Front-end BOT (Gulp) in Drupal

Who Am I ?

- UI Lead at Iksula
- Core committee member at Drupal Camp Mumbai 2017
- Lead Front-end developer for Drupal Camp Mumbai 2017 website
- Rap lover & writer
- Drupal ID: kirankadam911

Session will cover

- ➔ Why Gulp?
- ➔ What we're setting up
- ➔ Gulp Installation
- ➔ Creating a Gulp Project
- ➔ Writing First Gulp Task & Use it
- ➔ Combining Gulp tasks

Why Gulp?

- ➔ Referred as "**build tools**", Because running the tasks for building a website
- ➔ Gulp configurations much **shorter** and **simpler**
- ➔ Gulp also tends to run **faster**
- ➔ Gulp is a much **wider community support**

What we're setting up

By the end of this article, you'll have gulp task that will:

- ➔ Spin up a web server
- ➔ Compile Sass to CSS
- ➔ Refresh the browser automatically whenever you save a file
- ➔ Optimize all assets (CSS, JS, fonts, and images) for production

Gulp Installation

- ➔ Need to have **Node.js (Node)** installed onto your computer before you can install Gulp.
- ➔ Node.js® is a **JavaScript runtime** built on **Chrome's V8 JavaScript engine**.
- ➔ Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

Gulp Installation

➔ Now, Install Gulp by following command:
npm install gulp -g

➔ **npm install** command uses Node Package Manager (npm) to install Gulp onto your computer.

➔ The **-g** flag tells npm to install Gulp globally onto your computer.

Creating a Gulp Project

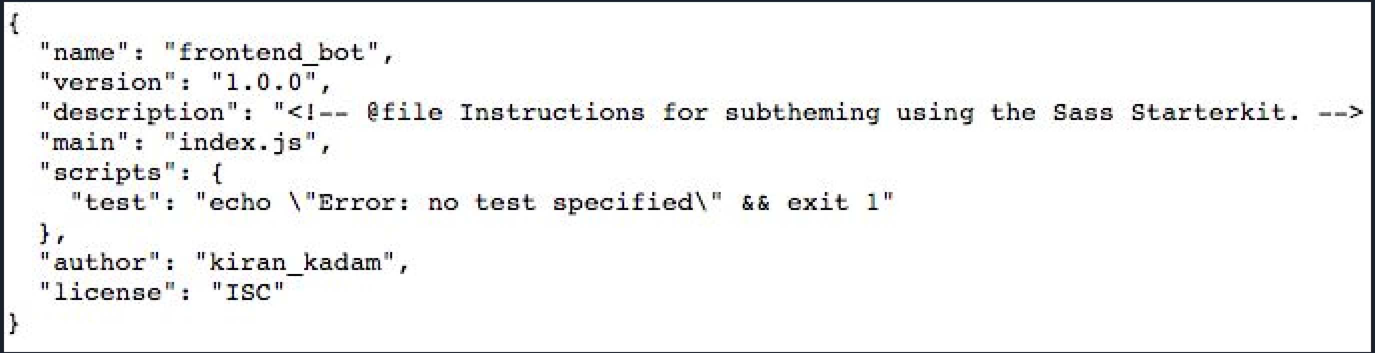
➔ Navigate to project folder from terminal:
(For e.g:
cd Applications/MAMP/htdocs/dcb2017/themes/custom/frontend_bot)

➔ Run the *npm init* command from inside that directory

➔ The **npm init** command creates a **package.json** file for project which stores information about the project.

Creating a Gulp Project

➔ Now package.json will look like



```
{
  "name": "frontend_bot",
  "version": "1.0.0",
  "description": "<!-- @file Instructions for subtheming using the Sass Starterkit. -->
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "kiran_kadam",
  "license": "ISC"
}
```

Creating a Gulp Project

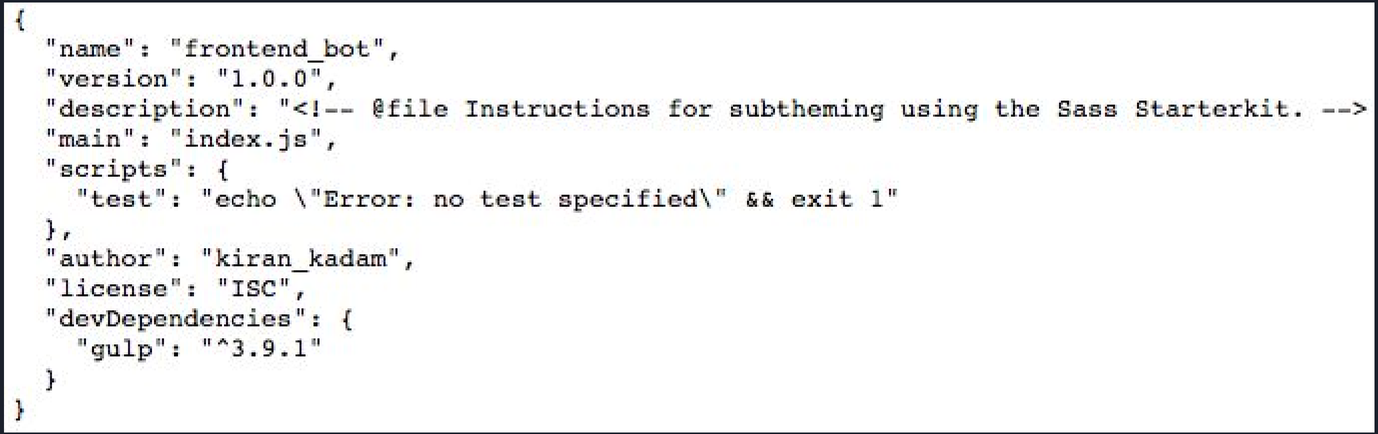
➔ Once the package.json file is created, we can install Gulp into the project by following command:
npm install gulp --save-dev

➔ This time, we're installing Gulp into **project** instead of installing it globally.

➔ **--save-dev**, which tells the computer to add **gulp** as a dev dependency in **package.json**

Creating a Gulp Project

- ➔ Now updated package.json will look like



```
{
  "name": "frontend_bot",
  "version": "1.0.0",
  "description": "<!-- @file Instructions for subtheming using the Sass Starterkit. -->
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "kiran_kadam",
  "license": "ISC",
  "devDependencies": {
    "gulp": "^3.9.1"
  }
}
```

- ➔ If you check the project folder Gulp has created a **node_modules** folder. You should also see a **gulp** folder within **node_modules.**

Writing First Gulp Task & Use it

- ➔ The first step to using Gulp is to require it in the gulpfile. (create new **gulp.js** in project root)
  var gulp = require('gulp');

- ➔ The require statement tells Node to look into the **node_modules** folder for a package named **gulp**.

- ➔ Once the package is found, we assign its contents to the variable **gulp.**

Writing First Gulp Task & Use it

- ➔ Write a gulp task with gulp variable, Syntax is:

```
gulp.task('task-name', function() {
  // Cool stuff here
});
```

- ➔ **task-name** refers to the name of the task, which would be used whenever you want to run a task in Gulp.

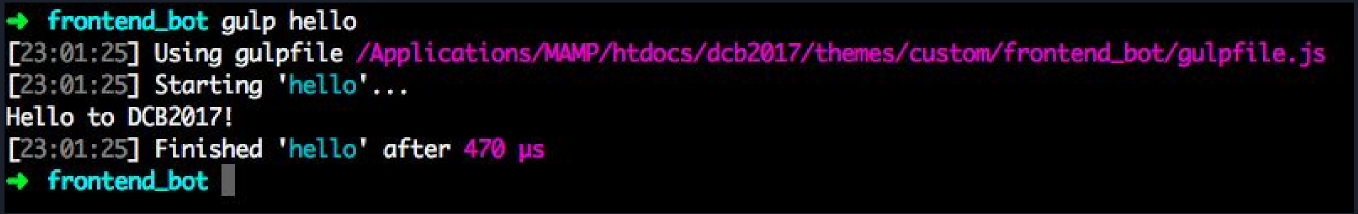
- ➔ You can also run the same task in the command line by writing *gulp task-name*.

Writing First Gulp Task & Use it

➔ Let's create a **hello** task that says Hello to DCB2017!

```
gulp.task('hello', function() {
  console.log('Hello to DCB2017!');
});
```

➔ Now run this task with *gulp hello* in the command line.

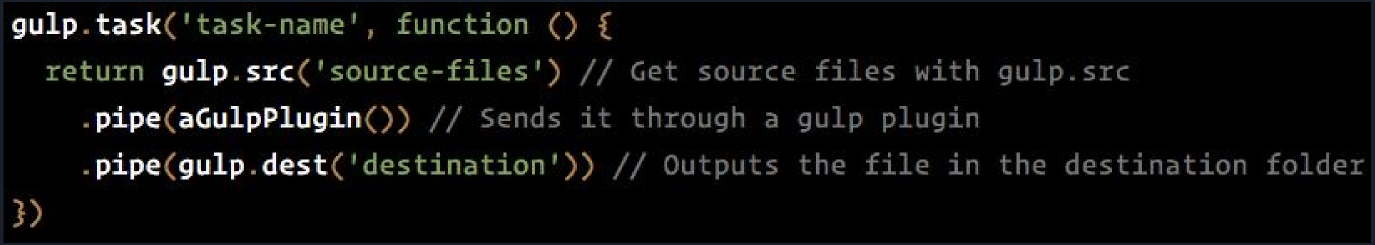


```
➜  frontend_bot gulp hello
[23:01:25] Using gulpfile /Applications/MAMP/htdocs/dcb2017/themes/custom/frontend_bot/gulpfile.js
[23:01:25] Starting 'hello'...
Hello to DCB2017!
[23:01:25] Finished 'hello' after 470 μs
➜  frontend_bot
```

Writing First Gulp Task & Use it

➔ Gulp tasks are usually a bit more complex than previous. It usually contains **two** additional Gulp methods, plus a variety of Gulp plugins.

➔ Real task may look like:



```
gulp.task('task-name', function () {
  return gulp.src('source-files') // Get source files with gulp.src
    .pipe(aGulpPlugin()) // Sends it through a gulp plugin
    .pipe(gulp.dest('destination')) // Outputs the file in the destination folder
})
```

Writing First Gulp Task & Use it

- ➔ We can compile **Sass to CSS** in Gulp with the help of a plugin called **gulp-sass**.
  Terminal: *npm install gulp-sass --save-dev*

- ➔ gulp.js: *var sass = require('gulp-sass');*
  We have to **require** gulp-sass from the **node_modules**.

- ➔ We can use gulp-sass by replacing **aGulpPlugin()** with **sass()**.

Writing First Gulp Task & Use it

- ➔ Sass task will look like below:

```
gulp.task('sass', function(){
  return gulp.src('sass/**/*.scss')
    .pipe(sass()) // Converts Sass to CSS with gulp-sass
    .pipe(gulp.dest('stylesheets'))
});
```

- ➔ Run *gulp sass* in the command line, you should now be able to see that a **style.css** file was created in **stylesheets**.

- ➔ That's how we know that the **sass** task works!

Writing First Gulp Task & Use it

➔ Continuous watching Sass files for changes task like:

```
gulp.task('sass-watch', function(){
  gulp.watch('sass/**/*.scss', ['sass']);
})
```

➔ Here **sass/**/*.scss** is a **files-to-watch** and **['sass']** is the **['tasks', 'to', 'run']** which is created in previous slide.

Writing First Gulp Task & Use it

- ➔ Live-reloading with **Browser Sync**
  Terminal: *npm install browser-sync --save-dev*

- ➔ In gulp.js:

```
var browserSync = require('browser-sync').create();
```

- ➔ Browser Sync task will look like below:

```
gulp.task('browserSync', function() {
  browserSync.init({
      proxy: "192.168.6.219:8080/dcb2017/"
  })
})
```

Browsersync

Writing First Gulp Task & Use it

- We also have to change our **sass** task slightly so Browser Sync can inject update the CSS.

- Ref. slide 17 for Sass task

```
gulp.task('sass', function() {
  return gulp.src('sass/**/*.scss')
    .pipe(sass())
    .pipe(gulp.dest('stylesheets'))
    .pipe(browserSync.reload({
      stream:true
    }));
});
```

Browsersync

Combining Gulp tasks

- ➔ You can run combine multiple tasks & run at a time by grouping them,
  Just like:

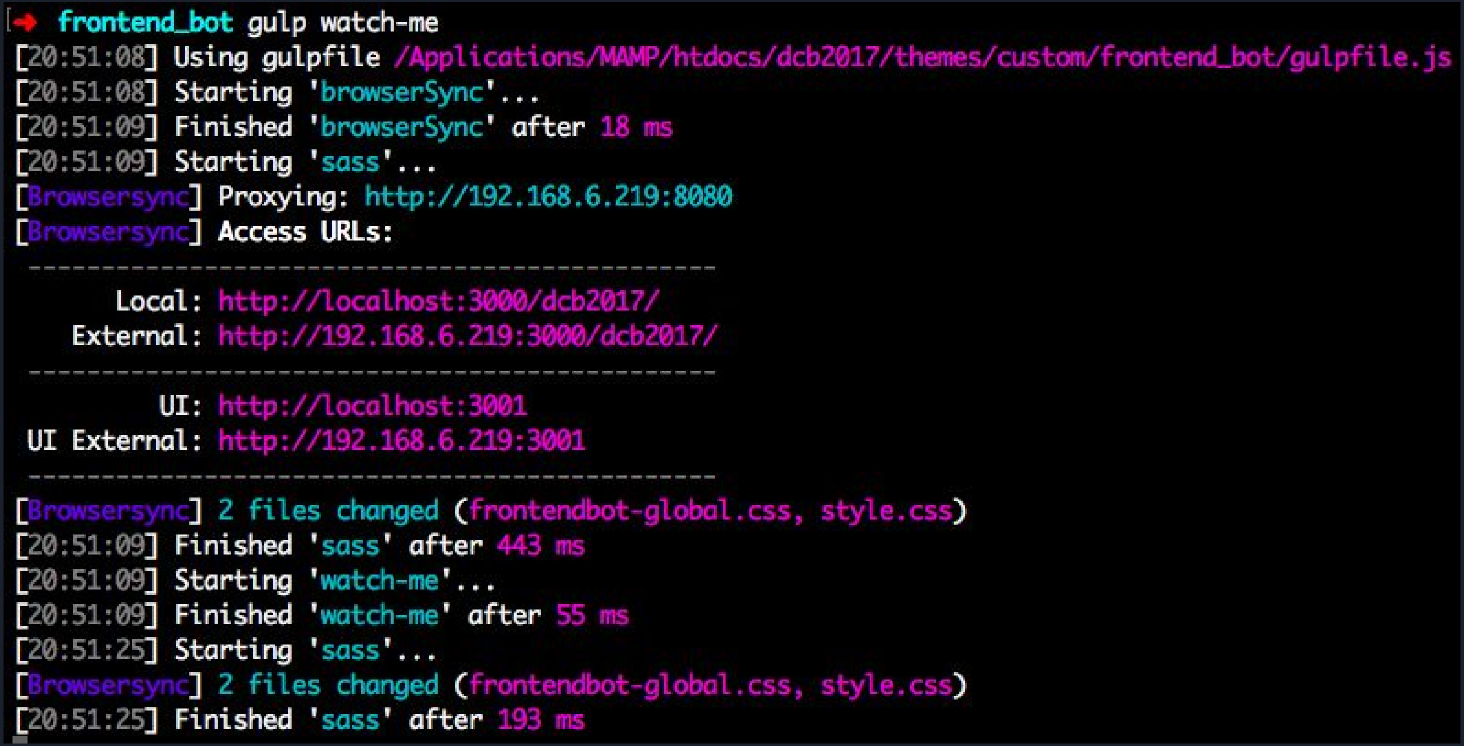
```
gulp.task('watch-me', ['browserSync', 'sass'], function (){
  gulp.watch('sass/**/*.scss', ['sass']);
  // Other watchers
});
```

- ➔ Here in watch-me task **['browserSync', 'sass']** is **array of tasks to complete before watch**.

Combining Gulp tasks

- ➔ Now, if you run **gulp watch-me** in the command line, Gulp should start both the **sass** and **browserSync** tasks concurrently.
- ➔ When both tasks are completed, **watch** will run.

Combining Gulp tasks



```
[➜  frontend_bot gulp watch-me
[20:51:08] Using gulpfile /Applications/MAMP/htdocs/dcb2017/themes/custom/frontend_bot/gulpfile.js
[20:51:08] Starting 'browserSync'...
[20:51:09] Finished 'browserSync' after 18 ms
[20:51:09] Starting 'sass'...
[Browsersync] Proxying: http://192.168.6.219:8080
[Browsersync] Access URLs:
 ----------------------------------------------
       Local: http://localhost:3000/dcb2017/
    External: http://192.168.6.219:3000/dcb2017/
 ----------------------------------------------
          UI: http://localhost:3001
 UI External: http://192.168.6.219:3001
 ----------------------------------------------
[Browsersync] 2 files changed (frontendbot-global.css, style.css)
[20:51:09] Finished 'sass' after 443 ms
[20:51:09] Starting 'watch-me'...
[20:51:09] Finished 'watch-me' after 55 ms
[20:51:25] Starting 'sass'...
[Browsersync] 2 files changed (frontendbot-global.css, style.css)
[20:51:25] Finished 'sass' after 193 ms
```

Other Gulp task

- ➔ For all other gulp task like **Optimizing CSS, JS & Images**, **CSS & JS Validation**, **Sort CSS properties alphabetically**, **CSS Auto prefixes** & **Sourcemap** you can use ready-made front-end automation, link given below.

- ➔ https://github.com/kiran-kadam911/frontend-automation

Questions

Thank you!

Be Drupaler! Spread Drupal!