Headless Browser experiments

John HR Schuster

Table of Contents

ootstrap	. 1
Node Initilazion	. 1
Install puppeteer	. 1
cample code	. 2
T / GITHub	. 3
.gitignore file	. 3
Create local GIT.	
Create remote GitHub	. 4
Sync local to remote	. 5

A headless browser is a web browser without a graphical user interface. Headless browsers provide automated control of a web page in an environment similar to popular web browsers, but are executed via a command-line interface or using network communication.

I plan to use the headless browser for the Camunda project.

This project is sourced in ~\Dropbox\MyDev\Headless\Browser

Bootstrap

There are quite a few headless browser opyions. After some research I have selected the puppeteer package.

Reference: https://github.com/GoogleChrome/puppeteer

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

Node Initilazion

This creates a Node package. json file used to configure the project.

npm Init

Install puppeteer

In general, the rule of thumb is:

- 1. If you're installing something that you want to use in your program, using require('whatever'), then install it locally, at the root of your project.
- 2. If you're installing something that you want to use in your shell, on the command line or something, install it globally, so that its binaries end up in your PATH environment variable.

I went with Option 1.

npm install puppeteer



When you install Puppeteer, it downloads a recent version of Chromium (~170Mb Mac, ~282Mb Linux, ~280Mb Win) that is guaranteed to work with the API. To skip the download



Remember to add node_modules to the .gitignore file!!

This added the following to the package. json file.

```
"dependencies": {
    "puppeteer": "^1.2.0"
}
```

Example code

The example code is based on the web site GeekMustHave.com, best website on the planet. The example code uses ES6 syntax.

```
// --- Sample code from https://github.com/GoogleChrome/puppeteer
       const and async and await are all advanced Node functions
const puppeteer = require('puppeteer');
(async () => {
 // --- open a browser object, wait until it's open before next statement
 const browser = await puppeteer.launch();
 console.log("-- browser object created.");
 // --- open a new blank browser page, you don't actually see a page. wait tills its
open
 //
         before proceeding
 const page = await browser.newPage();
 console.log("-- Open a new blank page")
 // --- Open up the GeekMustHave web page, await until it's loaded before next
command
         again you will not see any actual page, just imagine it's there
 //
 await page.goto('https://GeekMustHave.com');
 console.log("-- Load example.com");
 // --- Now do a screen shot of the imagined page
 await page.screenshot({path: 'example.png'});
 console.log("-- Snap a PNG of the web page");
 await browser.close();
 console.log("-- Close the browser out")
})();
```

When this code is run there will be a pause just after the browser object is created.

The image below will not display in the PDF version of the documnet.

```
[Run] | npmstart.gif
```

The Node app will create a file example.png which in this case looks like.

GIT / GITHub

I'm GitHub'ing everything.

.gitignore file

Create this file before you GIT anything.

.gitignore example for this project

```
node_modules ①
.gitignore
```

1 You dont need to the node_modeles libraries you can recreate

Create local GIT

Create GIT repository, add everything (except whats named in the .gitignore file), commit it.

```
git init
git add .
git commit -mFirst-One
```

Results are

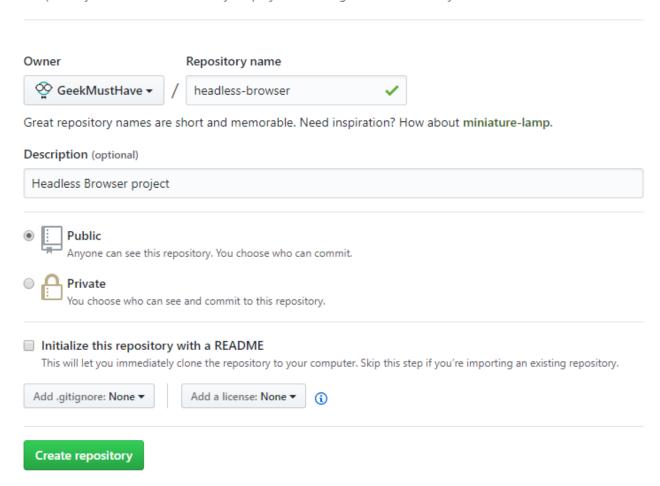
```
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser> git init
Initialized empty Git repository in
F:/users/jschust2/Dropbox/myDev/HeadlessBrowser/.git/
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser> git add .
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory.
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser>
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser> git commit -mFirst-one
[master (root-commit) d327034] First-one
 7 files changed, 1045 insertions(+)
 create mode 100644 example.js
 create mode 100644 example.png
 create mode 100644 images/npmstart.gif
 create mode 100644 package-lock.json
 create mode 100644 package.json
 create mode 100644 readme.adoc
 create mode 100644 readme.html
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser>
```

Create remote GitHub

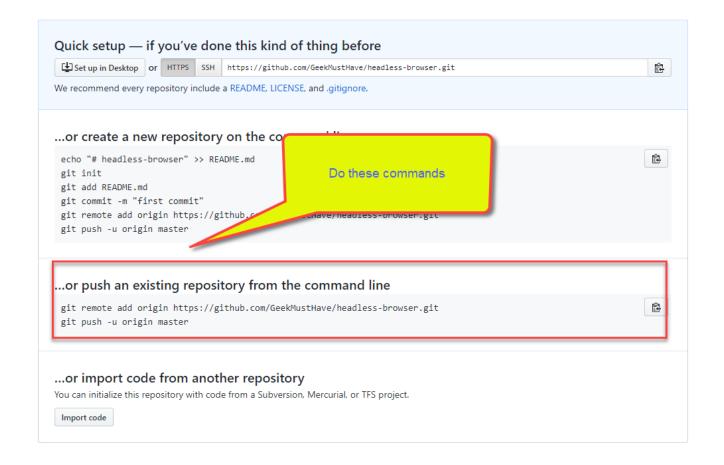
Create a new repository on GitHub.

Create a new repository

A repository contains all the files for your project, including the revision history.



Github will give you the commands to sync the local Git to the remote Git.



Sync local to remote

git remote add origin https://github.com/GeekMustHave/headless-browser.git git push -u origin master

Which results in

```
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser> git remote add origin https://github.com/GeekMustHave/headless-browser.git
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser> git push -u origin master
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (9/9), done.
Writing objects: 100% (10/10), 2.72 MiB | 1.05 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/GeekMustHave/headless-browser.git
  * [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
PS F:\users\jschust2\Dropbox\myDev\HeadlessBrowser>
```

Now the GitHub will be loaded with the project and the readme.adoc file is used as the documentation for the repository.

