Overview

The objective of the project is to create a useful NPM package and publish it on the nam store. The package I developed is a website wrapper. The project is built with the core job of taking existing web apps and wrapping them into a native application for OSX or Windows. The code structure based on the micro services pattern. Where they work independent of each other. As I noticed there was no exiting package that allows for this development for your own apps or so you can have a whatsapp app for your desktop. This solves it.

Github Repo: https://github.com/RobertJGabriel/websiteWrapper
Npm Location: https://www.npmjs.com/package/websitewrapper

Package uses

FS
Create and delete files in node

NW-BUILDER
Lets you call all Node.js modules directly from DOM and enables a new way of writing applications.
Npm Location: https://github.com/nwjs/nw-builder

FILE-MOVE
This is a module to move file
Npm Location: https://www.npmjs.com/package/file-move

GULP-RENAME
gulp-rename is a gulp plugin to rename files easily.
Npm Location: https://www.npmjs.com/package/gulp-rename

GULP
gulp is a toolkit that helps you automate painful or time-consuming tasks in your development
Npm Location: https://www.npmjs.com/package/gulp

OSENV
Look up environment settings specific to different operating systems.
Npm Location: https://www.npmjs.com/package/osenv

CHALK
Terminal string styling done right. Much color.
Npm Location: https://www.npmjs.com/package/chalk

username
Get the username of the current user.
Npm Location: https://www.npmjs.com/package/username

Key Features

1. Builds a native app for OS X automatically.
2. Builds a native exe for Windows automatically.
3. Allows for auto building of applications
4. Built to work in command and in Node/Gulp.js file.

Installation

Open up cmd/terminal and type the following
npm install websitewrapper -g

Usage

You need to pass three things into it.

-u or -url : The url of the website/web app you want to create the shortcut for.
-t or -title : The title of the website.
-i or icon: The icon (its full path) you want pass (To convert images to icns visit iconverticons)
It will then create a folder called build on your desktop. Everything is in there now.

Examples

COMMAND

websitewrapper -url http://www.google.ie -title Google -icon /Users/robertjgabriel/desktop/icon.icn

CODE

```
var websitewrapper = require('websitewrapper'),
    websitewrapper.create(“http://www.google.ie","google","logo.icns");
```

The way the code works is that the user passes in 3 parameters either by using required and the built in create function or though the command line. You can add the code example into your gulp file will allow for automated compiling of your app.

How the code works and problems I ran into

This information that is passed is then injected into a html template

```
  var html = '<!DOCTYPE html>' +
      '<html lang="en">' +
      '<head style="border:0px;margin:0px;">' +
      '<title>' + title + '</title>' +
      '<script>' +
      'var gui=require("nw.gui"),menu=new gui.Menu({type:"menubar"}),menuItems=new
gui.Menu;menu.createMacBuiltin("' + title + '",{hideEdit:!1,hideWindow:!
0}),gui.Window.get().menu=menu;' +
      '</script>' +
      '</head>' +
      '<body style="border:0px;margin:0px;">' +
      '<iframe src="' + url + '" style="border:0;width:100vw;height:100vw;padding:0;"></iframe>' +
      '</body>' +
      '</html>';
```

Which is then saved into a folder called temp under the file name index.html. This temp folder is used to build the app, so we also store the default package file settings for node and the icon.
As seen below it injects the html string into a file called index.html and save it. It error checks swell. It will return it to the user so they can also debug it.

```
      fs.writeFile(__dirname + '/temp/index.html', html, function(err) {
        if (err) {
          return console.log(chalk.red.bold(err));
        }
        console.log(chalk.green.bold("The file was saved!"));
      });
```

Then nw.js files moves these files along with the icon that the user passed in and complies it into a folder on the users desktop.

This was done with the username module in node. The reason for this, was cause if the user was in sudo node it appears as root and not as the login user. This module solves that.
The code also does checks for correct permissions and file handling on both windows and osx.
As you can see below it gets the user name of the logged in user even though the user might be in sudo mode, which changes it root.

```
    /**
     * Gets the current username that is logged in... Not root if in sudo mode
     * @param  {String} Path to the icons
     */
    username().then(username => {

                            });
```

Then after depending on the os, it will create different build paths. For win32 is windows and darwin is OSX. It uses nodes process function to get homedrive which is the drive that node is installed on. The homeopath is the location to the user path in windows and then I built the rest to the desktop. Where in OSX is just username.

```
/**
 * Gets the platform and the the build path for either windows and mac
 * @param  {String} Path to the icons
 */
if (platform === "darwin") { //Mac
  buildPath = "/Users/" + username + "/Desktop/" + "build";
} else if (platform === "win32") {
  buildPath = process.env.HOMEDRIVE + process.env.HOMEPATH + "\\Desktop\\" + "build";
} else {
  return console.log(chalk.red.bold("not supported platform"));
}
```

The use of the gulp-rename renames the icon the user passes in to the icon.icns,

```
gulp.src(icons)
  .pipe(rename("icon.icns"))
  .pipe(gulp.dest(__dirname + '/temp'));
```

The below is the example for the build settings used for compiling the files into the app or exe programme. As you can see we also state the platforms to compile for. All the information we have got ready is used here.

```
var nw = new NwBuilder({
          files: [__dirname + '/temp/index.html', __dirname + '/temp/icon.icns', __dirname + '/temp/
                  package.json', __dirname + 'node_modules/**'], // use the glob format
  platforms: ['osx64', 'win64'],
  appName: title,
  macIcns: __dirname + "/temp/icon.icns",
  buildDir: buildPath,
  version: "0.12.0",
  zip: false
});
```

* __dirname = the location of the module
* buildPath = the full path to the users desktop
* version = is the chrome api version to use for the node webkit

Conclusion

This Npm package has proven to be popular already with over 2000 plus downloads on NPM. Several (10) Stars and forks on github. It was shared on twitter and reddit aswell. The over experience of creating was fun as I never made a nam package correctly and I took the time to create the command line features which I loved. I found the username problem when in sudo su fun to solve to be honest. Over all I learnt a lot and got pretty good at writing clean code that works well .