How do I add a custom script to my package.json file that runs a javascript file?

Asked 3 years, 3 months ago Active 1 year ago Viewed 96k times

109

21

I want to be able to execute the command `script1` in a project directory that will run `node script1.js` .

`script1.js` is a file in the same directory. The command needs to be specific to the project directory, meaning that if I send someone else the project folder, they will be able to run the same command.

So far I've tried adding:

```
"scripts": {
    "script1": "node script1.js"
}
```

to my package.json file but when I try running `script1` I get the following output:

```
zsh: command not found: script1
```

Does anyone know the steps necessary to add the script mentioned above to the project folder?

*Note: the command can not be added to the bash profile (cannot be a machine specific command)

Please let me know if you need any clarification.

javascript node.js package.json run-script

edited Jul 12 '18 at 18:20
Hongarc
2,620 2 11 26

asked Apr 5 '16 at 17:45
Jake.JS
837 2 8 15

3 how are you trying to run it? are you using "npm run script1"? – niorad Apr 5 '16 at 17:49

3 did you run script1 using `npm run script1` ? – Claudiordgz Apr 5 '16 at 17:49

in my terminal I need to be able to type the one word command called `script1` which should run `node script1.js` – Jake.JS Apr 5 '16 at 18:25

4 Answers

179

Custom Scripts

```
npm run-script <custom_script_name>
```

or

```
npm run <custom_script_name>
```

In your example, you would want to run `npm run-script script1` or `npm run script1` .

See https://docs.npmjs.com/cli/run-script

Lifecycle Scripts

Node also allows you to run custom scripts for certain lifecycle events, like after `npm install` is run. These can be found here.

For example:

```
"scripts": {
    "postinstall": "electron-rebuild",
},
```

This would run `electron-rebuild` after a `npm install` command.

edited Aug 29 '17 at 0:12
Joshua Pinter
26.8k 9 155 178

answered Jan 27 '17 at 17:42
wesleysmyth
1,953 2 10 11

1 `npm run-script scriptname` worked for me, however `npm run scriptname` didn't! – blueprintchris Jul 31 '18 at 12:52

@GalGrünfeld did you install sails globally (npm install -g)? – Yves Dorfsman Apr 10 at 14:54

I have, and I did some reading, aand as far as can remember from Sail's website, installing it globally (via `-g`), found out that Sails installs bash/cmdlet scripts (e.g `sails generate api <api_name>` (bash/cmdlet according to the machine it's installs on) and adds global paths to those scripts on the machine. – Gal Grünfeld May 9 at 0:12

17

I have created the following, and it's working on my system. Please try this:

package.json:

```
{
  "name": "test app",
  "version": "1.0.0",
  "scripts": {
    "start": "node script1.js"
  }
}
```

script1.js:

```
console.log('testing')
```

From your command line run the following command:

```
npm start
```

Additional use case

My package.json file has generally the following scripts, which enable me to watch my files for typescript, sass compilations and running a server as well.

```
 "scripts": {
    "start": "concurrently \"sass --watch ./style/sass:./style/css\" \"npm run tsc:w\"
\"npm run lite\" ",
    "tsc": "tsc",
    "tsc:w": "tsc -w",
    "postinstall": "typings install"
  }
```

edited Jul 20 '18 at 16:20
ruffin
9,925 5 56 102

answered Apr 5 '16 at 18:01
Sujeet Jaiswal
721 4 12

1 the command to run the `script1.js` file needs to be a custom one word command called `script1` – Jake.JS Apr 5 '16 at 18:20

then change `"start"` to `script1` , you can use any name you like, I prefer to use `start` to clearly defined what should run – Sujeet Jaiswal Apr 5 '16 at 18:29

2 @sujeet-jaiswal simply changing from start to script1 is not going to work. The word "start" is reserved in npm, so it works. The word script1 is not and it will not be recognized, even when it is defined in package.json The wesleysmyth answer above is the correct one, just add run to the call. – Predrag Stojadinović Nov 27 '17 at 10:25

11

Steps are below:

1. In package.json add:

   ```
   "bin":{
       "script1": "bin/script1.js"
   }
   ```

2. Create a `bin` folder in the project directory and add file `runScript1.js` with the code:

   ```
   #! /usr/bin/env node
   var shell = require("shelljs");
   shell.exec("node step1script.js");
   ```

3. Run `npm install shelljs` in terminal
4. Run `npm link` in terminal
5. From terminal you can now run `script1` which will run `node script1.js`

Reference: http://blog.npmjs.org/post/118810260230/building-a-simple-command-line-tool-with-npm

2,646 7 34 40

837 2 8 15

0

Example:

```
   "scripts": {
     "ng": "ng",
     "start": "ng serve",
     "build": "ng build --prod",
     "build_c": "ng build --prod && del \"../../server/front-end/*.*\" /s /q & xcopy /s
 dist \"../../server/front-end\"",
     "test": "ng test",
     "lint": "ng lint",
     "e2e": "ng e2e"
   },
```

As you can see, the script "build_c" is building the angular application, then deletes all old files from a directory, then finally copies the result build files.

answered Apr 30 '18 at 5:10
FindOutIslamNow
522 1 5 18

2 Here's some useful info about `&&` - stackoverflow.com/questions/39172536/… – Simon_Weaver May 9 '18 at 20:53