npm-run-script

Run arbitrary package scripts

SYNOPSIS

```
npm run-script <command> [--silent] [-- <args>...]

alias: npm run
```

DESCRIPTION

This runs an arbitrary command from a package's **"scripts"** object. If no **"command"** is provided, it will list the available scripts. **run[-script]** is used by the test, start, restart, and stop commands, but can be called directly, as well. When the scripts in the package are printed out, they're separated into lifecycle (test, start, restart) and directly-run scripts.

As of **npm@2.0.0** , you can use custom arguments when executing scripts. The special option **--** is used by getopt to delimit the end of the options. npm will pass all the arguments after the **--** directly to your script:

```
npm run test -- --grep="pattern"
```

The arguments will only be passed to the script specified after **npm run** and not to any pre or post script.

The **env** script is a special built-in command that can be used to list environment variables that will be available to the script at runtime. If an "env" command is defined in your package, it will take precedence over the built-in.

In addition to the shell's pre-existing **PATH** , **npm run** adds **node_modules/.bin** to the **PATH** provided to scripts. Any binaries provided by locally-installed dependencies can be used without the **node_modules/.bin** prefix. For example, if there is a **devDependency** on **tap** in your package, you should write:

```
"scripts": {"test": "tap test/\*.js"}
```

instead of

```
"scripts": {"test": "node_modules/.bin/tap test/\*.js"}
```

to run your tests.

The actual shell your script is run within is platform dependent. By default, on Unix-like systems it is the **`/bin/sh`** command, on Windows it is the **`cmd.exe`** . The actual shell referred to by **`/bin/sh`** also depends on the system. As of **`npm@5.1.0`** you can customize the shell with the **`script-shell`** configuration.

Scripts are run from the root of the module, regardless of what your current working directory is when you call **`npm run`** . If you want your script to use different behavior based on what subdirectory you're in, you can use the **`INIT_CWD`** environment variable, which holds the full path you were in when you ran **`npm run`** .

`npm run` sets the **`NODE`** environment variable to the **`node`** executable with which **`npm`** is executed. Also, if the **`--scripts-prepend-node-path`** is passed, the directory within which **`node`** resides is added to the **`PATH`** . If **`--scripts-prepend-node-path=auto`** is passed (which has been the default in **`npm`** v3), this is only performed when that **`node`** executable is not found in the **`PATH`** .

If you try to run a script without having a **`node_modules`** directory and it fails, you will be given a warning to run **`npm install`** , just in case you've forgotten.

You can use the **`--silent`** flag to prevent showing **`npm ERR!`** output on error.

You can use the **`--if-present`** flag to avoid exiting with a non-zero exit code when the script is undefined. This lets you run potentially undefined scripts without breaking the execution chain.