Scripts

Command-line interface (CLI) applications.

Getting started

Setup takes around 2 minutes. Once setup you'll be able to write scripts using **Node.js**. Setup involves creating a terminal alias that runs the bin/cast.js node script. Then to add new commands you create new files in the `commands/` directory. This setup assumes you are using the Bash terminal, but other terminals work if you know how to setup an alias (*Google "How do I setup an alias in terminal XYZ" if you aren't using Bash*).

1. Fork the project to your GitHub account. (click me)

2. Clone the project locally. Then `cd` (short for *change directory*) into it.

```
git clone https://github.com/<your-username>/scripts
cd scripts
```

3. Install project dependencies via `npm install`.

```
npm install
```

4. Open the project in your favorite code editor. Popular options:

```
code .
~or~
atom .
```

5. Read the bin/cast.js file. It uses two built-in Node.js modules `fs` and `path`. It assigns the **command** constant to `process.argv[2]`, which is the third argument in the argument vector. Then it builds a path that should reference the relevant script file in the `commands/` directory. Notice that it appends the `.js` file extension to the **command** constant. It then does two checks; the first looks to see if you passed it a command argument and the second checks that the script file exists. It then requires the script and runs the script.

6. Open the `~/.bash_profile` file. The tilde (~) is an alias to your home directory. You can `cd` to your home directory by not passing any arguments. Popular options:

```
code ~/.bash_profile
~or~
atom ~/.bash_profile
```

7. Write your **cast** alias. This will allow you to run **cast** inside any of your terminals. The alias is simply a reference to **node** passing it the full path of our bin/cast.js file. A helpful way of getting the full path is to right click the `cast.js` file in your code editor via the tree view and selecting **Copy Full Path** or **Copy Path**.

```
alias cast='node "/full/path/to/scripts/bin/cast.js"'
```

8. Open a new terminal. Because `~/.bash_profile` is only read once when you start a terminal any changes to this file will not affect any open terminals. We'll need to open a new terminal to test new changes. Forgetting this fact can cause some confusion when adding aliases to `~/.bash_profile`.

9. Run `cast` in a new terminal. This is our new bash alias. If setup correctly in the `~/.bash_profile` file you should see the command synopsis (*shown below*). This is the first of the two checks in the bin/cast.js file.

```
cast
//=> usage: cast <command> [<args>]
```

10. Run `cast hello`.

```
cast hello
//=> Missing script: /home/wurde/Code/scripts/commands/hello.js
```

11. Write the `commands/hello.js` file. Copy and paste the following code:

```
module.exports = (args) => {
  console.log('Hello world!', args)
}
```

12. Run `cast hello` again. This time it'll execute our code.

```
cast hello
//=> Running script: /home/wurde/Code/scripts/commands/hello.js
//=> Hello world! [
//=>   '/usr/bin/node',
//=>   '/home/wurde/Code/scripts/bin/cast.js',
//=>   'hello'
//=> ]
```

13. Add a `gc` alias to the `~/.bash_profile` file. You're not limited to creating a single alias. Sometimes you'll want a shorthand reference to save yourself some typing.

```
alias cast='node "/full/path/to/scripts/bin/cast.js"'
alias gc='cast gitcommit'
```

14. Open a new terminal and run `gc` inside your scripts project.

```
gc
//=> Running script: /home/wurde/Code/scripts/commands/gitcommit.js
//=> Message: Added the *hello* command
```

License