Unit 14: NPM Scripts

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand npm Scripts
- Understand grunt and gulp

Introduction

NPM is a Node Package Manager. It is the world's largest Software Registry. This registry contains over 800,000 code packages. Many Open-source developers use npm to share software. Many organizations also use npm to manage private development.

"npm scripts" are the entries in the scripts field of the package.json file. The scripts field holds an object where you can specify various commands and scripts that you want to expose. These can be executed using the following command.

npm run <script-name>

NPM scripts are used to automate tasks like minifying CSS, uglifying JavaScript, building project. NPM scripts are versatile and simple and by learning fewer tools, we can automate tasks in our web project.

14.1 Implementing Tools with NPM Scripts

There are typically two types of JavaScript build tools: task runners and module bundlers. In this article, we'll look at how to use project specific npm build scripts as a build tool and task runner, instead of relying on task runners like Gulp and Grunt or module bundlers like Webpack.

Grunt and Gulp are task runners, they automate recurring manual tasks in a development process. Webpack, on the other hand, is a module bundler which takes separate pieces of application code, each with their own dependencies, and bundles them into static assets.

In our Grunt and Gulp article, we mentioned another build tool alternative that we didn't really go into: npm scripts. A few of you asked in the comments that you'd like to know more about how you can use npm scripts as a build tool/task runner, so that's exactly what we're going to look at.

Why Npm Scripts?

If you use build tools like Grunt, Gulp, and Webpack for long enough, you'll begin to find that you start fighting with the tool rather than focusing on writing the code for your application. Each tool has its own way of doing things and that means that each tool comes with its own quirks and gotchas that need to be learned.

Both Grunt and Gulp heavily rely on the use of plugins to add functionality, and most of those plugins are wrappers around existing npm packages. This can lead to problems like:

1. There isn't a plugin for the package you want to use.
2. The plugin is out of date and doesn't support the underlying package properly.
3. The plugin doesn't support a feature you'd like to use for the underlying package.
4. The plugin documentation is lacking or unclear.
5. The plugin doesn't handle errors well.

Since most packages usually have a CLI, a simple solution to these problems would be to remove the (sometimes complex) abstraction of build tools altogether and run the underlying packages manually on the command-line. This is a great solution, but how are you going to remember all of those CLI commands and their options? And how are you going to chain them together? Wouldn't it be nice if you could just run a single CLI command and have them all run in the right order, and at the same time?

Npm Scripts

Enter npm scripts, which have been around since at least version 6 of npm. Using the scripts property in your package.json file, it's possible to define custom scripts to run various CLI commands. This is very similar to the GNU Make tool, used by desktop application developers for Unix-like systems to build and manage their applications.

Once you define your scripts in your package.json, npm has a run command that can execute these scripts. If you've ever used a package that asked you to run a command like npm start or npm run test, then you've used npm scripts.

A script could be as simple as a single terminal command, or a more complex series of commands that need to be run in a particular order.

To use npm scripts as a build tool we're going to define a bunch of scripts in a package.json file, similar to defining the tasks we want to run in a config file in other build tools. The difference with npm scripts is that we're going to run the package CLI without any plugins, then chain the scripts together so we can trigger a build with a single command. For demonstration purposes we're going to recreate the same build process that we used in the Gulp vs Grunt article:

1. Compile Sass to CSS.
2. Concatenate and minify CSS and JavaScript.
3. Optimize images.

14.2 Grunt

Grunt is a JavaScript task runner, a tool used to automatically perform frequent tasks such as minification, compilation, unit testing, and linting. It uses a command-line interface to run custom

tasks defined in a file. Grunt was created by Ben Alman and is written in Node.js. It is distributed via npm.

Why Use Grunt?

Grunt can perform repetitive tasks very easily, such as compilation, unit testing, minifying files, running tests, etc.

Grunt includes built-in tasks that extend the functionality of your plugins and scripts.

The ecosystem of Grunt is huge; you can automate anything with very less effort.

History

The first lines of source code were added to GruntJS in 2011. The Grunt v0.4 was released on February 18, 2013. The Grunt v0.4.5 was released on May 12, 2014. The stable version of Grunt is 1.0.0 rc1 which was released on February 11, 2016.

Advantages

- Using Grunt, you can perform minification, compilation, and testing of files easily.
- Grunt unifies the workflows of web developers.
- You can easily work with a new codebase using Grunt because it contains less infrastructure.
- It speeds up the development workflow and enhances the performance of projects.

Disadvantages

Whenever npm packages are updated, you need to wait until the author of the Grunt updates it.

Every task is designed to do specified work. If you want to extend a specified task, then you need to use some tricks to get the work done.Grunt includes a large number of configuration parameters for individual plugins. Usually, Grunt configuration files are longer in length.

Some of the most prominent features of GruntJS are listed below −

1. Grunt makes the workflow as easy as writing a setup file.
2. You can automate repetitive tasks with minimum effort.
3. Grunt is a popular task runner based on NodeJS. It is flexible and widely adopted.
4. It has a straightforward approach which includes tasks in JS and config in JSON.
5. Grunt minifies JavaScript, CSS files, testing files, compiling CSS preprocessor files (SASS, LESS), etc.
6. Grunt includes built-in tasks that extend the functionality of your plugins and scripts.
7. It speeds up the development workflow and enhances the performance of projects.
8. You can easily work with a new codebase using Grunt because it contains less infrastructure.
9. The ecosystem of Grunt is huge; you can automate anything with very less effort.

14.3 Task Automator

Taskr is a highly performant task automation tool, much like Gulp or Grunt, but written with concurrency in mind. With Taskr, everything is a coroutine, which allows for cascading and composable tasks; but unlike Gulp, it's not limited to the stream metaphor.

Taskr is extremely extensible, so anything can be a task. Our core system will accept whatever you throw at it, resulting in a modular system of reusable plugins and tasks, connected by a declarative taskfile.js that's easy to read.

Features

- lightweight: with 6 dependencies, installation takes seconds
- minimal API: Taskr only exposes a couple methods, but they're everything you'll ever need
- performant: because of Bluebird, creating and running Tasks are quick and inexpensive
- cascading: sequential Task chains can cascade their return values, becoming the next Task's argument
- asynchronous: concurrent Task chains run without side effects & can be yielded consistently
- composable: chain APIs and Tasks directly; say goodbye to pipe() x 100!
- modular: easily share or export individual Tasks or Plugins for later use
- stable: requires Node >= 4.6 to run (LTS is 6.11)

 **Example**

Here's a simple taskfile (with shorthand generator methods) depicting a parallel chain.

```
const sass = 'src/{admin,client}/*.sass';
constjs = 'src/{admin,client}/*.js';
constdist = 'build';

module.exports = {
 *lint(task) {
   yield task.source(js).xo({ esnext:true });
 },
 *scripts(task) {
   yield task.source(js).babel({ presets:['es2015'] }).target(`${dist}/js`);
 },
 *styles(task) {
   yield task.source(sass).sass({ outputStyle:'compressed' }).autoprefixer().target(`${dist}/css`);
 },
 *build(task) {
   yield task.parallel(['lint', 'scripts', 'styles']);
 }
}
```

Concepts

Core

Taskr is a task runner. It's designed to get you from A to B -- that's it.

If it helps, imagine you're dining in a restaurant and Taskr is the food runner. Taskr's role is solely to collect meals from the kitchen (task.source) and deliver them to the correct table (task.target). As a food runner, Taskr may do this one plate at a time (task.serial) or deliver multiple plates at once (task.parallel). Either way, Taskr only cares about going from A to B. It may not be the most glamorous job, but as far as you (the patron) are concerned, it's incredibly important because it brings you food.

Plugins

Because Taskr is single-minded and cares only about executing tasks, everything else is a plugin. This keeps development with Taskr easy, approachable, and lightweight.

You see, installing Taskr gives access to a reliable task runner. You decide what it can do, provide it functionality, and dictate when to do it. You're in full control.

Through plugins, you are able to capture useful behavior and share them across tasks or projects for repeated use. Plugins come in three flavors:

external - installed via NPM; called "external" because they live outside your codebase

inline - generally simple, one-time functions; not sensible for reuse since declared within a task (hence "inline")

local - private, reusable plugins; appear exactly like external plugins but not public on NPM.

Tasks

Tasks are used to tell Taskr what to do. They are written as generator functions & converted to coroutines internally. They're also fully self-contained and, like plugins, can be shared across projects if desired.

Upon runtime, tasks are cheap to create, so are also destroyed once completed. This also helps Taskr remain efficient; history won't weigh it down.

Lastly, tasks have the power to start other Tasks, including serial and parallel chains!

Taskfiles

Much like Gulp, Taskr uses a taskfile.js (case sensitive) to read and run your Tasks. However, because it's a regular JavaScript file, you may also require() additional modules and incorporate them directly into your Tasks, without the need for a custom Plugin!

```
constbrowserSync = require('browser-sync');


exports.serve = function * (task) {
browserSync({
   port: 3000,
   server: 'dist',
   middleware: [
    require('connect-history-api-fallback')()
   ]
 });
 yield task.$.log('> Listening on localhost:3000');
}
```

Taskfiles should generally be placed in the root of your project, alongside your package.json. Although this is not required, Taskr (strongly) prefers this location.

14.4 Gulp

Gulp is a task runner that uses Node.js as a platform. It purely uses the JavaScript code and helps to run front-end tasks and large-scale web applications. Gulp builds system automated tasks like CSS and HTML minification, concatenating library files, and compiling the SASS files. These tasks can be run using Shell or Bash scripts on the command line.

This tutorial teaches you how to use Gulp to run front-end tasks and large-scale web applications in simple and easy steps. After completing this tutorial, you will find yourself at a moderate level of expertise in using Gulp from where you may take yourself to the next levels.

Why to use Gulp?

It is shorter, simpler and faster as compared to other task runner.

Uses SASS and LESS as CSS preprocessor.

Automatically refreshes page after editing the source files.

Easy to understand and build the Gulpfile.js because, it uses pure JavaScript code to build the task.

History

All the documentation of Gulp is covered by the CC0 license. Initially, Gulp v1.0.0 was released on January 15, 2015, and the current version of Gulp is v3.9.0.

Features

Provides minification and concatenation.

Uses pure JavaScript code.

Converts LESS or SASS to CSS compilation.

Manages file manipulation in the memory and enhances speed by using the Node.js platform.

Advantages

- Huge speed advantage over any other task runner
- Easy to code and understand.
- Easy to test the web applications..
- Plugins are simple to use and they are designed to do one thing at a time.
- Performs repetitive tasks repeatedly such as minifying stylesheets, compressing images, etc.

Disadvantages

- More number of dependencies and is a newcomer compared to Grunt.
- Using Gulp plugins, you cannot perform multiple tasks.
- Configuration is not as clean as Grunt.

Summary

- Grunt is a JavaScript based task runner which means it can automate repetitive tasks in a workflow and it can be used as a command line tool for JavaScript objects.
- Gulp is a task runner that uses Node.js as a platform. It purely uses the JavaScript code and helps to run front-end tasks and large-scale web applications.
- Annpm script is a convenient way to bundle common shell commands like a set of built-in and custom scripts for your project. They are typically terminal commands or a string of terminal commands that help automate repetitive tasks. In short, NPM scripts are terminal commands that perform a set of actions.

- The npm also acts as a command-line utility for the Node.js project for installing packages in the project, dependency management, and even version management.

Keywords

Npm: npm is a short form of Node Package Manager, which is the world's largest software registry. The open-source web project developers use it from the entire world to share and borrow packages.

Gulp: Gulp is a request line task runner for Node.js. Gulp lets us modernize cycles and get excess things done quickly.

Grunt: Grunt is a JavaScript task runner that helps us in automating mundane and repetitive tasks like minification, compilation, unit testing, linting, etc. Grunt has hundreds of plugins to choose from, you can use Grunt to automate just about anything with a minimum of effort.

SelfAssessment

1. Streams give higher flow control. Is this True or False?

A. True

B. False

2. Which plug-in has optimizers and compressors?

A. gulp-changed

B. gulp-plumber

C. gulp-imagemin

D. gulp-watch

3. The different kinds of Streams include _____.

A. readable,writable,divide, classic, transcend

B. readable, writable, divide, correspond, transform

C. readable,writable,duplex, classic, transform

D. readable,writable,divide, check, transform

4. While installing gulp, the first installation step would be _____.

A. $npm install gulp g savedev

B. $npm install gulp --save-dev

C. $npm install gulp g

D. $npm install gulp g --save-dev

5. Streams which play vital role in gulp, has its origin from _____.

A. Unix

B. Node.js

C. Gulpfile.js

D. All the options

6. Between Grunt and Gulp, which is relatively fast?

A. Both are of same speed

B. Neither of the same
C. Grunt
D. Gulp

7. While installing gulp with $npm install gulp --save-dev, what does --save-dev represents?
A. Missed out components of global installation will be taken here
B. Installs Gulp as a development dependency andpackage.json updated
C. Ensures CLI installation
D. triggers dependency between project and cli

8. Streams are Asynchronous. Is this True or False?
A. False
B. True

9. Before installation of Gulp, installation of _____ acts as pre-requisite.
A. Bower.js
B. Brocolli.js
C. Node.js
D. Gulp.js

10. How many types of Streams are available?
A. 5
B. 4
C. 6
D. 3

11. Which command in the CLI will trigger the 'default` task?
A. run default gulp
B. run gulp default
C. gulp
D. run gulp

12. In-memory caching is enabled with the help of which gulp plug-in?
A. gulp-sess
B. gulp-cached
C. gulp-cookie
D. gulp-tempcache

13. Which of the following command is used to start a REPL session?
A. $ node
B. $ node start
C. $ node repl

D. $ node console

14. Which of the following extension is used to save the Node.js files?

A. .js

B. .node

C. .java

D. .txt

15. All APIs of Node.JS are.

A. Asynchronous

B. Synchronous

C. Both of the above.

D. None of the above.

Answers for Self-Assessment

l.	A	2.	C	3.	C	4.	C	5.	A
6.	D	7.	B	8.	A	9.	C	10.	A
11.	C	12.	B	13.	A	14.	A	15.	A

Review Questions

1. What are the basic steps required to set up Gulp?
2. What are Task Runners in GulpJS?
3. What are the differences between gulp or grunt?
4. Why npm is required for gulp? Comment.
5. Why to use gulp?

Further Readings

Bootstrap by Example, Silvio MoratoLearning Bootstrap 4, Matt Lambart

Bootstrap in 24 Hours, By SamsJump Start Bootstrap,

Web Links

https://www.npmjs.com/