

# Headless Browser experiments

John HR Schuster

Version 2.1d, 04/04/2018

# Table of Contents

Bootstrap.....	1
Node Initilazion .....	1
Install puppeteer .....	1
Example code.....	2
GIT / GitHub .....	3
.gitignore file .....	3
Create local GIT.....	3
Create remote GitHub .....	4
Sync local to remote .....	6
Grunt .....	7
Bootstrap .....	7
Grunt CLI .....	7
Enable Grunt in a Node app .....	8
Create a gruntfile.js .....	8
Grunt tasks .....	9
ASCIIDoctor.....	9

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`

The HTML version of this document is at <https://geekmusthave.com/Courses/Headless-Browser/readme.html>

The PDF version of this document is at <https://geekmusthave.com/Courses/Headless-Browser/readme.pdf>

## Bootstrap

There are quite a few headless browser options. 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](https://GeekMustHave.com), best website on the planet. The example code uses ES6 syntax.

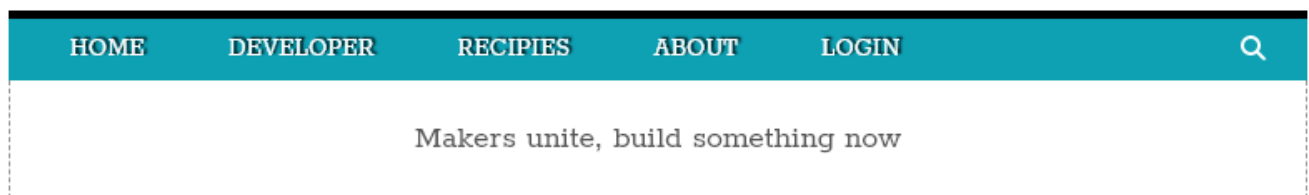
```
// --- Sample code from https://github.com/GoogleChrome/puppeteer  
//      const and async and await are all advanced Node functions  
  
const puppeteer = require('puppeteer');  
  
(async () => {  
  // --- open a browser object, wait until it's open before next statement  
  const browser = await puppeteer.launch();  
  console.log("-- browser object created.");  
  // --- open a new blank browser page, you don't actually see a page. wait till 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: './images/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 document.

[Run] | *npmstart.gif*

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



APRIL 4, 2018

Noderize Create a Node app in less than 30 seconds.

## 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 don't need to the `node_modules` libraries you can recreate

## Create local GIT

Create GIT repository, add everything (except what's 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.

Owner

 GeekMustHave ▾

Repository name

/ headless-browser ✓

Great repository names are short and memorable. Need inspiration? How about **miniature-lamp**.

Description (optional)

Headless Browser project



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾



Add a license: None ▾



Create repository

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

**Quick setup — if you've done this kind of thing before**

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/GeekMustHave/headless-browser.git` 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# headless-browser" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/GeekMustHave/headless-browser.git
git push -u origin master
```

**Do these commands**

**...or push an existing repository from the command line**

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

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

## 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.



GeekMustHave / headless-browser

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Headless Browser project

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

GeekMustHave Updated doco, created GitHub		Latest commit 85e14d6 21 seconds ago
images	Updated doco, created GitHub	21 seconds ago
example.js	Updated doco, created GitHub	21 seconds ago
example.png	First-one	20 minutes ago
package-lock.json	First-one	20 minutes ago
package.json	First-one	20 minutes ago
readme.adoc	Updated doco, created GitHub	21 seconds ago
readme.html	First-one	20 minutes ago

readme.adoc

## Headless Browser experiments

Table of Contents

- [Bootstrap](#)
  - [Node Initilazion](#)
  - [Install puppeteer](#)
- [Example code](#)
- [GIT / GitHub](#)

# Grunt

## What is Grunt?

In one word: **automation**. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it through a Gruntfile, a task runner can do most of that mundane work for you—and your team—with basically zero effort.

# Bootstrap

Before setting up Grunt ensure that your npm is up-to-date by running

```
npm update -g npm
```

## Grunt CLI

In order to get started, you'll want to install Grunt's command line interface (CLI) globally.

This will put the grunt command in your system path, allowing it to be run from any directory.

Note that installing grunt-cli does not install the Grunt task runner! The job of the Grunt CLI is simple: run the version of Grunt which has been installed next to a Gruntfile. This allows multiple versions of Grunt to be installed on the same machine simultaneously.

```
npm install -g grunt-cli
```

## Enable Grunt in a Node app

**package.json**: This file is used by npm to store metadata for projects published as npm modules. You will list grunt and the Grunt plugins your project needs as `'devDependencies'` in this file.

**Gruntfile**: This file is named Gruntfile.js and is used to configure or define tasks and load Grunt plugins. When this documentation mentions a Gruntfile it is talking about a file, which is a **Gruntfile.js**.

The easiest way to add Grunt and gruntplugins to an existing **package.json** is with the command `npm install grunt --save-dev`. Not only will this install 'grunt' locally, but it will automatically be added to the devDependencies section, using a tilde version range.

```
npm install grunt --save-dev
```

After this the **package.json** file will have a new section.

```
"devDependencies": {  
  "grunt": "^1.0.2"  
}
```

## Create a gruntfile.js

Every Gruntfile (and gruntplugin) uses this basic format, and all of your Grunt code must be specified inside this function:

```
module.exports = function(grunt) {  
  // Do grunt-related things in here  
};
```

Most Grunt tasks rely on configuration data defined in an object passed to the **grunt.initConfig** method.

In this example, `grunt.file.readJSON('package.json')` imports the JSON metadata stored in **package.json** into the grunt config. Because `<% %>` template strings may reference any config properties, configuration data like filepaths and file lists may be specified this way to reduce

repetition.

You may store any arbitrary data inside of the configuration object, and as long as it doesn't conflict with properties your tasks require, it will be otherwise ignored. Also, because this is JavaScript, you're not limited to JSON; you may use any valid JS here. You can even programmatically generate the configuration if necessary.

*Example of gruntfile.js*

```
module.exports = function(grunt) {
  grunt.initConfig({
    pkg: grunt.file.readJSON('package.json')
  });

  // --- Other grunty stuff goes here
};
```

## Grunt tasks

### ASCIIDoctor

One of the things done in this project is the documentation which requires ASCIIDoctor.

One of the ways to handle the HTML generation is to install Ruby, GEM and the ASCIIDOCTOR GEM. This is quite a bit of work if you have never done it before.

A Grunt plugin that uses AsciiDoctor via AsciiDoctor.js to process AsciiDoc source files within the project. These files have the *\*.ADOC* extension.

Install the Grunt File plugin.

```
npm install grunt-asciidoctor --save-dev
```

Now the *package.json* files dev dependancies section looks like

```
"devDependencies": {
  "grunt": "^1.0.2",
  "grunt-asciidoctor": "^0.2.1"
}
```

Add the *grunt-asciidoctor* to the *gruntfile.js*

*gruntfile.js* file

```
module.exports = function(grunt) {  
  grunt.initConfig({  
    pkg: grunt.file.readJSON('package.json')  
  });  
  
  // --- Other Grunty stuff goes here  
  
  // --- Load ASCIIIDoctor plugin here  
  grunt.loadNpmTasks('grunt-asciidoctor'); ①  
};
```

① Load the plug in to be used, not cofigured yet