Gulp.js

Instalación, configuración y uso

Introducción

Transpilers (Babel, SCSS, Less)

Linters (JSHint, ESLint)

Empaquetadores (Browserify, WebPack)

Test runners (Karma, Mocha, Nightwatch)

Ofuscadores/minificadores (Uglify)

¿Qué es GULP?

Gulp.js es un sistema de construcción que permite automatizar tareas comunes de desarrollo, tales como:

- Minificación de código JavaScript
- Recarga del navegador
- Compresión de imágenes
- Validación de sintaxis de código
- Concatenación de ficheros

Instalación de GULP

Primero debemos instalar **Node.js (**Descarga: https://nodejs.org/en/download/**)**

Una vez instalado Node.js, desde una consola, ejecutar:

```
npm install -g gulp
```

Si estás usando Linux o Mac, tal vez necesites anteponer la palabra `sudo` para poder ejecutar este comando con los permisos de administrador, así:

Verificamos que Gulp.js ha sido instalado correctamente.

```
gulp -v
```

Si lo tenemos instalado correctamente, nos mostrará lo siguiente:

```
CLI version 3.8.6
Local version undefined
```

Ejemplo de uso

Concatenar dos archivos .js en uno solo y luego lo minificará.

Así que configuraremos 2 tareas:

- concatenar y minificar

Todo esto contenido en una tarea llamada “demo”.

Ejemplo - desde un terminal

Crear una carpeta llamada: gulp-ejemplo. Acceder a esta carpeta mediante terminal.

Luego allí dentro creamos el archivo: gulpfile.js (archivo que Gulp.js necesita para saber qué tareas realizará). De momento no le podremos ningún contenido.

Ejemplo - desde un terminal

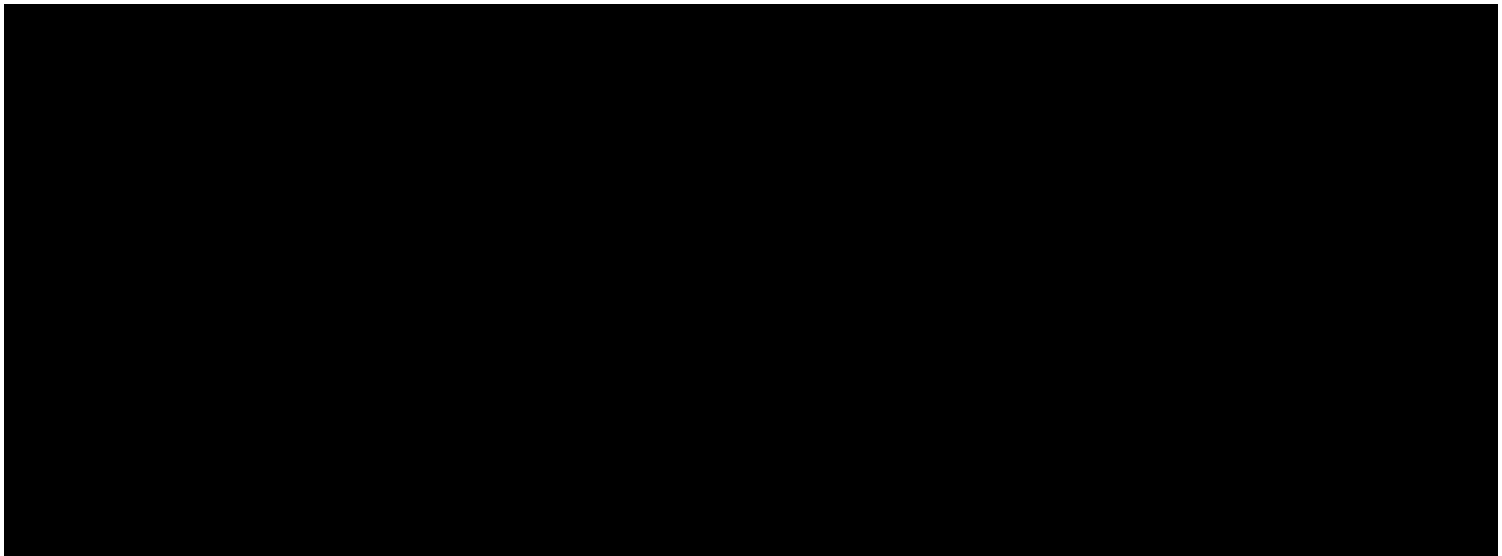
Ejecutar el comando `npm init`

Puesto que es un ejemplo, podemos dar a enter y dejar los campos

Con esto, npm nos debe haber creado un archivo llamado: package.json

Ejemplo - desde un terminal

package.json



Agregar plugins

Ahora agregaremos plugins de desarrollo al proyecto

Primero a instalar será: **gulp**, así que escribimos lo siguiente en nuestra terminal:

```
npm install --save-dev gulp
```

Luego instalamos: **gulp-concat**

```
npm install --save-dev gulp-concat
```

Y finalmente instalamos: **gulp-uglify**

```
npm install --save-dev gulp-uglify
```

También podemos escribir todas las dependencias en la misma orden

Agregar las dependencias

Como podremos observar, nuestro archivo package.json ha cambiado

Plugins - Gulp

gulp-concat

- Para juntar varios archivos en uno mismo.
- Menos peticiones al servidor

gulp-uglify

- Sirve para minificar archivos js
- Código se interpreta antes
- Permite ejecutar una tarea de manera condicional

Estructura de carpetas y ficheros

- Un **glob** es una cadena que coincida con uno o más archivos usando los patrones que usa el intérprete de comandos de unix
- Ejemplos de glob:
  - `js/source/1.js` coincide exactamente con el archivo.
  - `js/source/*.js` coincide con los archivos que terminen en .js dentro de la carpeta `js/source`.
  - `js/**/*.js` coincide con los archivos que terminen en .js dentro de la carpeta `js` y dentro de todas sus subcarpetas.
  - `!js/source/file3.js` Excluye especificamente el archivo llamado file3.js.
  - `static/*.+(js|css)` coincide con los archivos que terminen en .js ó .css dentro de la carpeta `static/`

Si necesitamos encontrar todos los archivos que terminen en .js dentro de la carpeta js/source

```
gulp.src('js/source/*.js')
```

Retomando el ejemplo

Creamos dentro de js/src un fichero llamado file1.js

```
// comentario para este fichero JS
function sumar (a){
  console.log("...realizando el incremento en 1");
  return a + 1;
};
```

En esa misma ruta crearemos un segundo fichero llamado file2.js

```
// otro comentario de línea de un fichero javascript
function restar (a){
  console.log("...realizando una resta");
  return a - 1;
};
```

Retomando el ejemplo

Añadir al fichero lo siguiente

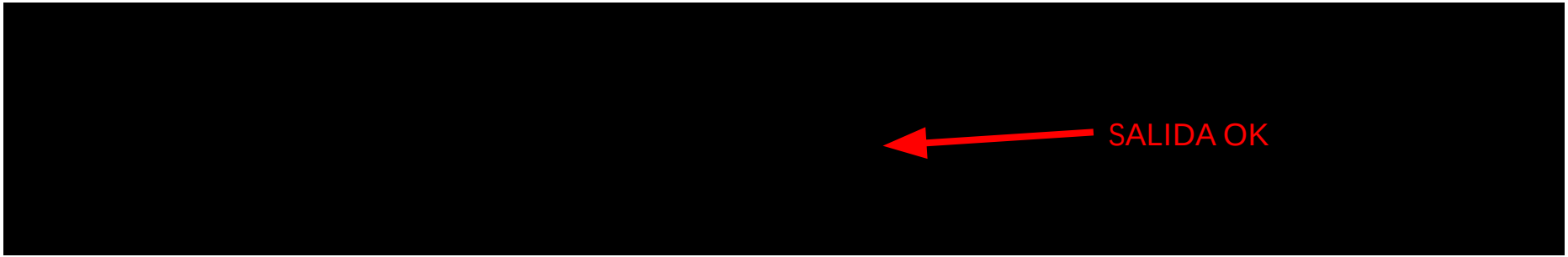
```
// Dependencias
var gulp = require('gulp'),
  concat = require('gulp-concat'),
  uglify = require('gulp-uglify');

// Configuración de la tarea 'demo'
gulp.task('demo', function(done) {
  gulp.src('js/src/*.js')
  .pipe(concat('todo.js'))
  .pipe(uglify())
  .pipe(gulp.dest('js/build/'))
  done(),
});
```

¿Cuál ha sido el resultado del ejemplo 2 (I)

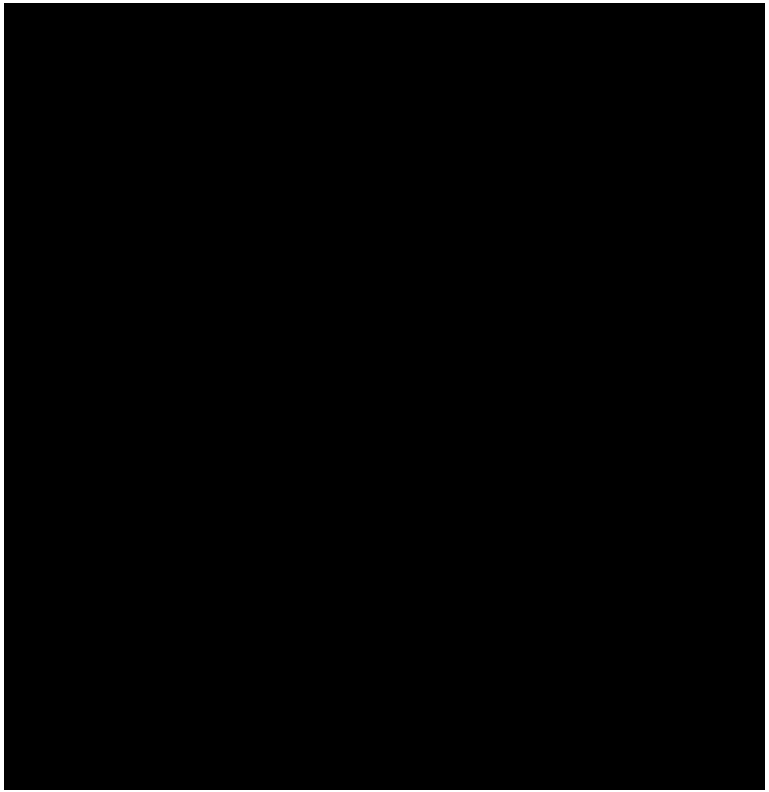
Una vez está todo configurado, vamos a ponerlo a prueba desde un terminal

Comprobar que la tarea creada está disponible

Invocamos la tarea



¿Cuál ha sido el resultado del ejemplo 2 (II)



¿Cuál ha sido el resultado del ejemplo 2 (III)

El resultado es un único fichero llamado todo.js (dentro de la carpeta build) que contiene el código de los ficheros js (file1.js y file2.js) de src

...el fichero gulpfile.js - explicación dependencias

Primeramente indicar los paquetes (separados por “comas”) necesarios para llevar a cabo las tareas que queremos.

En este caso, requerimos los siguientes paquetes: gulp, gulp-concat y gulp-uglify, así:

```
// Dependencias
var gulp = require('gulp'),
  concat = require('gulp-concat'),
  uglify = require('gulp-uglify');
```

...el fichero gulpfile.js - explicación API de gulp -

gulp.task()

- Define una tarea
- Tiene 3 argumentos:
  - el nombre de la tarea
  - dependencias de otras tareas
  - la función a ejecutar

```
ftko-s`rj'&gdkkn&+ etmbshnm'( z

  bnmrnkd-knf'&Gdkkn+ Vnqkc6&(:

|(:
```

El código anterior define la tarea llamada `gdkkn` que se ejecuta desde un terminal con la siguiente orden

```
ftko gdkkn
```

... el fichero gulpfile.js, explicación API de gulp -

gulp.src()

- Toma como parámetro una cadena (glob) que coincida con uno o más archivos
- Utiliza patrones que usa el intérprete de comandos de unix(shell)
- Retorna un stream que puede ser “pipeado” a un plugin adicional ó hacia el método gulp.dest()

```
ftko-s`rj'&sdrs&+ etmbshnm 'cnmd(z
    ftko-rqb'&sdrs-sws&(
          -ohod'ftko-cdrs'&nts&((:
    cnmd'(:
|(:
```

Dm drsd b`rn snl` bnln fkna k`
b`cdm` sdrs-sws-

@cdlÒr drs`akdbd bnln cdrshmn cd k`
s`qd` `k chqdbsnqhn þntsR- Dr cdbhq+
V ㇁ ᑐᑐᐸ k&nss-b snb ᴣrn'