Gulp

Gestión de assets en Symfony

PHPSevilla

28 Septiembre 2016

Juan Luis García Borrego

PHP Sevilla
Developers group of PHP technologies

Patrocinadores

Sobre mí

Juan Luis García Borrego

Desarrollador Symfony en Innn

@JuanluGarciaB

juanluisgarciaborrego.com

ÍNDICE

Flujo de trabajo

NPM

Gulp

Assets en Symfony

Ejemplo práctico

Flujo de trabajo

NPM

Gulp

Assets en Symfony

Ejemplo práctico

Flujo de trabajo

NPM

Gulp

Assets en Symfony

Ejemplo práctico

Flujo de trabajo

NPM

Gulp

Assets en Symfony

Ejemplo práctico

Flujo de trabajo

NPM

Gulp

Assets en Symfony

Ejemplo práctico

Flujo de trabajo

Flujo de trabajo

El flujo de trabajo de un desarrollador se ha convertido en una tarea un poco “*compleja*”.

Para una aplicación sencilla como mínimo en el frontend usamos algún framework css (Bootstrap, Materialize, PureCss, Foundation, otros.), un preprocesador css (Sass, LEES, otros), alguna librería JavaScript (jQuery, Angular, React, Vue, *infinitas*)

Además, necesito gestionar las versiones, unir todos los css/js en un archivo y minificarlo para que mi web cargue más rápido y lo que surja….

¿MANUALMENTE?

Flujo de trabajo

Al igual que Composer nos facilita la vida gestionando nuestros vendors

Existen numerosas herramientas y automatizadores de tareas FrontEnd que nos facilitan nuestro día a día

Grunt, Bower, Gulp, WebPack

NPM

NPM

Introducción

NPM

Node Package Manager

Gestor de paquetes FrontEnd y BackEnd [NodeJS]

Busca, gestiona y reutiliza librerías de otros programadores

Viene incluido al instalar Node JS

NPM

Node Package Manager

Gestor de paquetes FrontEnd y BackEnd [NodeJS]

Busca, gestiona y reutiliza librerías de otros programadores

Viene incluido al instalar Node JS

Misma funcionalidad que Composer en PHP

FRONTEND

BACKEND

NPM

package.json

NPM

package.json

En este archivo queda reflejado toda la configuración de un “proyecto node” (nombre, autor, licencia, versión, dependencias, etc.)

En nuestro caso informa de las dependencias node utilizadas.

El archivo package.json estará alojado en el directorio raíz del proyecto

FRONTEND

package.json

BACKEND

composer.json

NPM

Guía rápida de uso

NPM

Inicializar proyecto

$ npm init

Nos pedirá datos de nuestro proyecto y generará el archivo package.json

```
{
  “name”:”ejemplo“,
  "version": "1.0.0",
  "description": "Descripción del proyecto",
  "dependencies": {
  },
  "devDependencies": {
  },
  "keywords": [
    "phpsevilla",
    "gulp"
  ],
  "author": "Juan Luis García Borrego",
  "license": "MIT"
}
```

package.json

NPM

Añadir dependencias

$ npm install —save-dev *NOMBRE*

Instala el paquete *NOMBRE (guarda los archivos en el directorio node_modules)* y añade la librería al archivo package.json en el apartado de dependencias de desarrollo.

```
"devDependencies": {
  },
```

NPM

Eliminar dependencias

$ npm remove —save-dev *NOMBRE*

Elimina el paquete *NOMBRE (node_modules/)* y borra la librería al archivo package.json en el apartado de dependencias de desarrollo.

```
"devDependencies": {
  },
```

NPM

Chequea si existe nuevas versiones

$ npm outdated

```
Package  Current  Wanted  Latest  Location
gulp       2.7.0   2.7.0   3.9.1  php-sevilla-gulp
```

Para instalar la nueva versión:

$npm install gulp@3.9.1 —save-dev

Gulp

Gulp

Introducción

Gulp

Introducción

Una herramienta que nos permite automatizar tareas comunes de desarrollo a través del terminal y gracias su gran cantidad de plugins nos permiten realizar multitud de tareas como:

- Unir y mitificar archivos CSS / JavaScript
- Procesa los archivos CSS, que utilicen preprocesadores como Sass, Less u otros.
- Reduce el tamaño de las imágenes
- **http://gulpjs.com/plugins/**

Gulp

Instalación

Gulp

Instalación

Instalación global

$ sudo npm install —global gulp

Instalación como dependencia

$ npm install —save-dev gulp

Gulp

gulpfile.js

Gulp

gulpfile.js

En este archivo vamos a indicar cada una de las **task** a realizar.

El archivo **gulpfile.js** se tiene que crear en el **directorio raíz** del proyecto.

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

gulp.task('default', function(){
    console.log("Hola PHP Sevilla");
});
```

gulpfile.js

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

gulp.task('default', function(){
    console.log("Hola PHP Sevilla");
});
```

```
➜  phpsevilla gulp
[19:23:05] Using gulpfile ~/Sites/lab/phpsevilla/gulpfile.js
[19:23:05] Starting 'default'...
Hola PHP Sevilla
[19:23:05] Finished 'default' after 92 µs
➜  phpsevilla
```

gulpfile.js

Gulp

Sintaxis

Gulp

var gulp = require(‘gulp’);

En el archivo de configuración gulpfile.js tenemos que definir todos los plugins utilizados, incluido gulp.

var NombrePlugin1 **= require(**‘nombrePlugin1’**);**
var NombrePluginN **= require(**‘nombrePluginN’**);**

Gulp

gulp.task()

Las tareas se definen llamando al método .task().

Como primer parámetro se le pasa el nombre de la tarea, y como se fundo se define una función anónima que contendrá la lógica del mismo.

```
gulp.task(‘nombreTarea’, function() {
    //tarea
});
```

Gulp

gulp.task()

Por defecto, gulp tiene una tarea definida llama *default*.

Al ejecutar en la terminal el comando **gulp** es la tarea que se ejecuta.

gulp.task('default', ['styles', 'fonts', 'images', ‘scripts’]);
Al ejecutar el comando gulp: Se ejecuta la tarea default, que llama a las tareas incluidas en el array

Gulp

gulp.src()

El método src() es el encargado de especificar la localización de los ficheros/assets existentes.

gulp.src(‘assets/js/*.js’);
Todos los archivos con extensión .js dentro del directorio assets/js

Gulp

gulp.src()

El método src() es el encargado de especificar la localización de los ficheros/assets existentes.

gulp.src(‘assets/js//*.js’);**

Todos los archivos con extensión .js que estén dentro de assets/js/ o alguna carpeta que pueda contener js/. (Búsqueda recursiva por directorios)

Gulp

gulp.dest()

El método gulp.dest(), indica el directorio de **dest**ino.

gulp.dest(‘web/js’**)**

Gulp

.pipe()

El método pipe, lo inicializa el método src(), y actúa como un tubo o canal, en el que se le van a uniendo funciones/plugins a desempeñar.

```
gulp.src(‘files’)
    .pipe(‘acción1’)
    .pipe(‘acciónN’)
    .pipe(gulp.dest(‘web/js’))
```

Gulp

gulp.watch()

Proporciona la utilidad de comprobar automáticamente cambios en un recurso especificado.

No tenemos que estar continuamente escribiendo el comando gulp para procesar cambios.
Abrimos un segundo proceso en la consola con: **gulp watch**

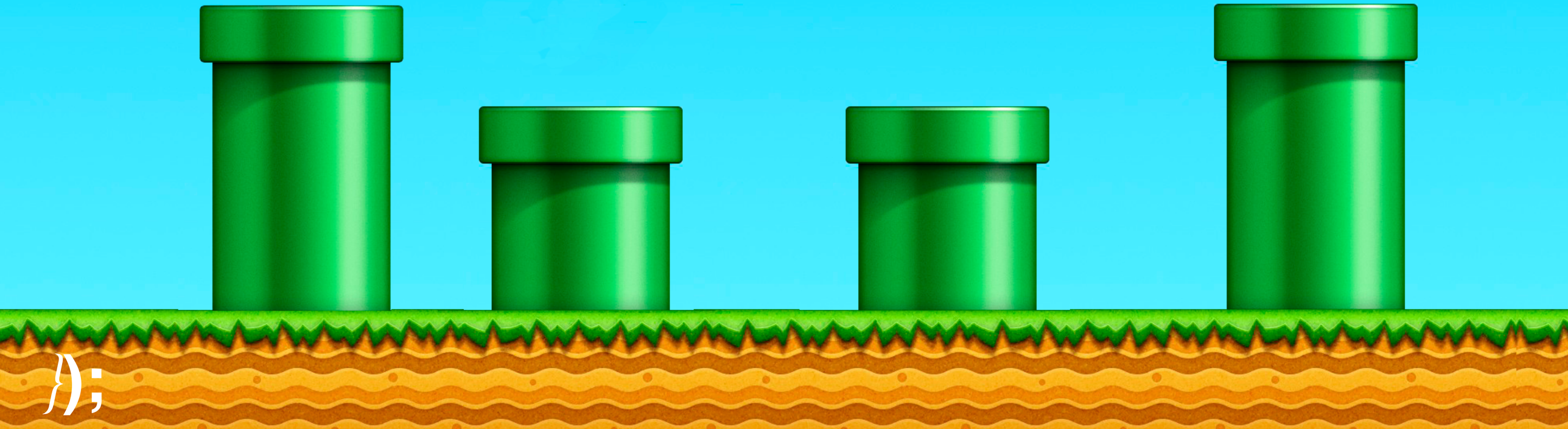
```
gulp.task(‘watch', function(){
    return gulp.watch(‘ficheros’, [‘styles’, ‘js’])
});
```

Gulp

Sintaxis

- Ejemplo gráfico de task()

gulp.task(‘*mario*’, *function( ) {*



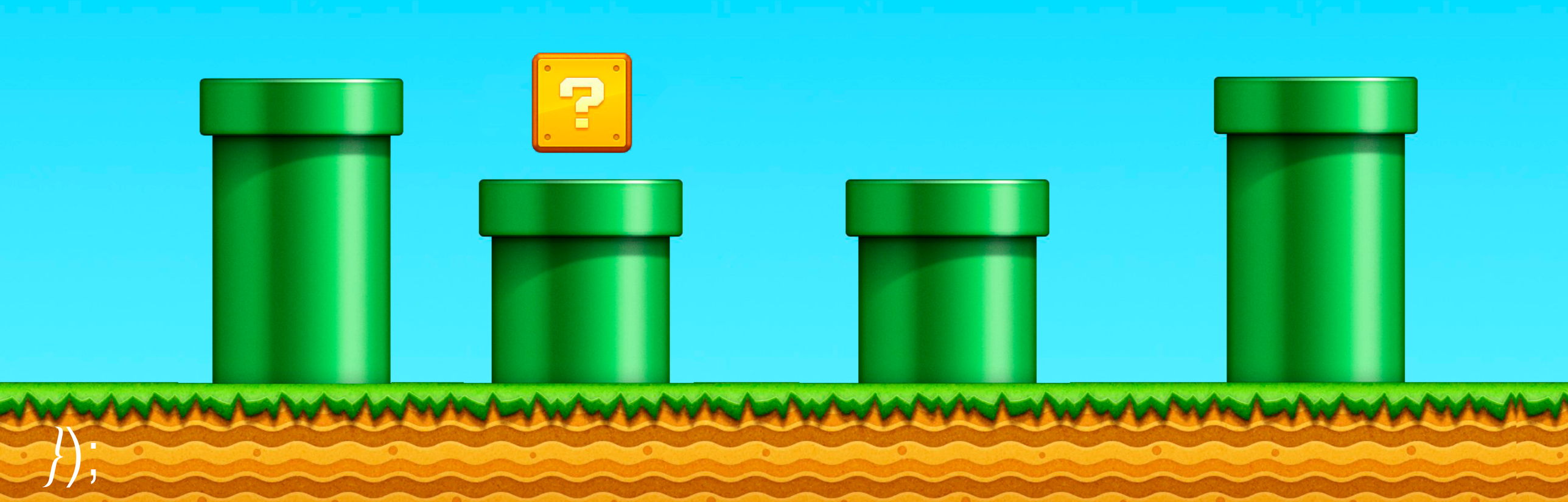
});

gulp.task(‘*mario*’, *function( ) {*

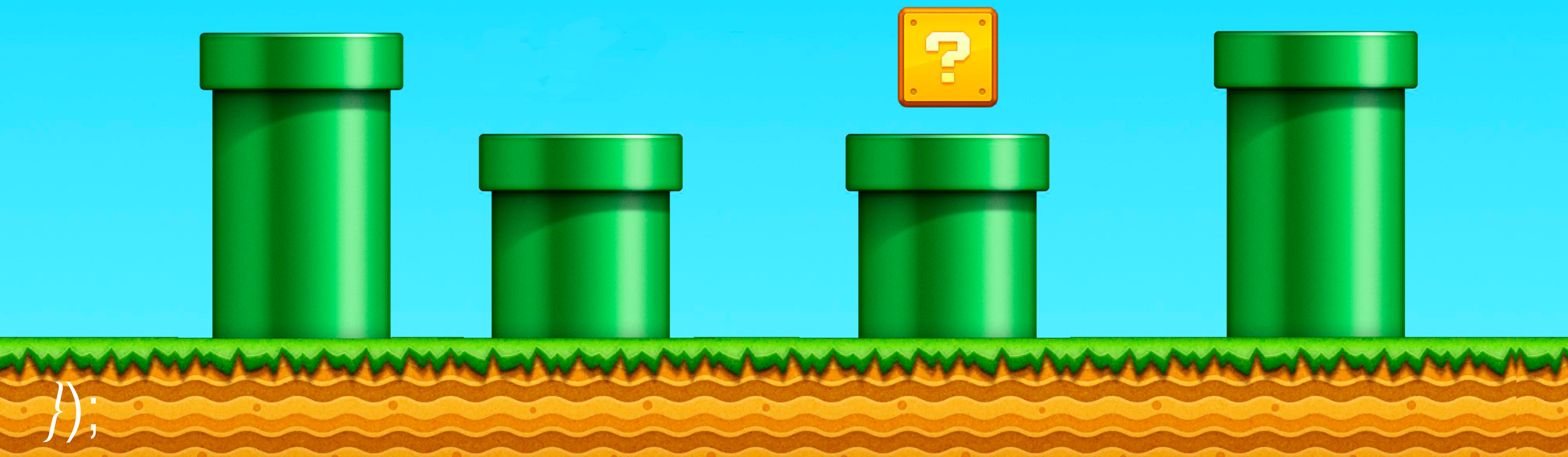
gulp.src(*‘asset/*.scss’****)***

});

```
gulp.task('mario', function( ) {
        gulp.src('asset/*.scss')
            .pipe('pluginProcesarSass'())
});
```



```
gulp.task('mario', function( ) {
        gulp.src('asset/*.scss')
            .pipe('pluginProcesarSass'())
            .pipe('pluginMinificar'())
});
```



```
gulp.task('mario', function( ) {
        gulp.src('asset/*.scss')
            .pipe('pluginProcesarSass'())
            .pipe('pluginMinificar'())
            .pipe(gulp.dest('web/css'));
});
```

Gulp

Sintaxis

- Ejemplo procesando archivos sass y minificado del archivo css

```
var gulp = require('gulp');
var sass = require('gulp-sass');
var cleanCSS = require('gulp-clean-css');

gulp.task('styles', function () {
    gulp.src('app/assets/sass/*.scss')
        .pipe(sass())
        .pipe(cleanCSS())
        .pipe(gulp.dest('web/css'));
});

gulp.task('default', ['styles']);
```

gulpfile.js

Gulp

Plugins básicos

Gulp

Plugins básicos

gulp-util: Permite añadir un poco de lógica a gulp, incluir mensajes logs mejorados con colores etc.

https://github.com/gulpjs/gulp-util

npm install —save-dev gulp-util

#gulpfile.js

```
var uglify = require('gulp-util');

gulp.task('status', function() {
    gutil.log(gutil.colors.yellow('Gulp en modo' + gutil.env.prod ? ' producción' : ' desarrollo'));
    gutil.log(gutil.colors.blue('Para activar Gulp en producción: gulp --prod'));
});
```

Gulp

Plugins básicos

gulp-uglifyjs: Crea un archivo JavaScript a partir de varios y lo minifica https://www.npmjs.com/package/gulp-uglifyjs

npm install —save-dev gulp-uglifyjs

#gulpfile.js

```
var uglify = require('gulp-uglifyjs');

gulp.task('uglify', function() {
  gulp.src('public/js/*.js')
    .pipe(uglify())
    .pipe(gulp.dest('dist/js'))
});
```

Gulp

Plugins básicos

gulp-sass: Procesa archivos sass a css.
https://www.npmjs.com/package/gulp-sass

npm install —save-dev gulp-sass

#gulpfile.js

```
var uglify = require('gulp-uglifyjs');

gulp.task('styles', function() {
  gulp.src('assets/css/*.scss')
    .pipe(sass())
    .pipe(gulp.dest('web/css'))
});
```

Gulp

Plugins básicos

gulp-sass: Procesa archivos sass a css.
https://www.npmjs.com/package/gulp-sass

npm install —save-dev gulp-sass

#gulpfile.js

```
var uglify = require('gulp-uglifyjs');

gulp.task('styles', function() {
  gulp.src('assets/css/*.scss')
    .pipe(sass())
    .pipe(gulp.dest('web/css'))
});
```

Gulp

Plugins básicos

gulp-clean-css: Minifica archivos css

https://www.npmjs.com/package/gulp-clean-css

npm install —save-dev gulp-clean-css

#gulpfile.js

```
var cleanCSS = require(‘gulp-clean-css');

gulp.task('styles', function() {
  gulp.src('assets/css/*.scss')
    .pipe(sass())
    .pipe(cleanCSS())
    .pipe(gulp.dest('web/css'))
});
```

Gulp

Plugins básicos

gulp-sourcemaps: Permite poder ver desde el inspector de código del navegador, el archivo original donde se encuentra el código.

https://www.npmjs.com/package/gulp-sourcemaps

npm install —save-dev gulp-sourcemaps

```
#gulpfile.js
var sourcemaps = require(‘gulp-clean-css’);

gulp.task('styles', function() {
  gulp.src(‘assets/css/*.scss')
    .pipe(sourcemaps.init())
    .pipe(sass())
    .pipe(cleanCSS())
    .pipe(sourcemaps.write())
    .pipe(gulp.dest('web/css'))
});
```

Assets en Symfony

Assets Symfony

Adios a assetic

Desde la versión 2.8, Symfony no incluye por defecto AsseticBundle, aunque podemos seguir utilizándolo instalándonos en bundle

http://symfony.com/doc/current/assetic/asset_management.html

Assets Symfony

Ubicando nuestros assets

En Symfony ubicamos nuestros assets en los directorios Resources/public de los *bundles* o en *app/*

En el libro de buenas prácticas de Symfony, recomienda almacenar todos los assets en el directorio público *web/*

Assets Symfony

Ubicando nuestros assets

Para no guardar los assets originales en la carpeta publica, los guardamos en `app/Resources/assets` y a través Gulp (u otras herramientas) se procesan y generan los assets dentro de la carpeta `/web`

Assets Symfony

Estructura de un proyecto

```
app
├── Resources
│   ├── assets
│   ├── public
│   └── views
├── [ . . .]
├── composer.json
├── composer.lock
├── node_modules
├── package.json
├── gulpfile.js
├── src
├── vendor
└── web
```

Assets Symfony

Estructura de un proyecto

```
app
├── Resources
│   └── assets
│       ├── img
│       ├── js
│       └── sass
├── [ . . .]
├── composer.json
├── composer.lock
├── node_modules
├── package.json
├── gulpfile.js
├── src
├── vendor
└── web
```

Assets originales

Assets Symfony

Estructura de un proyecto

```
app
├── Resources
│   ├── assets
│   ├── public
│   └── views
├── [ . . .]
├── composer.json
├── composer.lock
├── node_modules
├── package.json
├── gulpfile.js
├── src
├── vendor
└── web
```

Gestión de dependencias Frontend

Assets Symfony

Estructura de un proyecto

```
app
├── Resources
│   ├── assets
│   ├── public
│   └── views
├── [ . . .]
├── composer.json
├── composer.lock
├── node_modules
├── package.json
├── gulpfile.js
├── src
├── vendor
└── web
```

Dependencias Frontend

Gestión de dependencias Frontend

Assets Symfony

Estructura de un proyecto

```
app
├── Resources
│   ├── assets
│   ├── public
│   └── views
├── [ . . .]
├── composer.json
├── composer.lock
├── node_modules
├── package.json
├── gulpfile.js
├── src
├── vendor
└── web
```

Configuración Gulp

Assets Symfony

Estructura de un proyecto

```
app
├── Resources
│   ├── assets
│   ├── public
│   └── views
├── [ . . .]
├── composer.json
├── composer.lock
├── node_modules
├── package.json
├── gulpfile.js
├── src
├── vendor
└── web
```

Assets públicos

¡Usas Git!

.gitignore

Recuerda no incluir:

node_modules/
web/assets

Ejemplo práctico

Symfony + Gulp

https://github.com/JuanLuisGarciaBorrego/symfony-pack

Gracias

FeedBack:

blog.juanluisgarciaborrego.com

@JuanluGarciaB

juanluisgarciaborrego@gmail.com