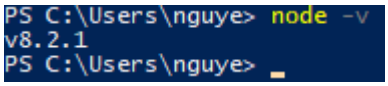
1. NodeJS & NPM

1.1 Giới thiệu Node & NPM

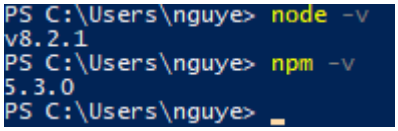
- Node là một chương trình dùng để chạy các file javascript.
- Bình thường file js chạy trên trình duyệt (chrome, firefox...) và NodeJS là một chương trình để chạy các file js trên máy tính. JAVA cần JVM để chạy, C# cần .NET để chạy... thì javascript cần NodeJS để chạy.
- Các lập trình viên viết các chương trình NodeJS rồi chia sẻ, đưa lên một nơi chung là npmjs https://www.npmjs.com.
- NPM là một tool dùng để tải và quản lí các chương trình NodeJS (package) từ cộng đồng.

1.2 Cài đặt NodeJS & NPM

- Tải về và cài đặt NodeJS như một chương trình bình thường tại đây https://nodejs.org/en/.
- Dùng lệnh node –v trong giao diện console để kiểm tra, nếu thành công sẽ báo như sau



- Sau khi cài NodeJS thì tự động đã có npm. Kiểm tra bằng lệnh npm –v



1.3 Cấu hình package.json

- Package.json là file cấu hình của npm, chứa thông tin về project các package cần cài đặt cho project, giúp cho npm hiểu cần phải cài đặt cái gì.
- File package.json được viết bằng json và được đặt ở thư mục gốc của project.

- Các thành phần của Package.json

```
{
    "name": "gardeners",
    "version": "1.0.0",
    "description": "project front-end",
    "main": "gulpfile.js",
    "dependencies": {
        "browser-sync": "^2.18.13",
        "del": "^3.0.0",
        "gulp": "^3.9.1",
        "gulp-autoprefixer": "^4.0.0",
        "gulp-cache": "^0.4.6",
        "gulp-cssnano": "^2.1.2",
        "gulp-file-include": "^1.2.0",
        "gulp-if": "^2.0.2",
        "gulp-rename": "^1.2.2",
        "gulp-sass": "^3.1.0",
        "gulp-sourcemaps": "^2.6.1",
        "gulp-uglify": "^3.0.0",
        "gulp-useref": "^3.1.2",
        "run-sequence": "^2.1.0"
    },
    "devDependencies": {},
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "nguyenhoangsin@gmail.com",
    "license": "ISC"
}
```

- Package.json chứa rất nhiều thông tin, thường thì ta chỉ quan tâm đến vài thuộc tính chính:

  name: tên của project hoặc package, đây là thuộc tính bắt buộc.

  version: phiên bản của project, cách ghi version hiện nay được quy đinh bởi Semantic Versioning http://semver.org/.

  description: đoạn mô tả của project, cần viết ngắn gọn dễ hiểu.

  author: thông tin về tác giả.

  dependencies: chứa thông tin các package sử dụng trong project.

- Tạo package.json bằng cách mở thư mục chứa project trong console rồi dùng lệnh npm init sau đó nhập lần lượt các thuộc tính hoặc bỏ trống để npm cài đặt thuộc tính mặc định

```
PS C:\Users\nguye\Documents\CNM> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cnm)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\nguye\Documents\CNM\package.json:

{
  "name": "cnm",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this ok? (yes) yes
PS C:\Users\nguye\Documents\CNM>
```

1.4 Chạy chương trình NodeJS đầu tiên

- Tạo file javascript tên helloworld.js có nội dung như sau

```
console.log('hello ', 'world')
```

- Lưu file và dùng lệnh node helloworld.js để chạy sẽ được kết quả như sau

```
PS C:\Users\nguye\Documents\CNM\1. NodeJS & NPM> node helloworld.js
hello  world
PS C:\Users\nguye\Documents\CNM\1. NodeJS & NPM>
```

1.5 Quản lí package với NPM

- Để tải các chương trình được chia sẻ ở npmjs về ta cú pháp npm install --save <tên gói> --save để cài package vào project, nếu không có lệnh này thì mặc định package sẽ tải về trong trong ổ C (chứa hệ điều hành) và cài trong phạm vi toàn cục, để cài package trong phạm vi toàn cục thì dùng từ khóa --global thay cho --save (npm install --global <tên gói>)

- Ví dụ để tải package jquery (một thư viện, framework js) vào project ta dùng lệnh sau

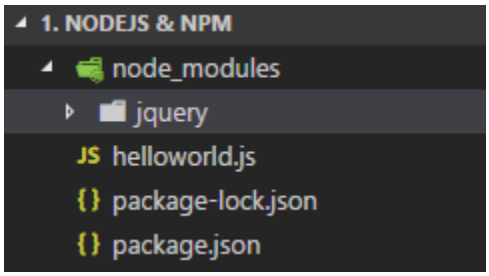
  npm install --save jquery (lưu ý cấu hình package.json trước)

  Kết quả báo install package jquery thành công trên màn hình console

```
PS C:\Users\nguye\Documents\CNM\1. NodeJS & NPM> npm install --save jquery
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN 1. NodeJS & NPM No description
npm WARN 1. NodeJS & NPM No repository field.
npm WARN 1. NodeJS & NPM No license field.

+ jquery@3.2.1
added 1 package in 2.333s
PS C:\Users\nguye\Documents\CNM\1. NodeJS & NPM>
```

  Package tải về chứa trong thư mục node_modules của project



  Mở file package.json ra xem ta sẽ thấy thông tin package vừa tải về đã có trong thuộc tính dependencies

```
{
  "name": "",
  "version": "",
  "dependencies": {
    "jquery": "^3.2.1"
  }
}
```

- Để xem danh sách các package đã cài trong phạm vi toàn cục dùng lệnh

  npm list -g --depth=0

- Để xem danh sách các package đã cài trong phạm vi cục bộ của project dùng lệnh

  npm list -s --depth=0

- Để gở bỏ một package đã cài đặt dùng lệnh (gở bỏ cả phạm vi toàn cục và cục bộ)

  npm uninstall <package_name>

- Dùng lệnh npm install để cài đặt lại toàn bộ các package được khai báo trong thuộc tính dependencies của package.json.

2. Tự động hóa task với GULP

2.1 Giới thiệu GULP

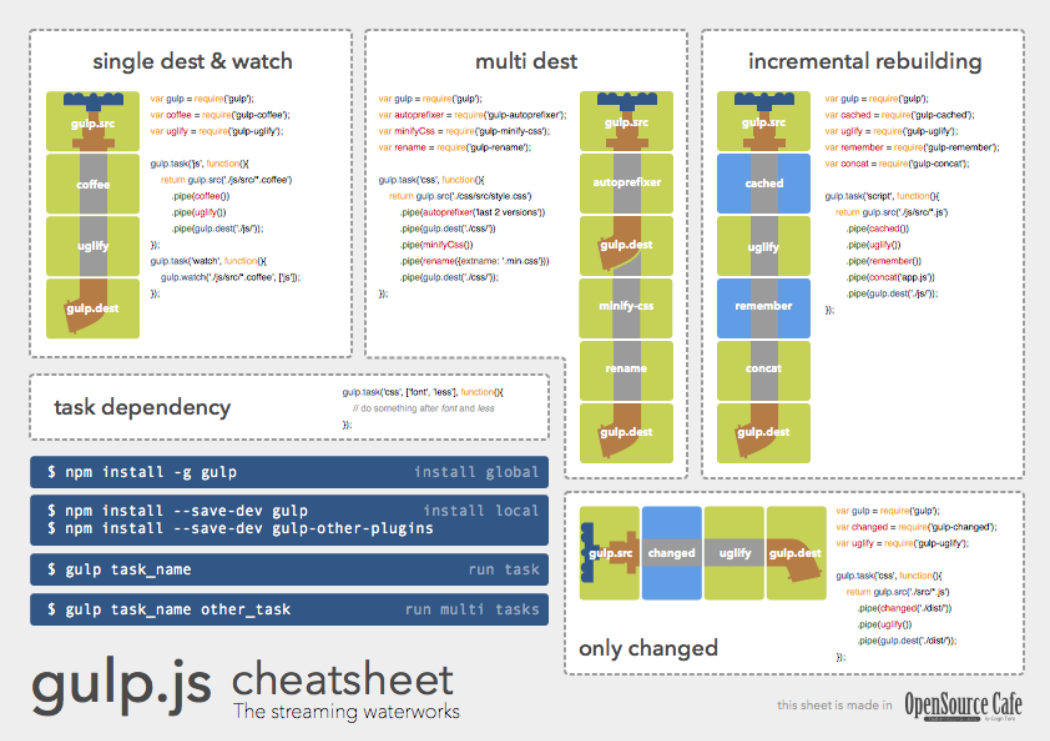
- Là web developer, hẳn ai cũng lâm vào trường hợp phải làm đi làm lại những công việc nhàm chán như việc nhấn refresh trên trình duyệt khi có file nào được lưu...
- Gulp giúp tự động hóa nhiều task (nhiệm vụ) trong quá trình phát triển web, nó thường được sử dụng để làm các tác vụ front-end như:

  Tạo ra một web server.

  Tự động refresh trình duyệt bất cứ khi nào một file được lưu.

  Tối ưu hóa các tài nguyên như CSS, JavaScript, hình ảnh.

  Đây chỉ là một trong rất nhiều những thứ mà GULP có thể làm.
- Gulp là một ứng dụng chạy các lệnh tự động giống file .bat trên dos & window.
- Cheat sheet: https://github.com/osscafe/gulp-cheatsheet/blob/master/README.md



2.2 Cài đặt GULP

- Cài đặt gulp bằng lệnh sau (phải cấu hình package.json trước):

  npm install --global gulp

  npm install --save gulp

  Phải cài gulp ở 2 nơi toàn cục và cục bộ

- Sau khi cài đặt thành công dùng lệnh gulp -v để kiểm tra sẽ được kết quả như dưới

```
PS C:\Users\nguye\Documents\CNM\2. GULP> gulp -v
[16:39:53] CLI version 3.9.1
[16:39:53] Local version 3.9.1
PS C:\Users\nguye\Documents\CNM\2. GULP>
```

2.3 Cấu hình gulpfile.js

- Các phương thức cơ bản của gulp:

  gulp.task khai báo task để gulp sẽ thực hiện những task này theo một kịch bản định sẵn.

  gulp.src khai báo đầu vào dữ liệu cho gulp task.

  gulp.dest nơi dữ liệu gulp task sẽ xuất ra sau khi xử lí.

  gulp.watch theo dõi tài nguyên trong project khi có sự thay đổi.

- Mỗi gulp.task luôn có:

  Một source (nguồn - gulp.src).

  Một destination (điểm đến - gulp.dest).

  Giữa source và destination có nhiều pipe (ống) để gọi mỗi module xử lí, và xuất kết quả đã biến đổi vào pipe kế tiếp.

- Module là một khái niệm trong NodeJS, có thể xem nó như một thư viện JS, tập hợp những tính năng ta muốn đưa vào ứng dụng.

  Để thêm một module vào ứng dụng dùng cú pháp sau

```
var <tên biến> = require('<tên module>');
```

Gulp cũng là một module, thêm module gulp vào ứng dụng với cú pháp sau

```
var gulp = require('gulp')
```

Trong bài viết này hai khái niệm module và package được xem là tương đương, cùng mang ý nghĩa là một chương trình JS chạy trên nền NodeJS.

- Cấu hình đơn giản gulpfile.js

```
var gulp = require('gulp');
gulp.task('default', function() {
    gulp.watch('src/**/*', ['default']);
    return gulp.src('src/**/*')
        .pipe(gulp.dest('dest'))
});
```

Task default sau khi chạy sẽ copy mọi thứ trong thư mục src/ sang thư mục dest/ (thuộc project).

gulp.watch tiến hành theo dõi src/ , nếu có bất kì thay đổi nào trong thư mục này thì sẽ chạy lại task default.

/**/* là ký tự đại diện có nghĩa phù hợp với mọi pattern trong thư mục hiện tại và thư mục con, trong trường hợp này là tất cả các file trong thư mục src và thư mục con của nó.

- Chạy một task xác định bằng lệnh gulp <tên task>, trong ví dụ trên dùng lệnh gulp default. Task default sẽ là task chạy mặc định khi chỉ dùng lệnh gulp.
- Để chạy một lần nhiều task chỉ vớt một lệnh gulp ta cấu hình gulp.task như sau

```
gulp.task('default', ['task1', 'task2', 'task3',...], function() {
});
```

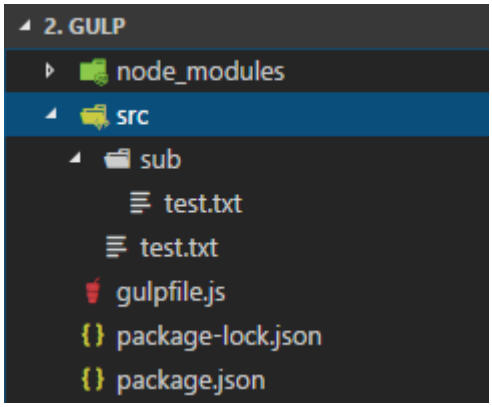
Với task1, task2, task3... là những task đã được khai báo trước và những task này được truyền vào task default thông qua đối số là một mảng.

2.4 Chạy chương trình GULP

- Thêm vào nội dung sau vào file gulpfile.js

```
var gulp = require('gulp');
gulp.task('default', function() {
    gulp.watch('src/**/*', ['default']);
    return gulp.src('src/**/*')
        .pipe(gulp.dest('dest'))
});
```

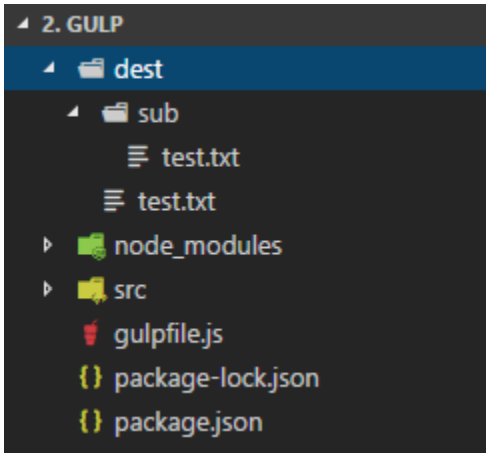
- Trong thu mục chứa project tạo thư mục src, trong src/ tiếp tục tạo một tệp tin bất kì và thư mục sub, trong src/sub/ cũng tạo một tệp tin bất kì



- Chạy lệnh gulp trong console ta sẽ được kết quả như nhau

```
PS C:\Users\nguye\Documents\CNM\2. GULP> gulp
[10:05:48] Using gulpfile
[10:05:48] Starting 'default'...
[10:05:49] Finished 'default' after
```

Bây giờ trong thư mục chứa project đã có thêm thư mục dest/ chứa nội dung như src/.



Bất cứ thay đổi nào trong src/ đều được sao chép sang dest/.

2.5 Package dependencies của GULP

- Khái niệm này nói về các package hay còn gọi là module được sử dụng trong project.
- Các package dependencies sử dụng trong project được khai báo trong thuộc tính dependencies của file "package.json".
- Một số dependencies thông dụng được giới thiệu trong bài viết này như:

  gulp tự động hóa task.

  gulp-sass biên dịch file scss thành css.

  gulp-file-include module hóa html thành templates.

  browser-sync tạo localhost và tự động reload trình duyệt.

  run-sequence chạy các task gulp đồng bộ.

  Những module có tiền tố gulp- là những module phụ thuộc vào gulp, chứa các tính năng phục vụ cho gulp task, chỉ sử dụng được khi module gulp đã được đưa vào ứng dụng.

3. Tiền xử lí CSS với SASS

3.1 Giới thiệu SASS và GULP-SASS

- CSS Preprocessor (tiền xử lí css) là một ngôn ngữ kịch bản mở rộng của css và được biên dịch thành cú pháp css giúp viết css nhanh hơn, có cấu trúc rõ ràng hơn. CSS Preprocessor giúp tiết kiệm thời gian viết css, dễ dàng bảo trì và phát triển css.
- SASS là một CSS Preprocessor cung cấp thêm các quy tắc như nested rule, variable, mixin, ... Với sass ta có thể viết css theo thứ tự rõ ràng, quản lý các biến đã được định nghĩa sẵn, có thể tự động nén tập tin css.
- GULP-SASS là một module phụ thuộc của gulp, giúp biên dịch sass thành css.
- Vì phần đuôi mở rộng của sass là scss nên đôi khi ta gọi scss thay cho sass.

3.2 Cài đặt GULP-SASS

- Cài đặt gulp-sass bằng lệnh sau (phải cấu hình package.json trước):

  npm install --global gulp-sass

  npm install --save gulp-sass

  Có thể cài gulp-sass ở cục bộ (trong project) vì việc cài đặt gulp-sass ở phạm vi toàn cục là không cần thiết.

3.3 Cấu hình task GULP-SASS trong gulpfile.js

- Thêm nội dung vào file gulpfile.js như sau

```
var gulp = require('gulp');
var sass = require('gulp-sass');
gulp.task('default', ['sass'], function() {
    gulp.watch('app/scss/*.scss', ['sass']);
});
gulp.task('sass', function() {
    return gulp.src('app/scss/*.scss')
        .pipe(sass())
        .pipe(gulp.dest('app/css'));
});
```

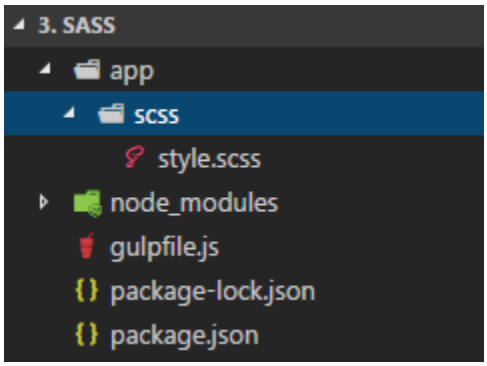
- Task sass trên sẽ đọc tất cả các file .scss trong thư mục app/scss/ của project rồi biên dịch thành file .css lưu vào thư mục app/css/ (cùng trong project).

  * là ký tự đại diện có nghĩa phù hợp với mọi pattern trong thư mục hiện tại, trong trường hợp này là tất cả file kết thúc với .scss trong thư mục app/scss/.

- gulp.watch theo dõi khi một file được lưu sẽ tự động chạy lại task sass.

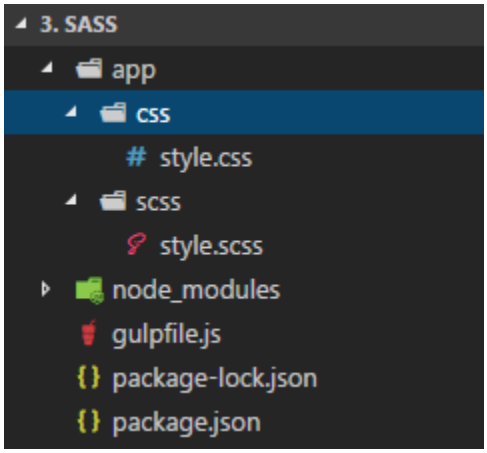
3.4 Biên dịch SCSS thành CSS với GULP-SASS

- Tạo file style.scss trong thư mục app/scss/ của project và thêm vào đoạn mã scss sau




```
ul {
    color: red;
    li {
        color: blue;
    }
}
```

- Chạy lệnh gulp để thực hiện biên dịch, sau khi hoàn tất thì một file style.css sinh ra trong thư mục app/css/ (cùng project) có nội dung như sau




```
ul {
  color: red; }
  ul li {
    color: blue; }
```

- Mỗi lần file style.scss được lưu thì task sass sẽ tự động chạy lại để biên dịch scss thành css.

4. Module hóa HTML với GULP-FILE-INCLUDE

4.1 Giới thiệu HTML TEMPLATES và GULP-FILE-INCLUDE

- Giao diện website sẽ có những phần nội dung giống và khác nhau. Module hóa html là việc chia nhỏ những phần nội dung này để quản lí và tái sử dụng. Những phần chia nhỏ này thường được gọi là template.

  Ví dụ nội dung phần header, footer của một website là phần chung và được tái sử dụng trong nhiều page khác nhau của website.
- GULP-FILE-INCLUDE là một module phụ thuộc của gulp giúp chia nhỏ, quản lý, tái sử dụng html.

4.2 Cài đặt GULP-FILE-INCLUDE

- Cài đặt gulp-file-include bằng lệnh sau (phải cấu hình package.json trước):

  npm install --global gulp-file-include

  npm install --save gulp-file-include

  Có thể cài gulp-file-include ở cục bộ (trong project) vì việc cài đặt gulp-file-include ở phạm vi toàn cục là không cần thiết.

4.3 Cấu hình GULP-FILE-INCLUDE trong gulpfile.js

- Thêm nội dung vào file gulpfile.js như sau

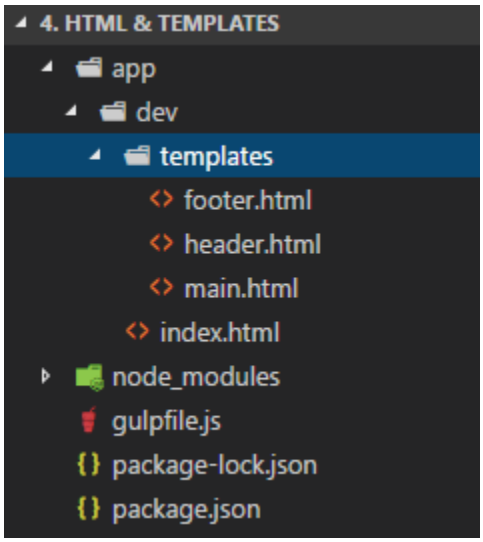
```
var gulp = require('gulp');
var fileinclude = require('gulp-file-include');
gulp.task('default', ['fileinclude'], function() {
    gulp.watch('app/dev/**/*', ['fileinclude']);
});
gulp.task('fileinclude', function() {
    return gulp.src('app/dev/*html')
        .pipe(fileinclude({
            prefix: '@@',
            basepath: '@file'
        }))
        .pipe(gulp.dest('app'));
});
```

- Task fileinclude trên sẽ đọc nội dung trong tất cả các file .html trong thư mục app/dev/ của project tiến hành xử lí rồi biên dịch thành file .html hoàn chỉnh lưu vào thư mục app/ (cùng trong project).

  * là ký tự đại diện có nghĩa phù hợp với mọi pattern trong thư mục hiện tại, trong trường hợp này là tất cả file kết thúc với .html trong thư mục app/dev/.
- gulp.watch theo dõi khi một file được lưu sẽ tự động chạy lại task fileinclude.

4.4 Nối TEMPLATES thành trang HTML hoàn chỉnh

- Trong thư mục app/dev/templates/ (của project) tạo các tệp tin với nội dung sau



header.html

```
<!-- HEADER -->
<header>
    <h1>THIS IS HEADER</h1>
</header>
<!-- END HEADER -->
```

main.html

```
<!-- MAIN -->
<main>
    <h1>THIS IS MAIN</h1>
</main>
<!-- END MAIN -->
```

footer.html

```
<!-- FOOTER -->
<footer>
    <h1>THIS IS FOOTER</h1>
</footer>
<!-- END FOOTER -->
```

- Trong thư mục app/dev/ (của project) tạo tệp tin với nội dung sau

  Index.html

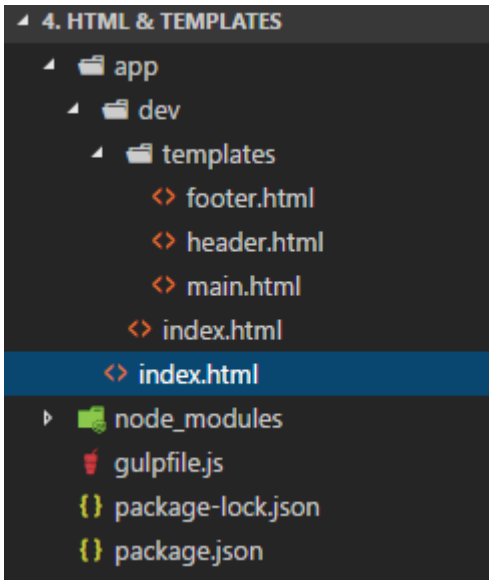
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>GULP</title>
</head>

<body>
    @@include("templates/header.html")
    <!-- // -->
    @@include("templates/main.html")
    <!-- // -->
    @@include("templates/footer.html")
</body>

</html>
```

- Chạy lệnh gulp để thực hiện biên dịch, sau khi hoàn tất thì một file index.html sinh ra trong thư mục app/ (cùng project) có nội dung như sau



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>GULP</title>
</head>

<body>
    <!-- HEADER -->
    <header>
        <h1>THIS IS HEADER</h1>
    </header>
    <!-- END HEADER -->
    <!-- // -->
    <!-- MAIN -->
    <main>
        <h1>THIS IS MAIN</h1>
    </main>
    <!-- END MAIN -->
    <!-- // -->
    <!-- FOOTER -->
    <footer>
        <h1>THIS IS FOOTER</h1>
    </footer>
    <!-- END FOOTER -->
</body>

</html>
```

- Mỗi lần các file .html được lưu thì task fileinclude sẽ tự động chạy lại để nối các template html và biên dịch thành trang html hoàn chỉnh app/index.html.

SCSS có thể được chia nhỏ để quản lí giống như module hóa HTML.

5. Live-reloading với BROWSER-SYNC

5.1 Giới thiệu BROWSER-SYNC

- Browser Sync giúp việc phát triển web dễ dàng hơn bằng cách tạo ra một web server.
- Live-reloading (tự động refresh) cửa sổ trình duyệt trỏ đến web server (tạo ra trước đó) khi sửa đổi các file javascript, css, html,... trong project.
- Đồng bộ tất cả các của sổ trình duyệt cùng trỏ đến web server khi scroll, form,...
- BROWSER-SYNC là một mudule độc lập của NodeJS, không phụ thuộc vào gulp nhưng có thể được sử dụng tích hợp với gulp.

5.2 Cài đặt BROWSER-SYNC

- Cài đặt gulp-file-include bằng lệnh sau (phải cấu hình package.json trước):

  npm install --global browser-sync

  npm install --save browser-sync

  Có thể cài browser-sync ở cục bộ (trong project) vì việc cài đặt browser-sync ở phạm vi toàn cục là không cần thiết.

5.3 Cấu hình BROWSER-SYNC trong gulpfile.js

- Tạo một web server với cấu hình như sau trong gulpfile.js

```
var gulp = require('gulp');
var browserSync = require('browser-sync').create();
gulp.task('browserSync', function() {
    browserSync.init({
        server: {
            baseDir: 'app'
        },
        port: 80
    });
});
```

  Phương thức create() khởi tạo một server.

  Phương thúc init() truyền vào một đối tượng chứa các thuộc tính có các giá trị cần thiết để cấu hình cho server.

  Thuộc tính baseDir chứa giá trị mà một chuỗi nhằm xác định thư mục root của web server.

  Thuộc tính port chứa giá trị làm một số nhằm xác định web server lắng nghe cổng nào.

- Lệnh gulp browserSync sẽ chạy task browserSync và khởi tạo một web server từ thư mục app (được xem là thư mục root của web server) của project, web server này lắng nghe trên cổng 80.

  Một cửa sổ trình duyệt sẽ tự động mở lên với url: http://localhost/ và trỏ đến app/index.html.

5.4 Tạo một web server live-reloading kết nối với project

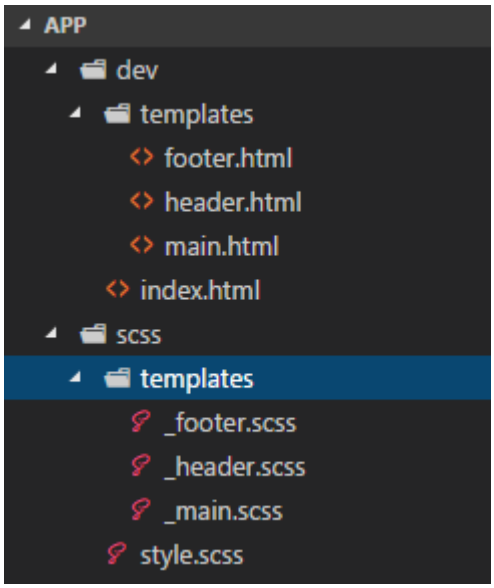
- Phần này sẽ kết hợp với gulp-sass cùng gulp-file-include để tạo một project hoàn chỉnh.
- Cấu hình gulpfile.js như sau

```
var gulp = require('gulp');
var sass = require('gulp-sass');
var fileinclude = require('gulp-file-include');
var browserSync = require('browser-sync').create();
gulp.task('sass', function() {
    return gulp.src('app/scss/*.scss')
        .pipe(sass())
        .pipe(gulp.dest('app/css'));
});
gulp.task('fileinclude', function() {
    return gulp.src('app/dev/*html')
        .pipe(fileinclude({
            prefix: '@@',
            basepath: '@file'
        }))
        .pipe(gulp.dest('app'));
});
gulp.task('watch', function() {
    gulp.watch('app/scss/**/*', ['sass']);
    gulp.watch('app/dev/**/*', ['fileinclude']);
    gulp.watch('app/css/**/*', browserSync.reload);
    gulp.watch('app/*html', browserSync.reload);
});
gulp.task('browserSync', function() {
    browserSync.init({
        server: {
            baseDir: 'app'
        },
        port: 80
    });
});
gulp.task('default', ['sass', 'fileinclude', 'browserSync',
'watch'], function() {});
```

Phần này gulp.watch đã được tách khỏi task default trở thành task watch độc lập.

browserSync.reload sử dụng phương thức reload để refresh lại của sổ trình duyệt kết nối đến web server khi các tệp tin thư mục được gulp.watch theo dõi có sự thay đổi.

- Tạo thư mục app/ trong thư mục chứa project có cấu trúc như sau



Tương tự như cấu trúc đã tạo ở 3 & 4 nhưng phần này cấu trúc quản lý scss sẽ có chút thay đổi. Stylesheet cho trang html sẽ được tách thành từng phần nhỏ (tương ứng với các template html) và include và một file chính là style.scss.

Để thực hiện include thì những file scss thành phần phải được đặt tên theo quy định như trên, phải có tiền tố _ trước tên gọi.

- Phần html nội dung như ở 4, không thay đổi nhiều chỉ có file app/dev/index.html được thêm link stylesheet, link này sẽ trỏ tới file app/css/style.sss (là kết quả sau khi biên dịch scss)

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>GULP</title>
    <link rel="stylesheet" href="css/style.css">
</head>

<body>
    @@include("templates/header.html")
    <!-- // -->
    @@include("templates/main.html")
    <!-- // -->
    @@include("templates/footer.html")
</body>

</html>
```

- Phần scss với nội dung sau

style.scss

```
@import 'templates/header.scss';
@import 'templates/main.scss';
@import 'templates/footer.scss';
```

_header.scss

```
header {
    background: red;
    h1 {
        padding: 72px 0;
        font-size: 36px;
        text-align: center;
        color: white;
    }
}
```

_main.scss

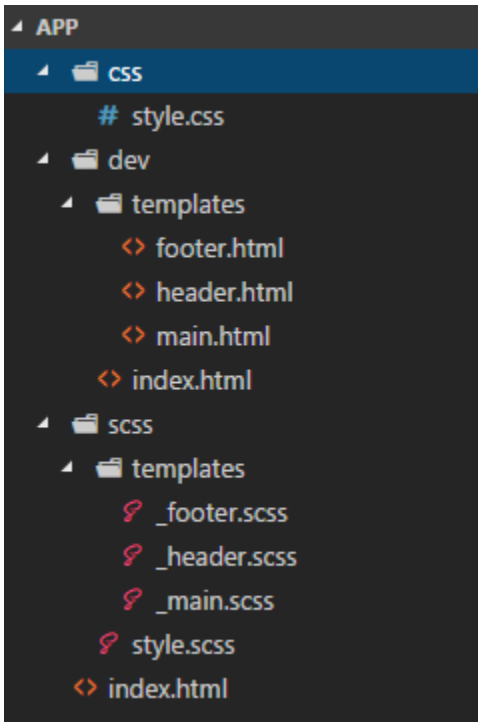
```
main {
    background: green;
    h1 {
        padding: 72px 0;
        font-size: 36px;
        text-align: center;
        color: white;
    }
}
```

_footer.scss

```
footer {
    background: blue;
    h1 {
        padding: 72px 0;
        font-size: 36px;
        text-align: center;
        color: white;
    }
}
```

- Chạy lệnh gulp, một cửa sổ trình duyệt sẽ tự động bật lên có nội dung như sau

- Cấu trúc thư mục sau khi chạy lệnh gulp



Bây giờ mỗi lần các tập tin trong thư mục dev và scss được lưu thì cửa sổ trình duyệt đã mở ra trước đó sẽ tự động refresh. Điều này gọi là web server live-reloading.

6. Xử lí các task đồng bộ (synchronous) với RUN-SEQUENCE

6.1 Giới thiệu RUN-SEQUENCE

- JAVASCRIPT là một ngôn ngữ xử lí bất đồng bộ (async) nên các module của NodeJS cũng là async. Điều này là một lợi thế của javascript nhưng cũng gây khá nhiều rắc rối.
- Mỗi task trong gulp cũng là async nên kịch bản gulp đã tạo sẽ chạy không chính xác trong một số trường hợp.

  Giả sử chương trình gulp có 2 task là taskA & taskB, taskB cần kết quả trả về từ taskA thì mới hoạt động chính xác được.

  Nếu xử lý đồng bộ (sync) thì taskA chạy xong, taskB lấy kết quả taskA rồi chạy tiếp thì sẽ không có vấn đề gì.

  Nhưng nếu xử lí bất đồng bộ (async) thì taskB sẽ có thể được khởi chạy khi taskA chưa hoàn thành, điều này sẽ khiến chương trình gulp phát sinh sai sót không kiểm soát được.
- BROWSER-SYNC là một mudule độc lập của NodeJS, không phụ thuộc vào gulp nhưng có thể được sử dụng tích hợp với gulp, hỗ trợ chương trình gulp xử lí đồng bộ, giúp các task của gulp chạy lần lượt, task trước xử lí xong rồi mới đến task sau.

6.2 Cài đặt RUN-SEQUENCE

- Cài đặt run-sequence bằng lệnh sau (phải cấu hình package.json trước):

  npm install --global run-sequence

  npm install --save run-sequence

  Có thể cài run-sequence ở cục bộ (trong project) vì việc cài đặt run-sequence ở phạm vi toàn cục là không cần thiết.

6.3 Cấu hình RUN-SEQUENCE trong gulpfile.js

- Cấu hình như sau trong gulpfile.js

```
var gulp = require('gulp');
var runSequence = require('run-sequence');
gulp.task('default', function() {
    runSequence('sass', 'fileinclude', 'browserSync', 'watch');
});
```

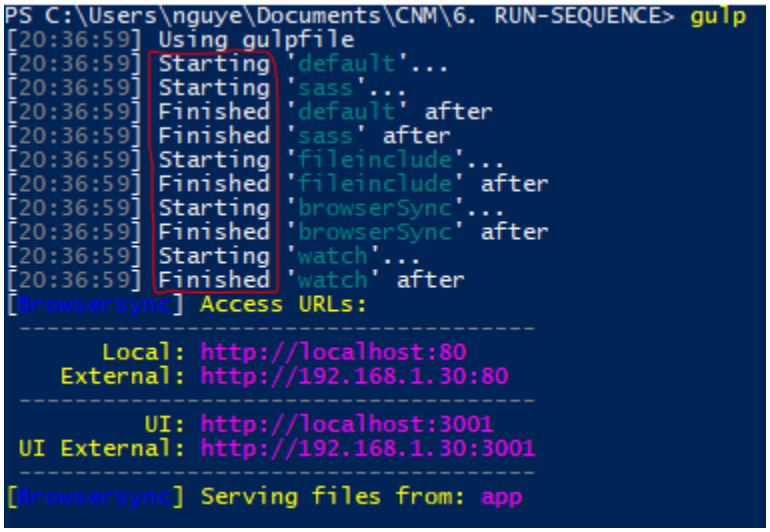
  Các task sass, fileinclude, browserSync, watch,... là những task đã được khai báo. Những task này chạy lần lượt từ task đầu cho đến task cuối, task trước xử lí xong thì task sau mới chạy.

6.4 Chạy task đồng bộ với RUN-SEQUENCE

- Sử dụng lại dữ liệu đã dùng ở 5, cài đặt thêm module run-sequence và cấu hình lại gulpfile.js

```
var gulp = require('gulp');
var runSequence = require('run-sequence');
var sass = require('gulp-sass');
var fileinclude = require('gulp-file-include');
var browserSync = require('browser-sync').create();
gulp.task('sass', function() {
    return gulp.src('app/scss/*.scss')
        .pipe(sass())
        .pipe(gulp.dest('app/css'));
});
gulp.task('fileinclude', function() {
    return gulp.src('app/dev/*html')
        .pipe(fileinclude({
            prefix: '@@',
            basepath: '@file'
        }))
        .pipe(gulp.dest('app'));
});
gulp.task('watch', function() {
    gulp.watch('app/scss/**/*', ['sass']);
    gulp.watch('app/dev/**/*', ['fileinclude']);
    gulp.watch('app/css/**/*', browserSync.reload);
    gulp.watch('app/*html', browserSync.reload);
});
gulp.task('browserSync', function() {
    browserSync.init({
        server: {
            baseDir: 'app'
        },
        port: 80
    });
});
gulp.task('default', function() {
    runSequence('sass', 'fileinclude', 'browserSync', 'watch');
});
```

- Chạy lệnh gulp trong console sẽ được kết quả như sau

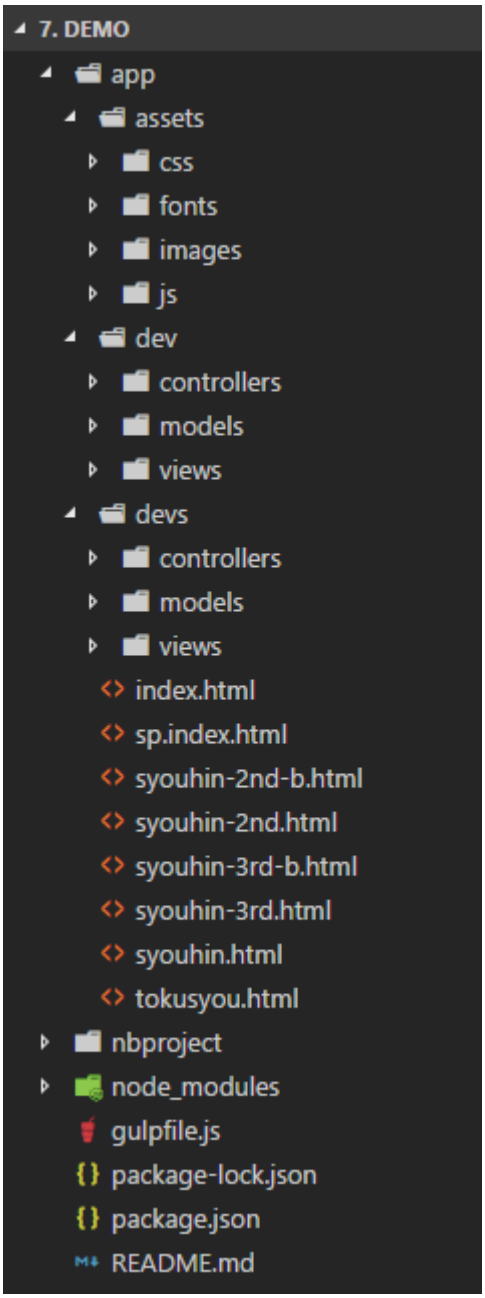


Để ý phần khoanh đỏ, đây là thông báo tiến trình xử lí của các task từ lúc bắt đầu chạy đến khi kết thúc.

Dễ thấy rằng các task sass, fileinclude, browserSync, watch chạy dúng theo thứ tự đã thiết lập trong kịch bản gulpfile.js.

7. DEMO PROJECT FRONT-END

7.1 Giới thiệu cấu trúc của một dự án FRONT-END thực tế



7.2 Mô tả cấu trúc

PROJECT STRUCTURE

- Thư mục phát triển, các page html sau khi render sẽ được xuất ra tại đây
    * app/
- Thư mục chứa tài nguyên tĩnh: css, js, images, fonts
    * app/assets/
- Thư mục chứa source phát triển giao diện cho màn hình lớn
    * app/dev/
- Thư mục chứa layout cho các trang, layout sẽ include các link css, js và các template html vào để xuất ra các page thành phẩm:
    * app/dev/controllers/
- Thư mục chứa file scss cho project, các file scss ở đây sẽ được xuất sang app/assets/ bằng gulp
    * app/dev/models/
- Thư mục chứa template sử dụng cho layout
    * app/dev/views/

FONTS

- Các fonts sử đụng được khai báo ở đây
    * app/dev/models/models/_typography.scss

SCSS & HTML

- SCSS dùng cú pháp BEM style 1 cấp
- Các biến được khai báo trong file
    * app/dev/models/models/variables.scss
- Các style dùng chung được viết trong file
    * app/dev/models/models/_general.scss
- Các scss trong
    * app/dev/models/controllers/
        * tương ứng với html trong
    * app/dev/models/
- Các scss trong
    * app/dev/models/views/
        * tương ứng với html trong
    * app/dev/views/

* Một page được chia thành nhiều section độc lập để code riêng rồi dùng layout include các phần đã viết vào (scss cũng tương tự như vậy) để dễ bảo trì quản lí code.

NOTE

- Thư mục chứa source phát triển giao diện cho màn hình nhỏ, thiết bị smartphone
    * app/devs/
    * Thư mục này sẽ có cấu trúc và quản lí tương tự app/dev/
- Tất cả các page cho một loại màn hình (PC or SP) sẽ sử dụng chung file css & js
- Source của PC & SP sẽ độc nhau, sử dụng css & js riêng

8. Cơ bản về GIT

8.1 Giới thiệu GIT

- Git là một trong các phần mềm (hệ thống) quản lý mã nguồn (source code) phổ biến nhất.
- Git hoạt động theo cơ chế "quản lý phiên bản phân tán" (Distributed Version Control - DVC):

  Không cần có chung một nơi để lưu trữ mã nguồn (source).

  Mỗi thành viên sẽ có một kho chứa (repository) ở máy tính (local) của họ để lưu trữ.

  Khi quản lý file bằng Git, lịch sử cập nhật (sự thay đổi của source như thêm, xóa, sửa) sẽ được lưu trong Git, vì vậy có thể đưa trạng thái source về một thời điểm nào đó trong lịch sử.
- Git có khả năng chạy trên nhiều hệ điều hành khác nhau như Linux, Windows, Mac OSX.
- Có thể sử dụng git với dòng lệnh (console) hoặc với giao diện người dùng (UI).

8.2 Cài đặt GIT trên Windows

- Tải git và cài đặt như một phần mềm bình thường tại đây https://git-scm.com/.

8.3 Repository (kho chứa source code)

*Tổng quát:

- Repository (nhà kho) hay được gọi tắt là repo đơn giản là nơi chứa/cơ sở dữ liệu (database) tất cả những thông tin cần thiết để duy trì và quản lý các sửa đổi lịch sử của dự án.
- Bằng việc đặt thư mục muốn quản lý lịch sử thay đổi dưới sự quản lý của repository (git), có thể ghi chép lại lịch sử thay đổi của thư mục và file trong thư mục đó.

*Phân loại

- Local repository: là repo bố trí trên máy của cá nhân, dành cho một người dùng sử dụng.
- Remote repository: Là repo để chia sẻ giữa nhiều người và bố trí trên server chuyên dụng.

*Tạo remote repository:

- Tạo với dịch vụ của GitHub hoặc GitLap...

*Tạo local repository:

- Cách 1: tạo repository hoàn toàn mới (cách này ta ít dùng nên sẽ không đề cập ở đây).
- Cách 2: sao chép (clone) từ remote repository của dịch vụ của GitHub hoặc GitLap...

8.4 Commit trong GIT

- Để ghi lại việc thêm/ thay đổi file vào repository thì sẽ thực hiện thao tác gọi là commit, nghĩa là sẽ đánh dấu mốc lịch sử và lưu lại trạng thái của file (source code) tại thời điểm commit.
- Điều kiện tiên quyết để commit là tập tin phải được theo dõi.

8.5 Working Tree trong GIT

- Cây làm việc, không gian chứa source code để làm việc trực tiếp với source code ở đấy.
- Với một thư mục (folder) được đặt dưới sự quản lí của git thì các dữ liệu sẽ được lưu trữ ở hai nơi:

  Một là thư mục đang làm việc trên máy tính (working tree), source ở đây có thể thấy được và làm việc trực tiếp.

  Hai là kho chứa mã nguồn (repository), source code ở đây bị ẩn đi và không thể làm việc trực tiếp.

8.6 Quy trình làm việc với các dịch vụ GIT (với GitHub hay GitLap đều như nhau)

- Cấu hình Git trên local:

  git config --global user.name "nguyenhoangsin"

  git config --global user.email "nguyenhoangsin@gmail.com"

  user.name và user.email tùy ý nhưng nên trùng với thông tin đã đăng trên các dịch vụ git.
- Xem lại thiết lập:

  cat ~/.gitconfig

  cat .git/HEAD

  git config –list
- Clone remote repository: sao chép mã nguồn trên server

  git clone https://github.com/nguyenhoanhsin/github-front-end.git

  Phần link phía sau sẽ do dich vụ git đang sử dụng cung cấp khi tạo remote repo.
- Nếu đã clone rồi thì cập nhật mã nguôn mới nhất từ server (chú ý băt đầu từ đây vị trí của console or cmd phải trỏ ở thư mục clone về)

  git pull origin master

  Bản chất của quá trình này là đồng bộ bản ghi lich sử và source mới nhất từ server về local.
- Đưa tập tin đã thêm, sửa hoặc xóa vào theo dõi (tracked) để commit

  git add .

  Lệnh nãy sẽ dưa tất cả tệp tin có thay đổi vào theo dỏi.
- Commit: tạo mốc ghi lịch sử cho trạng thái tập tin, thư mục

  git commit -m "ở đây sẽ là mô tả nội dung chú thích về sự thay đổi tập tin"
- Đẩy tập tin lên server git (GitHub or GitLap)

  git push -u origin master

  Bản chất của quá trình này là đồng bộ bản ghi lich sử và source mới nhất từ local lên server.

- Kiểm tra lại lịch sử commit bằng lệnh

  git log

- Kiểm tra trạng thái của kho chứa

  git status

8.7 Sơ đồ quy trình làm việc với các dịch vụ GIT