



NODE, NPM & GULP

INTRODUCTION TO FRONT-END BUILD SYSTEMS

WHO AM I

FRONT-END DEVELOPER

SASS, ES6, EMBER.JS

@LUISDDM_

Luis de Dios Martín

WHAT IS A BUILD SYSTEM?

- ▶ A build system **automatizes a series of small and tedious repetitive tasks.**
- ▶ They otherwise will be easily forgotten, especially when our small projects start to grow into much larger applications, heavy and bloated.

WHY USE A BUILD SYSTEM?

- ▶ A build system carries on some tasks that are meant to
 - ▶ keep our code tidy while we are developing it and efficient when we execute it
 - ▶ preprocess our code somehow so it will be able to run wherever we need

BUILD SYSTEMS THEN AND NOW

- ▶ **Make** was one of the first build systems. In the early days they were used to **compile code into executable formats** for an operating system.
- ▶ However, in web development, we have a completely different set of practices and operations.
- ▶ Over the past few years, there has been an interest in using build systems to more capably handle the growing complexities of our applications.

WHAT IS NODE?

- ▶ A **JavaScript platform** built on top of Google Chrome's JavaScript runtime engine, V8.
- ▶ This gives us the ability to **execute JavaScript code outside the browser**.
- ▶ Using Node, we can actually write both the **backend** and **frontend** of a web application entirely in **JavaScript**.
- ▶ Node ships with **npm**, a **package manager** that facilitates the installation, storage, and creation of modular components that can be used to create applications.

WHAT IS GULP?

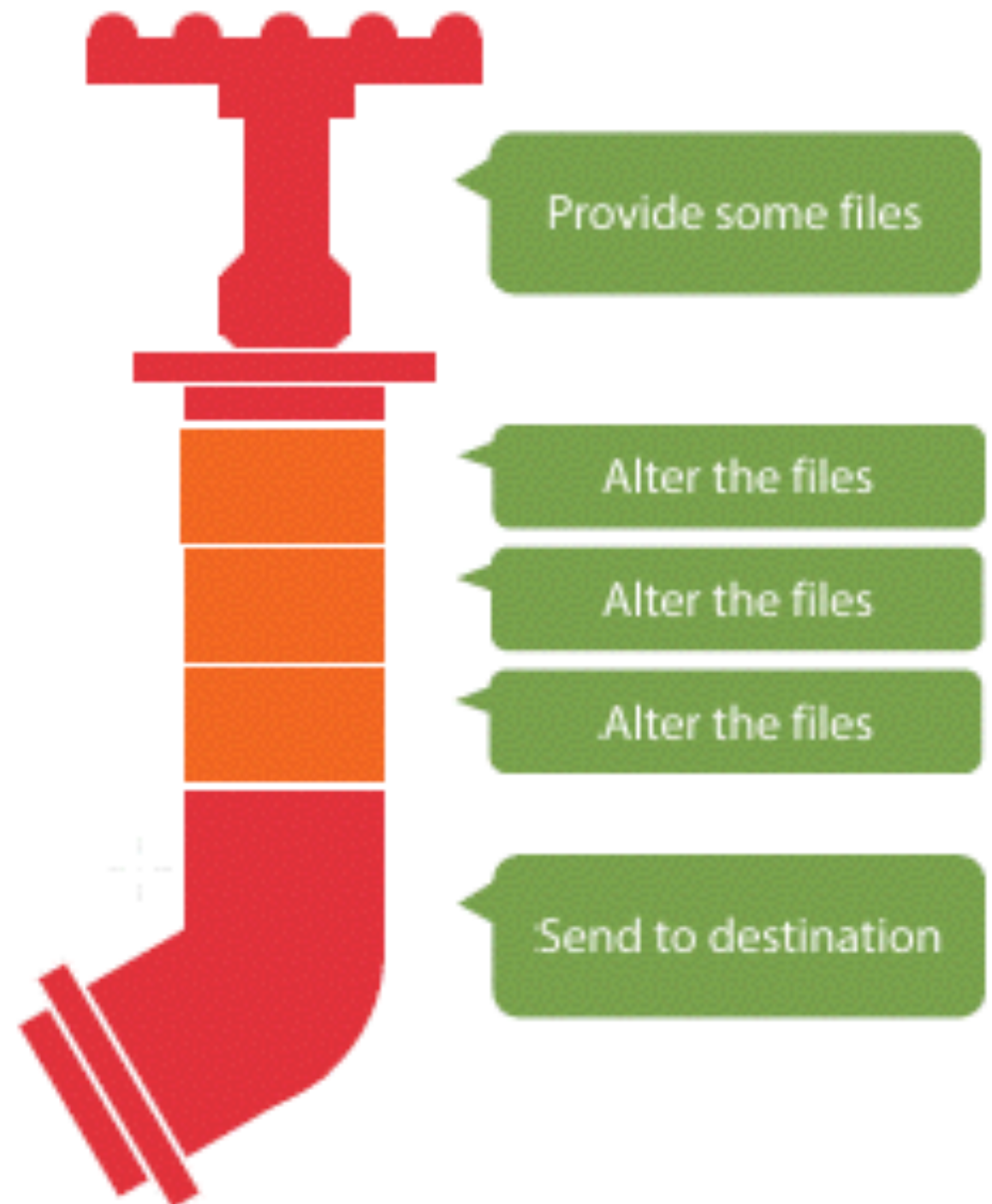
- ▶ Gulp is a **streaming JavaScript build system**.
- ▶ It leverages the power of **streams** to automate, organize, and run development tasks very quickly and efficiently.
- ▶ By simply creating a **small file of instructions**, gulp can perform just about any development task.
- ▶ Gulp uses **small, single-purpose plugins** to modify and process our project files. Additionally, we can chain, or **pipe**, these plugins together into more complex actions.
- ▶ Gulp is built upon **Node** and **npm**.

WHY USE GULP? – STREAMS (I)

- ▶ Streams were originally introduced in Unix as a way to **"pipe" together small, single-purpose applications** to perform complex, flexible operations.
- ▶ Additionally, streams were created to operate on data **without the need to buffer the entire file**, leading to quicker processing.
- ▶ Like Unix, Node has its own built-in stream module. This is what gulp uses to operate on our data and perform tasks.

WHY USE GULP? – STREAMS (II)

- ▶ This allows developers to create small **gulp plugins** or **node modules** that perform **single operations** and then **pipe** them together with **others** to perform an entire chain of actions.



WHY USE GULP? – DEPENDENCIES

- ▶ We need to **keep track of all the dependencies** that we use in our projects.
- ▶ A dependency is a software package that **contains one or several modules or plugins**.
- ▶ Node uses a file named **package.json** to store information about our project.
- ▶ npm uses this same file to **manage all of the dependencies** our project requires to run properly.

MAIN METHODS USED IN A GULP FILE

- ▶ Apart from the `package.json`, we need a `gulpfile.js` which gulp will execute each time we want to carry on a task.
- ▶ It will contain **all the tasks that we'll be able to perform**. Each of them will be build using some methods that operate on the modules or plugins provided by the dependencies.
- ▶ Each method represent a **specific purpose** and will act as the building blocks of our gulp file.

WRITING A TASK

```
gulp.task(taskName, () =>
  gulp.src(srcPath)
    .pipe(plugin1)
    .pipe(plugin2)
    .pipe(gulp.dest(destPath));
);
```


THE `.SRC()` METHOD

`.src(string || array)`

- ▶ The **source files** that we plan on modifying. It accepts either a single string or an array of strings as an argument.
- ▶ Normally we store these files in a **src** directory that will contain all our codebase. This is what we **store in our repository when we want to distribute our source code** so others can collaborate on it.

THE `.TASK()` METHOD

`.task(string, function)`

- ▶ The **basic wrapper for which we create our tasks**. It takes two arguments:
 - ▶ a string value representing the name of the task.
 - ▶ a function that will contain the code to execute upon running that task.

THE .WATCH() METHOD

`.watch(string, array)`

- ▶ **Looks for changes in the files.**
- ▶ This will **keep gulp running again right after any changes are made** so that we don't need to rerun gulp manually any time we need to process our tasks.

THE **.PIPE()** METHOD

`.pipe(function)`

- ▶ Will allow us to **pipe together smaller single-purpose plugins or applications** into a pipechain.
- ▶ This is what gives us full control of the order in which we would need to process our files.

THE `.DEST()` METHOD

`.dest(string)`

- ▶ Sets the **output destination of our processed file**. Most often, this will be used to output our data into a distribution directory.
- ▶ This distribution directory will be usually called **dist** and it will contain **all the production code**, ready to be deployed to a web server so it can be **executed** in a browser. It could also be **distributed** as a library or a plugin to use within another project.
- ▶ Note that **this code should not be uploaded to any repository**, since it will be generated automatically from **src** using gulp.

```

4  const sass          = require('gulp-sass');
5  const uglify        = require('gulp-uglify');
6  const cleanCss      = require('gulp-clean-css');
7  const htmlReplace   = require('gulp-html-replace');
8  const babel         = require('gulp-babel');
9  const tar           = require('gulp-tar');           // https://www.npmjs.com/package/gulp-tar
10 const gzip          = require('gulp-gzip');          // https://www.npmjs.com/package/gulp-gzip
11
12 gulp.task('scss', () =>
13   gulp.src('src/scss/styles.scss')
14     .pipe(sass().on('error', sass.logError))
15     .pipe(cleanCss())
16     .pipe(gulp.dest('dist/css'))
17 );
18
19 gulp.task('es6', () =>
20   gulp.src('src/js/*.js')
21     .pipe(concat('scripts.js'))
22     .pipe(babel({ presets: ['es2015'] }))
23     .pipe(uglify())
24     .pipe(gulp.dest('dist/js'))
25 );
26
27 gulp.task('html', () =>
28   gulp.src('src/index.html')
29     .pipe(htmlReplace({
30       css: 'css/styles.css',
31       js: 'js/scripts.js',
32     }))
33     .pipe(gulp.dest('dist'))
34 );
35
36 gulp.task('compress', () =>
37   gulp.src('dist/*')
38     .pipe(tar('code.tar')) // Pack all the files together
39     .pipe(gzip())          // Compress the package using gzip
40     .pipe(gulp.dest('.'))
41 );

```

SHOW ME THE CODE!

GET THIS!

github.com/luisddm



gulp-examples

Get the code and this presentation