Simple abstraction to use Chrome as a Headless Browser with Node JS

#node-browsers #unit-testing #testing-tools #chrome-headless #google-chrome #chrome #horseman

254 commits | 2 branches | 66 releases | 17 contributors | MIT

Branch: master | New pull request | Find file | Clone or download

LucianoGanga 4.3.7 — Latest commit 86a49ca 24 minutes ago

.vscode	First commit	3 months ago
build	Change package.json	25 minutes ago
examples	Add the screenshots examples	21 days ago
lib	Fix	30 minutes ago
.babelrc	Change directories names for build and src	a month ago
.gitignore	Remove 'build' directory from .gitignore to fix #55	an hour ago
.npmignore	Change directories names for build and src	a month ago
.travis.yml	chore(travis): initial version	2 months ago
LICENSE	Initial commit	3 months ago
README.md	Rebuild + docs fix	39 minutes ago
index.js	Point index.js to /build	28 days ago
package.json	4.3.7	24 minutes ago
release-notes.md	Add release notes for v4.0.0	28 days ago
yarn.lock	add #injectRemoteScript and #injectScript	17 days ago

README.md

build passing

simple-headless-chrome

Important version >= 3.3.0

Version 3.3.0 includes a new feature that allows managing browser tabs.

This new feature comes with some breaking changes that will allow us future scalability.

To avoid problems for people that uses version >= 3.3.0 of this module, we supported those breaking changes with methods that will be deprecated in version 4.0.0.

Introduction

This is an abstraction to use a Headless version of Google Chrome in a very simple way. I was inspired by the next projects:

- Doffy (https://github.com/qieguo2016/doffy)
- Horseman (https://github.com/johntitus/node-horseman)
- chrome-remote-interface (https://github.com/cyrus-and/chrome-remote-interface)
- lighthouse (https://github.com/googlechrome/lighthouse)

And I had to read a lot here too:

- https://developers.google.com/web/updates/2017/04/headless-chrome
- https://chromedevtools.github.io/devtools-protocol

And you can also use this in heroku thanks to https://github.com/heroku/heroku-buildpack-google-chrome

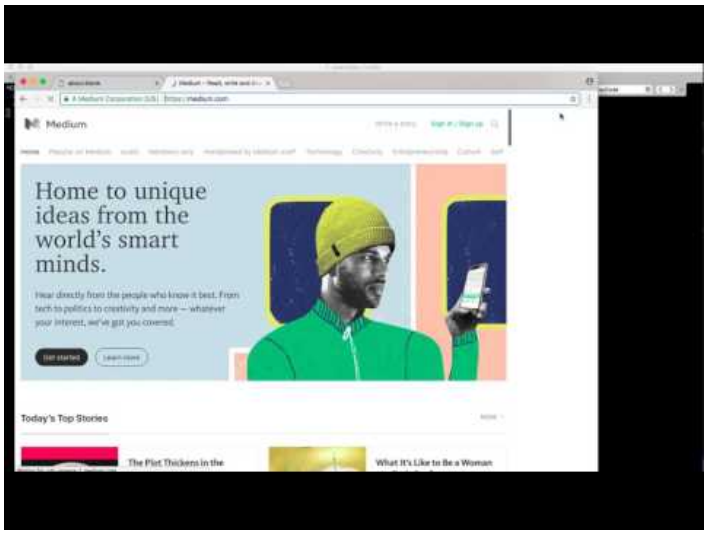
I built this basically because I got tired of an error I received in an edge case when using PhantomJS (Unhandled reject Error: Failed to load url). So I decided to make my own abstraction, to be used in a heroku app, and simple to use as Horseman.

I didn't have time to document here in the readme, but every method in the source code is documented.

It's really simple to use. I hope I can get some time to make a QuickStart guide + document the API methods here.

You can read my post in Medium about this module: How to tell to a headless Google Chrome to write a post in Medium for you

You can check a video of the module in action clicking in the image below



Collaboration

If you want to collaborate with the project, in any way (documentation, examples, fixes, etc), just send a PR :)

If you rock at making tests, it would be very useful if you can help us making this module better. It's not necesary to build all the tests, but if someone knows how to code the base to add tests to this module, it would really help for someone else to start with this part.

Thank you to everyone who already help submitting a PR! :D

Installation

1) Install Google Chrome Headless

In your PC

Mac: Chrome Headless is shipped in Chrome Canary. You can install it here:
https://www.google.com/chrome/browser/canary.html

Linux: Chrome headless is shipped on chrome 59. so you can install Chrome 59 to use the headless mode:

https://askubuntu.com/questions/79280/how-to-install-chrome-browser-properly-via-command-line

```
sudo apt-get install libxss1 libappindicator1 libindicator7
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo dpkg -i google-chrome*.deb  # Might show "errors", fixed by next line
sudo apt-get install -f
```

In a NodeJS Heroku App

Just add the buildpack for Heroku and vualá! Everything is ready You can check the buildpack repository here:
https://github.com/heroku/heroku-buildpack-google-chrome

Using a Docker image

With the addition of Chrome Remote Interface into Chrome 59, a simple way to install is using the Docker image for Chrome Headless, such as https://hub.docker.com/r/justinribeiro/chrome-headless/ or https://hub.docker.com/r/yukinying/chrome-headless/

If using Docker, in your app, configure for headless as follows:

```
const browser = new HeadlessChrome({
  headless: true,
  launchChrome: false,
  chrome: {
    host: 'localhost',
    port: 9222, // Chrome Docker default port
    remote: true,
  },
  browserlog: true
})
```

2) Install the NPM Module

```
npm install --save simple-headless-chrome
```

Compatibility

Thanks to @lewisf, simple-headless-chrome is compatible on NodeJS >= 4! I hope more persons can benefit of this now :)

Usage

```
const HeadlessChrome = require('simple-headless-chrome')

const browser = new HeadlessChrome({
  headless: true, // If you turn this off, you can actually see the browser navigate with your instructions,
  chrome: {
    userDataDir: '/tmp/headlessDataDir' // This can be null, so a tmp folder will be created and then destroyed
  }
})
```

Once you have the browser instance, you can call the methods to interact with it.

Methods

inject

Injects JavaScript in the page

Modules available: jQuery, jquery, jQuery.slim and jquery.slim

Parameters

- `moduleOrScript` **string** Javascript code, file, url or name of the module to inject.

Examples

```
inject('jquery')

You can use jsdelivr to inject any npm or github package in the page


inject('https://cdn.jsdelivr.net/npm/lodash@4/lodash.min.js')


inject('https://cdn.jsdelivr.net/npm/jquery@3/dist/jquery.min.js')

You can inject a local Javascript file


inject('./custom-file.js')


inject(__dirname + '/path/to/file.js')

Note: the path will be resolved with `require.resolve()` so you can include
files that are in `node_modules` simply by installing them with NPM


inject('jquery/dist/jquery.min')


inject('lodash/dist/lodash.min')
```

injectRemoteScript

Injects a remote script in the page

Parameters

- `src` **string** Url to remote JavaScript file

Examples

```
injectRemoteScript(https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js)
```

injectScript

Injects code in the DOM as script tag

Parameters

- `script` **string** Code to be injected and evaluated in the DOM

evaluate

Evaluates a fn in the context of the browser

Parameters

- `fn` {function} - The function to evaluate in the browser
- `args` **...any** {*} - The arguments to pass to the function

evaluateAsync

Evaluates an async fn in the context of the browser

Parameters

- `fn` {function} - The function to evaluate in the browser
- `args` **...any** {*} - The arguments to pass to the function

evaluateOnNode

Evaluates a fn in the context of a passed node

Parameters

- `node` **NodeObject** The Node Object used to get the context
- `fn` {function} - The function to evaluate in the browser
- `args` **...any** {*} - The arguments to pass to the function

goTo

Navigates to a URL

Parameters

- `url` **string** The URL to navigate to
- `opt` (optional, default `{}`)
- `options` **object** The options object. options:

Properties

- `timeout` **number** Time in ms that this method has to wait until the "pageLoaded" event is triggered. If the value is 0 or false, it means that it doesn't have to wait after calling the "Page.navigate" method

getNodeValue

Get the value of an Node.

Parameters

- `node` **NodeObject** The Node Object

Returns **object** Object containing type and value of the element

getValue

Get the value of an element.

Parameters

- `selector` **string** The target selector
- `frameId` **string** The FrameID where the selector should be searched

Returns **object** Object containing type and value of the element

setNodeValue

Set the value of an element.

Parameters

- `node` **NodeObject** The Node Object
- `value` **string** The value to set the node to (it may be an array of values when the node is a multiple "HTMLSelectElement")

setValue

Set the value of an element.

Parameters

- `selector` **string** The selector to set the value of.
- `value` **string?** The value to set the selector to
- `frameId` **string** The FrameID where the selector should be searched

fill

Fills a selector of an input or textarea element with the passed value

Parameters

- `selector` **string** The selector
- `value` **string** The value to fill the element matched in the selector
- `frameId` **string** The FrameID where the selector should be searched

clear

Clear an input field.

Parameters

- `selector` **string** The selector to clear.
- `frameId` **string** The FrameID where the selector should be searched

querySelector

Returns the node associated to the passed selector

Parameters

- `selector` **string** The selector to find
- `frameId` **string** The FrameID where the selector should be searched

focus

Focus on an element matching the selector

Parameters

- `selector` **string** The selector to find the element
- `frameId` **string** The FrameID where the selector should be searched

type

Simulate a keypress on a selector

Parameters

- `selector` **string** The selector to type into.
- `text` **string** The text to type.
- `frameId` **string** The FrameID where the selector should be searched
- `opts`
- `options` **object** Lets you send keys like control & shift

typeText

Types text (doesn't matter where it is)

Parameters

- `text` **string** The text to type.
- `opts`
- `options` **object** Lets you send keys like control & shift

select

Select a value in an html select element.

Parameters

- `selector` **string** The identifier for the select element.
- `value` **string** The value to select.
- `frameId` **string** The FrameID where the selector should be searched

keyboardEvent

- **See:** https://chromedevtools.github.io/devtools-protocol/tot/Input/#method-dispatchKeyEvent

Fire a key event.

Parameters

- `type` **string** The type of key event. (optional, default `keypress`)
- `key` **string** The key to use for the event. (optional, default `null`)
- `modifier` **number** The keyboard modifier to use. (optional, default `0`)
- `windowsVirtualKeyCode` (optional, default `0`)

wait

Waits certain amount of ms

Parameters

- `time` **number** Ammount of ms to wait

onConsole

Binding callback to handle console messages

Parameters

- `listener` is a callback for handling console message

waitForPageToLoad

Waits for a page to finish loading. Throws error after timeout

Parameters

- `timeout` **number** The timeout in ms. (Default: "loadPageTimeout" property in the browser instance options)

waitForFrameToLoad

Waits for all the frames in the page to finish loading. Returns the list of frames after that

Parameters

- `url` **(regexp | string)** The URL that must be waited for load
- `timeout`

Returns **object** List of frames, with childFrames

waitForSelectorToLoad

Waits for a selector to finish loading. Throws error after timeout

Parameters

- `selector` **string** The identifier for the select element.
- `interval` **number** The interval in ms. (Default: "loadPageTimeout" property in the browser instance options)
- `timeout` **number** The timeout in ms. (Default: "loadPageTimeout" property in the browser instance options)

mouseEvent

- **See:** https://chromedevtools.github.io/devtools-protocol/tot/Input/#method-dispatchMouseEvent

Fire a mouse event.

Parameters

- `$0` **Object**
  - `$0.type` (optional, default `'mousePressed'`)
  - `$0.x` (optional, default `0`)
  - `$0.y` (optional, default `0`)
  - `$0.modifiers` (optional, default `0`)
  - `$0.button` (optional, default `'left'`)
  - `$0.clickCount` (optional, default `1`)
- `type` **string** Type of the mouse event. Allowed values: mousePressed, mouseReleased, mouseMoved. (optional, default `mousePressed`)
- `x` **number** X coordinate of the event relative to the main frame's viewport. (optional, default `0`)
- `y` **number** Y coordinate of the event relative to the main frame's viewport. 0 refers to the top of the viewport and Y increases as it proceeds towards the bottom of the viewport. (optional, default `0`)
- `modifier` **number** Bit field representing pressed modifier keys. Alt=1, Ctrl=2, Meta/Command=4, Shift=8 (default: 0). (optional, default `0`)
- `button` **string** Mouse button (default: "none"). Allowed values: none, left, middle, right. (optional, default `left`)

click

Click on a selector by firing a 'click event' directly in the element of the selector

Parameters

- `selector` **string** Selector of the element to click
- `frameId` **string** The FrameID where the selector should be searched

clickOnSelector

Clicks left button hover the centroid of the element matching the passed selector

Parameters

- `selector` **string?**
- `frameId` **string** The FrameID where the selector should be searched

getNodeCentroid

Calculates the centroid of a node by using the boxModel data of the element

Parameters

- `nodeId` **string** The Node Id

Returns **object** { x, y } object with the coordinates

getCookies

Get the browser cookies

Returns **object** Object with all the cookies

setCookie

Set the browser cookies

Parameters

- `name` **string** The name of the cookie.
- `value` **string** The value of the cookie.
- `options` (optional, default `{}`)
- `url` **string** The request-URI to associate with the setting of the cookie.

Properties

- `options` **object** Options object
- `domain` **string?** If omitted, the cookie becomes a host-only cookie
- `path` **string?** Defaults to the path portion of the url parameter
- `secure` **boolean?** Defaults to false.
- `httpOnly` **boolean?** Defaults to false.
- `sameSite` **string?** Represents the cookie's 'SameSite' status: https://tools.ietf.org/html/draft-west-first-party-cookies
- `expirationDate` **number?** If omitted, cookie becomes a session cookie }} options - additional options for setting the cookie (more info here: https://chromedevtools.github.io/devtools-protocol/tot/Network/#method-setCookie)

Returns **boolean** True if successfully set cookie

clearBrowserCookies

Clear the browser cookies

exist

Checks if an element matches the selector

Parameters

- `selector` **string** The selector string
- `frameId` **string** The FrameID where the selector should be searched

Returns **boolean** Boolean indicating if element of selector exists or not

visible

Checks if an element matching a selector is visible

Parameters

- `selector` **string** The selector string
- `frameId` **string** The FrameID where the selector should be searched

Returns **boolean** Boolean indicating if element of selector is visible or not

printToPDF

Prints the page to PDF

Parameters

- `options` **object** Options object Options properties: (optional, default `{}`)
- `returnBinary` **boolean** If true, returns as binary. Otherwise, returns a base64 string (optional, default `false`)

Properties

- `landscape` **boolean** Paper orientation. Defaults to false.
- `displayHeaderFooter` **boolean** Display header and footer. Defaults to false.
- `printBackground` **boolean** Print background graphics. Defaults to false.
- `scale` **number** Scale of the webpage rendering. Defaults to 1.
- `paperWidth` **number** Paper width in inches. Defaults to 8.5 inches.
- `paperHeight` **number** Paper height in inches. Defaults to 11 inches.
- `marginTop` **number** Top margin in inches. Defaults to 1cm (~0.4 inches).
- `marginBottom` **number** Bottom margin in inches. Defaults to 1cm (~0.4 inches).
- `marginLeft` **number** Left margin in inches. Defaults to 1cm (~0.4 inches).
- `marginRight` **number** Right margin in inches. Defaults to 1cm (~0.4 inches).
- `pageRanges` **string** Paper ranges to print, e.g., '1-5, 8, 11-13'. Defaults to the empty string, which means print all pages.

Returns **string** Binary or Base64 string with the PDF data

getScreenshot

Takes a screenshot of the page and returns it as a string

Parameters

- `captureOptions` **object** Options object Options properties:
  - `captureOptions.format` (optional, default `'png'`)
  - `captureOptions.quality`
  - `captureOptions.clip` (optional, default `{x:0,y:0,width:this.options.deviceMetrics.width,height:this.options.deviceMetrics.height,scale:this.options.deviceMetrics.deviceScaleFactor}`)
  - `captureOptions.fromSurface`
  - `captureOptions.selector`
  - `captureOptions.fullPage`
- `returnBinary` **boolean** If true, returns as binary. Otherwise, returns a base64 string (optional, default `false`)

Properties

- `format` **string?** Image compression format (defaults to png). Allowed values: jpeg, png.
- `quality` **integer?** Compression quality from range [0..100] (jpeg only).
- `clip` **ViewPort?** Capture the screenshot of a given viewport/region only (https://chromedevtools.github.io/devtools-protocol/tot/Page/#type-Viewport)
- `fromSurface` **boolean?** Capture the screenshot from the surface, rather than the view. Defaults to false. EXPERIMENTAL
- `selector` **string?** The selector to be captured. If empty, will capture the page
- `fullPage` **boolean?** If true, captures the full page height

Returns **string** Binary or Base64 string with the image data

saveScreenshot

Saves a screenshot of the page

Parameters

- `fileName` **boolean** Path and Name of the file (without the extension) (optional, default `` `screenshot-${Date.now()}` ``)
- `captureOptions` **object** Options object Options properties: (optional, default `{}`)
  - `captureOptions.format` (optional, default `'png'`)
  - `captureOptions.quality`
  - `captureOptions.clip`
  - `captureOptions.fromSurface`
  - `captureOptions.selector`
  - `captureOptions.fullPage`

Properties

- `format` **string** Image compression format (defaults to png). Allowed values: jpeg, png.
- `quality` **integer** Compression quality from range [0..100] (jpeg only).
- `clip` **ViewPort** Capture the screenshot of a given region only (https://chromedevtools.github.io/devtools-protocol/tot/Page/#type-Viewport)
- `fromSurface` **boolean** Capture the screenshot from the surface, rather than the view. Defaults to false. EXPERIMENTAL
- `selector` **string?** The selector to be captured. If empty, will capture the page
- `fullPage` **boolean?** If true, captures the full page height

Returns **string** Binary or Base64 string with the image data

getSelectorViewport

Get the Viewport of the element matching a selector

Parameters

- `selector` **string** The selector string
- `frameId` **string** The FrameID where the selector should be searched

Returns **Viewport** Object with the viewport properties (https://chromedevtools.github.io/devtools-protocol/tot/Page/#type-Viewport)

getFrames

Get the list of frames in the loaded page

Returns **object** List of frames, with childFrames

resizeFullScreen

Resize viewports of the page to full screen size

handleDialog

Accepts or dismisses a JavaScript initiated dialog (alert, confirm, prompt, or onbeforeunload)

Parameters

- `accept` **boolean** Whether to accept or dismiss the dialog (optional, default `true`)
- `promptText` **string?** The text to enter into the dialog prompt before accepting. Used only if this is a prompt dialog. (optional, default `''`)

post

Post data from the browser context

Parameters

- `url` **string** The URL or path to POST to
- `data` **object?** The data object to be posted (optional, default `{}`)
- `options` **object?** Options of the request (optional, default `{}`)

Returns **object** Request status and data

value

TODO: Take the value from the DOM Node. For some reason, there're some pages where is not possible to get the textarea value, as its nodeId refreshes all the time

setNodeValue

TODO: Take the value from the DOM Node. For some reason, there're some pages where is not possible to get the textarea value, as its nodeId refreshes all the time

browserIsInitialized

Checks if the browser is initialized. Exits the process if it's not

fixSelector

As the selectors may contain colons, it's necessary to escape them in order to correctly match an element

Parameters

- `selector` **string** The selector string

Returns **string** The selector with colons escaped (One backslash to escape the ':' for CSS, and other to escape the first one for JS)

promiseTimeout

Runs a promise and throws an error if it's not resolved before the timeout

Parameters

- `promise` **promise** The promise to run
- `timeout` **number** The timeout time, in ms

interleaveArrayToObject

Transforms an interleave array into a key - value object

Parameters

- `interleaveArray` **array** The interleave array

Returns **object** The key value object

objectToEncodedUri

Given an object, transforms it's properties to a URL encoded string

Parameters

- `object` **object** The object to transform

Returns **string** The URL Enconded object

sleep

Creates some delay

Parameters

- `delay` **number** Delay in miliseconds

Returns **promise** The promise that will solve after the delay

Example

```
const HeadlessChrome = require('simple-headless-chrome')

const browser = new HeadlessChrome({
  headless: true // If you turn this off, you can actually see the browser navigate with your instructions
  // see above if using remote interface
})
async function navigateWebsite() {
  try {
    await browser.init()

    const mainTab = await browser.newTab({ privateTab: false })

    // Navigate to a URL
    await mainTab.goTo('http://www.mywebsite.com/login')

    // Fill an element
    await mainTab.fill('#username', 'myUser')

    // Type in an element
    await mainTab.type('#password', 'Yey!ImAPassword!')

    // Click on a button
    await mainTab.click('#Login')

    // Log some info in your console
    await mainTab.log('Click login')

    // Wait some time! (2s)
    await mainTab.wait(2000)

    // Log some info in your console, ONLY if you started the app in DEBUG mode (DEBUG='HeadlessChrome*' npm start)
    await mainTab.debugLog('Waiting 5 seconds to give some time to all the redirects')

    // Navigate a little...
    await mainTab.goTo('http://www.mywebsite.com/myProfile')

    // Check the select current value
    const myCurrentSubscriptionPlan = await mainTab.getValue('#subscriptionSelect')
    console.log(myCurrentSubscriptionPlan) // {type: 'string', value: '1 month' }

    // Edit the subscription
    await mainTab.select('#subscriptionSelect', '3 months')
    await mainTab.click('#Save')
```

```
    // Resize the viewport to full screen size (One use is to take full size screen shots)
    await mainTab.resizeFullScreen()

    // Take a screenshot
    await mainTab.saveScreenshot('./shc.png')

    // Get a HTML tag value based on class id
    const htmlTag = await mainTab.evaluate(function(selector) {
        const selectorHtml = document.querySelector(selector)
        return selectorHtml.innerHTML
    }, '.main'); // returns innerHTML of first matching selector for class "main"

    // Close the browser
    await browser.close()
  } catch (err) {
    console.log('ERROR!', err)
  }
}
navigateWebsite()
```

TODO:

Better docs

Add more methods

- [x] .waitForSelector
- [x] .setCookie (set individual cookie) Thanks @saidganim !
- [ ] .setCookies (set a full object of cookies, like the one from .getCookies())

Support more Chrome flags

- [x] --disable-translate
- [x] --disable-extensions
- [x] --no-first-run
- [x] And many more! Only those useful... All supported thanks to @hugorodrigues. Now just pass an array in the init settings, like this:

```
const browser = new HeadlessChrome({
    headless: false, // If you turn this off, you can actually see the browser navigate with your instructions
    chrome: {
      flags: [
        '--use-fake-device-for-media-stream',
        '--use-fake-ui-for-media-stream'
      ]
    }
  })
```

And more...

- [ ] Handle xpath besides regular selectors
- [ ] Separate the methods in the actions file in actions per Domain (see left menu here: https://chromedevtools.github.io/devtools-protocol/tot/)
- [x] Allow adding new targets/tabs and controlling them at the same time (https://github.com/cyrus-and/chrome-remote-interface#cdpnewoptions-callback and https://github.com/cyrus-and/chrome-remote-interface/wiki/Inspect-a-new-tab). Thanks @iyttor ! This was a great contribution! :D

- [ ] Improve existing methods: .getCookies - Should receive a cookie name and return only that one, or all the cookies if no key is specified
- [x] Bypass Certificate Errors (https://github.com/cyrus-and/chrome-remote-interface/wiki/Bypass-certificate-errors-(%22Your-connection-is-not-private%22)) Thanks @trevan !
- [x] Add Target domain API So we can create tabs: https://chromedevtools.github.io/devtools-protocol/tot/Target/#method-createTarget

Tests

I was thinking on using this HTML page to make all the tests: https://github.com/cbracco/html5-test-page

It'd be great to have some unit tests for each HTML element; besides, those test may be useful examples for everyone.

More examples!!!