStage': 'Request'
                },
            ]
            # listen 5 seconds
            async with tab.iter_fetch(RequestPatternList, timeout=5) as f:
                await tab.goto(url, timeout=0)
                # handle all the matched requests
                async for event in f:
                    if f.match_event(event, RequestPatternList[0]):
                        print('abort request image:',
                              tab.get_data_value(event, 'params.request.url'),
                              flush=True)
                        await f.failRequest(event, 'Aborted')
                    elif f.match_event(event, RequestPatternList[1]):
                        print('abort request css:',
                              tab.get_data_value(event, 'params.request.url'),
                              flush=True)
                        await f.failRequest(event, 'ConnectionRefused')
                    elif f.match_event(event, RequestPatternList[2]):
                        print('abort request js:',
                              tab.get_data_value(event, 'params.request.url'),
                              flush=True)
                        await f.failRequest(event, 'AccessDenied')
                await asyncio.sleep(5)


if __name__ == "__main__":
    asyncio.run(main())
```

Incognito Mode

Set a new proxy.

The startup speed is much faster than a new `Chrome Daemon`.

a. But slower than a new `Tab`

i. https://github.com/ClericPy/ichrome/issues/87

ii. 40% performance lost

Proxy Authentication: https://github.com/ClericPy/ichrome/issues/86

```
import asyncio

from ichrome import AsyncChromeDaemon


async def main():
    async with AsyncChromeDaemon() as cd:
        proxy = None
        async with cd.incognito_tab(proxyServer=proxy) as tab:
            # This tab will be created in the given BrowserContext
            await tab.goto('http://httpbin.org/ip', timeout=10)
            # print and watch your IP changed
            print(await tab.html)


asyncio.run(main())
```

Example Code: examples_async.py & Classic Use Cases