WebPack

Antonio Espín Herranz

Introducción

- Webpack es un **empaquetador de módulos** que toma tus archivos JavaScript, CSS, imágenes y otros recursos, y los combina en un solo archivo o en varios optimizados para el navegador.

- Es especialmente útil para proyectos modernos de desarrollo frontend.
  - Utilizado por frameWorks de JS como **Angular**, **React** o **Vue.js**

Instalación

- En la carpeta del proyecto:, instalar webpack:
- **npm install --save-dev webpack webpack-cli**

- La opción **--save-dev** indica que las dependencias que estamos instalando sólo son **necesarias** para **el desarrollo del proyecto** y no se necesitan para el entorno de producción.

- Con esta opción se instalan:
  - Compiladores
  - Herramientas de prueba

- Si instalamos así: **npm install webpack webpack-cli** (se van a instalar también en producción)

Conceptos

- **Entry Point o punto de entrada**. Sirve para indicar el punto exacto desde el que Webpack empezará a analizar el código para generar los paquetes.
  - Normalmente será un fichero: **index.ts** o **index.js** situado en la carpeta **src** del proyecto.

- **Output** o punto de salida. Indica el lugar donde se colocarán los paquetes que se hayan generado: JavaScript, CSS, HTML, etc
  - Normalmente puede ser la carpeta: **dist**

- **Loaders** o cargadores. Son los sistemas que hacen posible que Webpack transforme todos los archivos o ficheros requeridos (todo aquello que deba modificarse en la aplicación debe cargarse con un loader).
  - Se utilizarán loaders para las imágenes, ficheros ts, css, etc.

- **Plugins**. Los plugins amplían el rango de funcionalidades por defecto de Webpack; permiten tareas como la optimización del código empaquetado, la gestión y optimización de las imágenes, incluir trazas entre cargadores, comprobar cómo ha ido la ejecución o añadir código o variables en los archivos de nuestra aplicación.

Conceptos

- **Punto de inicio predeterminado.** En el caso de que no proveamos a Webpack de archivo de configuración **webpack.config.js**, éste interpreta que el punto de inicio predeterminado para JavaScript es ***/src/index.js***, y comenzará a leer todo nuestro JavaScript desde ese archivo, generando un paquete con todo el código JavaScript que necesite.

- **Punto de salida predeterminado.** Para JavaScript el archivo "**dist/main.js**" es el punto de salida predeterminado de Webpack, dentro del cual se coloca el ***bundle*** que contiene todo nuestro código

Conceptos

- Dentro de fichero **package.json** se puede programar un script:

  'scripts': {

  > 'build': 'webpack --mode production',

  'builddev': 'webpack --mode development'

  },

  - El modo se puede indicar en el fichero de configuración: webpack.config.js

- Lo hacemos correr con:
  - **npm run build**

Configuración: webpack.config.ts

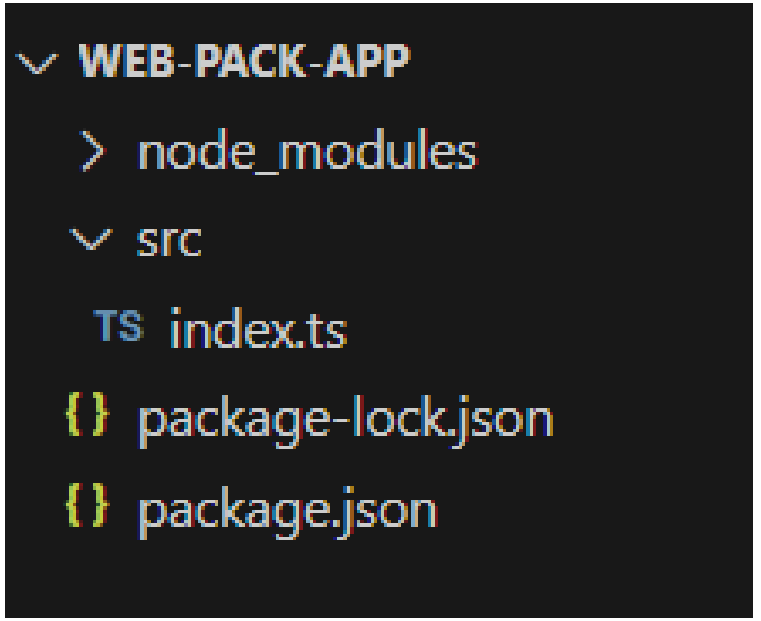
```
const path = require('path');

module.exports = {
  output: {
    path: path.resolve(__dirname, 'public_html/js'),
    filename: 'app.js'
  },
  entry: {
    main: './fuentes/inicio.js'
  }
}
```

- Path se utiliza para resolver rutas.
- Con las **Entry** definimos dónde **Webpack** empieza a tratar el código de nuestra aplicación para formar los paquetes; mientras que **Output filename** sirve para indicar el nombre de archivo del paquete que va a producir.

webpack

- Empezamos creando una carpeta para el proyecto.
- Dentro de la carpeta tendremos una subcarpeta **src**
- Y dentro de esta el fichero index.ts (el punto de entrada)
- Inicializar la aplicación:
  - **npm init -y** (desde la carpeta raíz del proyecto)
  - Sin la opción –y nos permite personalizar los valores



- La estructura de la aplicación será así:
  - **app**
    - src
      - index.ts
    - package.json

- Instalamos las dependencias:
  - **npm install --save-dev typescript ts-loader webpack webpack-cli**

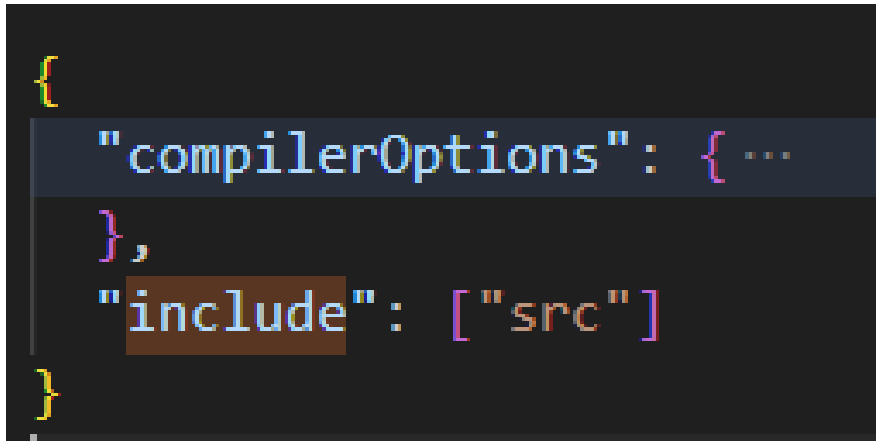
Las dependencias

- Instalar siempre en cada proyecto distinto, para evitar problemas con las versiones:
- **typescript**: compilador de typescript
- **ts-loader**: Loader de webpack para manejar los archivos de typescript
- **webpack** y **webpack-cli** para empaquetar el proyecto.

Archivo tsconfig.json

- Para crear el archivo de configuración:
- **tsc --init**

- Al final del documento **json** incluimos:



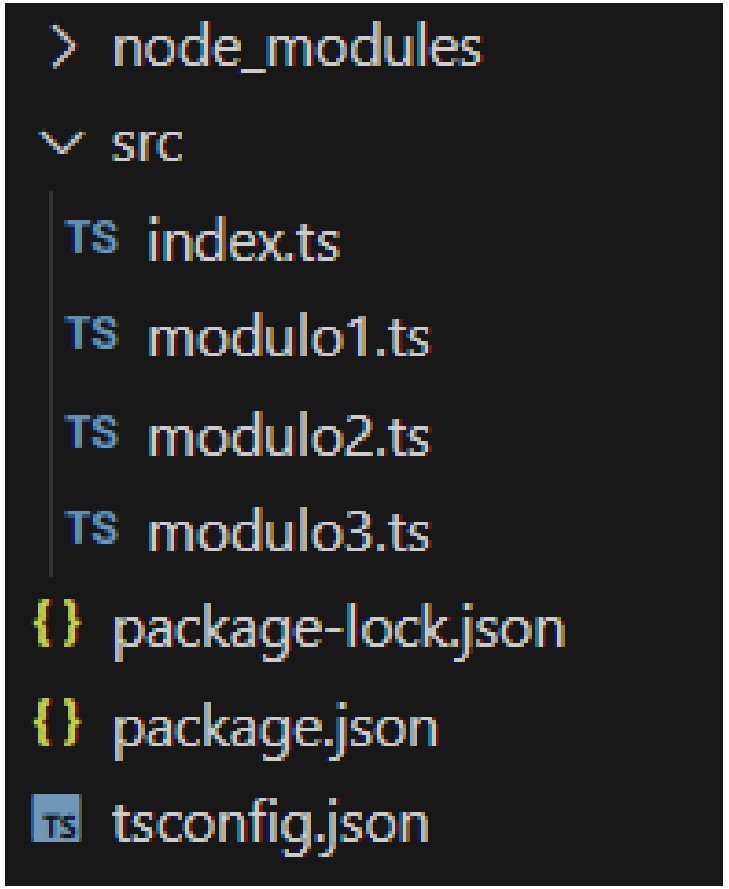
Crear los módulos ts

- Dentro de la carpeta **src** creamos los módulos de **ts** necesarios:

- En el fichero **index.ts** importamos los módulos y vamos llamando a las funciones.
- No es necesario indicar la extensión.

```ts
import { saludar } from "./modulo1";
import { despedir } from "./modulo2";
import { obtenerHora } from "./modulo3";

console.log(saludar("Juan"));
console.log(despedir("Juan"));
console.log(obtenerHora());
```



Estructura final

```
mi-proyecto/
├── dist/               # Carpeta de salida (generada por Webpack)
├── node_modules/       # Dependencias instaladas
├── src/                # Archivos fuente
│   ├── index.ts        # Archivo principal
│   ├── modulo1.ts      # Primer módulo
│   ├── modulo2.ts      # Segundo módulo
│   └── modulo3.ts      # Tercer módulo
├── package.json        # Archivo de configuración de NPM
├── tsconfig.json       # Archivo de configuración de TypeScript
├── webpack.config.js # Archivo de configuración de Webpack
```

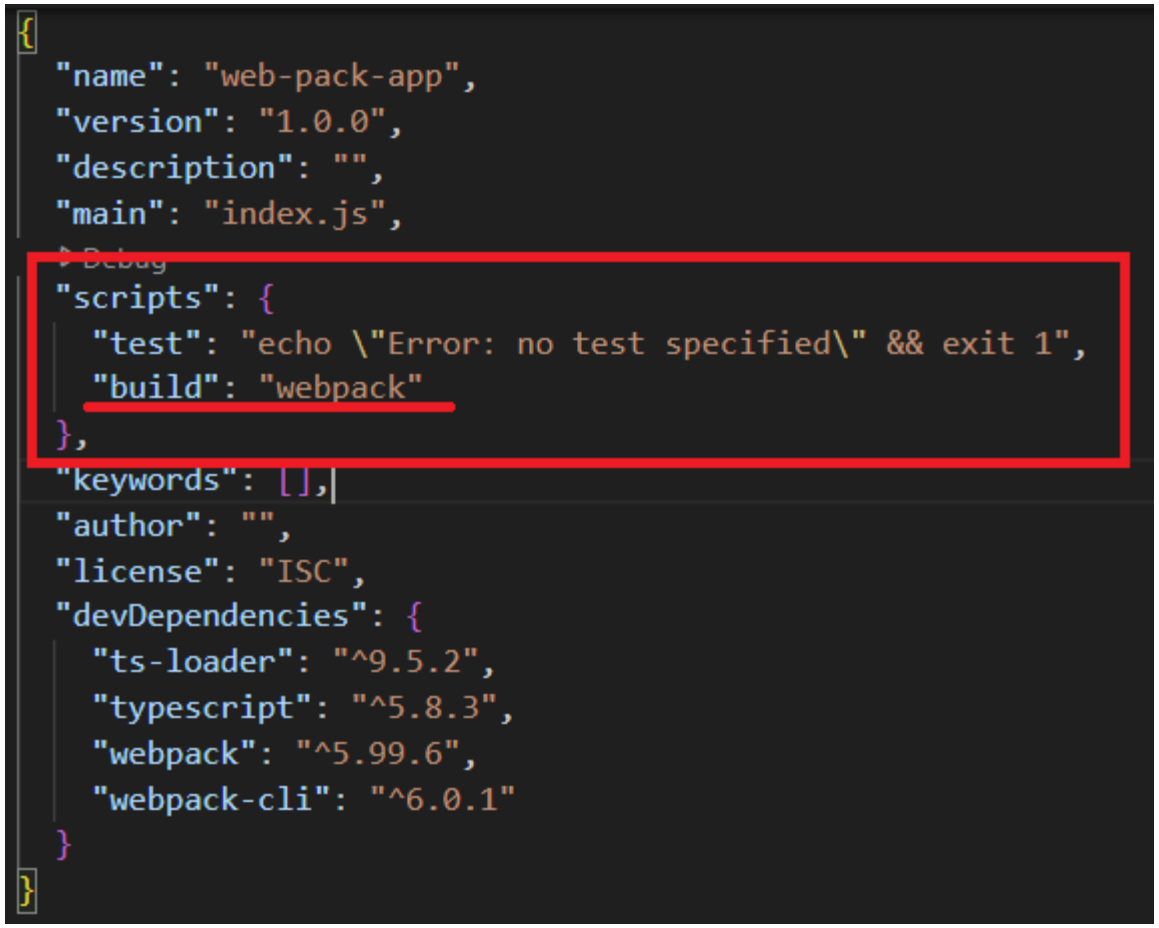
El archivo webpack.config.js

```
const path = require('path');

module.exports = {
    entry: './src/index.ts', // Punto de entrada
    output: {
        filename: 'bundle.js', // Archivo de salida
        path: path.resolve(__dirname, 'dist'), // Carpeta de salida
    },
    resolve: {
        extensions: ['.ts', '.js'], // Extensiones que manejará Webpack
    },
    module: {
        rules: [
            {
                test: /\.ts$/,
                use: 'ts-loader',
                exclude: /node_modules/,
            },
        ],
    },
    mode: 'development', // Modo: desarrollo o producción
};
```

Empaquetar el proyecto

- **Modificar package.json**
  - En la sección de “**scripts**” está en el nivel superior del archivo **package.json** se añade el script “**build**”.



Empaquetar la aplicación

- Para empaquetar la aplicación lanzamos el comando:
  - **npm run build**

```
> web-pack-app@1.0.0 build
> webpack

asset bundle.js 3.98 KiB [emitted] (name: main)
./src/index.ts 369 bytes [built] [code generated]
./src/modulo1.ts 192 bytes [built] [code generated]
./src/modulo2.ts 198 bytes [built] [code generated]
./src/modulo3.ts 258 bytes [built] [code generated]
webpack 5.99.6 compiled successfully in 1352 ms
```

- Tiene que generar el fichero **bundle.js** en la carpeta **dist** del proyecto.

Prueba final

- Creamos un fichero index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo Webpack</title>
</head>
<body>
    <script src="./dist/bundle.js"></script>
</body>
</html>
```

Verificar

- En la consola del navegador:

```
Hola Juan
Adiós Juan
La hora es 17:22:56
```

Otro Proyecto con CSS, img, archivos ttf

- Configurar Webpack para manejar **TypeScript**, archivos **CSS**, fuentes **.ttf**, e imágenes **requiere añadir loaders específicos** y actualizar la configuración del archivo **webpack.config.js**

- ***npm install --save-dev style-loader css-loader file-loader ts-loader webpack webpack-cli***
  - **style-loader**: Inyecta los estilos CSS directamente en el DOM.
  - **css-loader**: Permite importar archivos CSS dentro del proyecto.
  - **file-loader**: Maneja archivos como imágenes y fuentes, copiándolos al directorio de salida.
  - **ts-loader**: Compila los archivos TypeScript para que Webpack los procese.

Estructura del proyecto

```
mi-proyecto/
├── dist/                    # Carpeta de salida generada por Webpack
├── node_modules/            # Dependencias instaladas
├── src/                     # Archivos fuente
│   ├── index.ts             # Archivo principal
│   ├── modulo1.ts           # Primer módulo
│   ├── modulo2.ts           # Segundo módulo
│   ├── modulo3.ts           # Tercer módulo
│   ├── styles.css           # Archivo CSS principal
├── assets/                  # Recursos estáticos
│   ├── fonts/               # Carpeta de fuentes
│   │   └── fuente.ttf
│   ├── images/              # Carpeta de imágenes
│       └── logo.png
├── webpack.config.js        # Configuración de Webpack
├── package.json             # Configuración de NPM
├── tsconfig.json            # Configuración de TypeScript
```

Modificar el fichero: webpack.config.js

```
},
resolve: {
    extensions: ['.ts', '.js'], // Extensiones que Webpack manejará
},
module: {
    rules: [
        // Manejar archivos TypeScript
        {
            test: /\.ts$/,
            use: 'ts-loader',
            exclude: /node_modules/,
        },
        // Manejar archivos CSS
        {
            test: /\.css$/,
            use: ['style-loader', 'css-loader'], // Procesa los estilos
        },
        // Manejar fuentes TTF
        {
            test: /\.(ttf|woff|woff2|eot|otf)$/,
            type: 'asset/resource',
            generator: {
                filename: 'fonts/[name][ext]', // Ruta de salida para fu
            },
        },
        // Manejar imágenes
        {
            test: /\.(png|jpe?g|gif|svg)$/,
            type: 'asset/resource',
            generator: {
                filename: 'images/[name][ext]', // Ruta de salida para i
            },
        },
    ],
},
mode: 'development', // Modo de desarrollo
```

- Se añaden reglas para los archivos CSS, fuentes TTF e imágenes.
  - TypeScript (ts-loader):
    - Procesa archivos .ts para convertirlos a JavaScript antes de empaquetarlos.
    - CSS (style-loader y css-loader):
    - css-loader: Permite importar CSS en los archivos .ts o .js.
    - style-loader: Inyecta los estilos CSS en el DOM.
    - Fuentes (file-loader o asset/resource):
    - Usa asset/resource para copiar las fuentes a la carpeta de salida (dist/fonts).
    - Imágenes (file-loader o asset/resource):
      - Copia las imágenes a la carpeta de salida (dist/images).

CSS: styles.css

```css
.boton:hover {
    background-color: #3700b3;
}

.footer {
    text-align: center;
    padding: 10px 0;
    background-color: #e0e0e0;
}
```

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f9;
    color: #333;
}

.header {
    background-color: #6200ea;
    color: white;
    text-align: center;
    padding: 20px;
}

main {
    padding: 20px;
    text-align: center;
}

.boton {
    background-color: #6200ea;
    color: white;
    border: none;
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    border-radius: 5px;
}
```

El fichero index.html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aplicación Webpack + TypeScript</title>
    <!-- Estilo CSS incluido desde el bundle generado -->
</head>
<body>
    <header class="header">
        <h1>¡Bienvenido a mi aplicación TypeScript!</h1>
    </header>
    <main>
        <p>Esta es una página sencilla empaquetada con Webpack.</p>
        <button class="boton">Haz clic aquí</button>
    </main>
    <footer>
        <p class="footer">Creado por Antonio 🚀</p>
    </footer>
    <!-- Incluimos el bundle.js generado por Webpack -->
    <script src="./dist/bundle.js"></script>
</body>
</html>
```

Modificar el fichero index.ts

- Hay que importar el fichero: **styles.css**

```
// Punto de entrada de la aplicación

import './styles.css'; // Importar estilos CSS

import { saludar } from "./modulo1";
import { despedir } from "./modulo2";
import { obtenerHora } from "./modulo3";

console.log(saludar("Juan"));
console.log(despedir("Juan"));
console.log(obtenerHora());
```

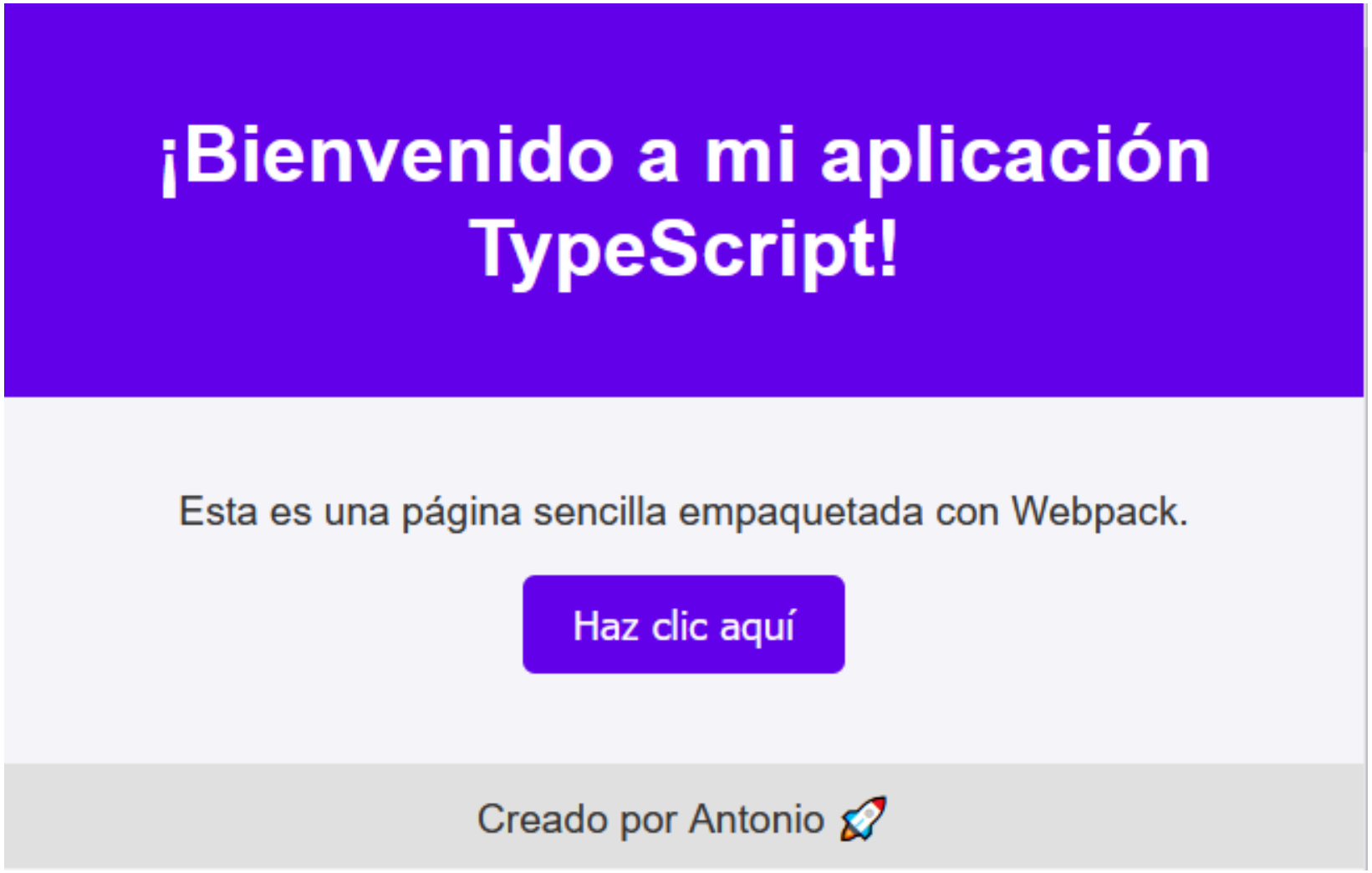
Empaquetar el proyecto

- **npm run build**

```
> web-pack-app@1.0.0 build
> webpack

asset bundle.js 25.6 KiB [emitted] (name: main)
runtime modules 972 bytes 5 modules
cacheable modules 11.3 KiB
  modules by path ./node_modules/ 8.15 KiB
    modules by path ./node_modules/style-loader/dist/runtime/*.js 5.84 KiB 6 modules
    modules by path ./node_modules/css-loader/dist/runtime/*.js 2.31 KiB
      ./node_modules/css-loader/dist/runtime/noSourceMaps.js 64 bytes [built] [code generated]
      ./node_modules/css-loader/dist/runtime/api.js 2.25 KiB [built] [code generated]
  modules by path ./src/ 3.17 KiB
    modules by path ./src/*.ts 1.04 KiB
      ./src/index.ts 418 bytes [built] [code generated]
      + 3 modules
    modules by path ./src/*.css 2.13 KiB
      ./src/styles.css 1.1 KiB [built] [code generated]
      ./node_modules/css-loader/dist/cjs.js!./src/styles.css 1.03 KiB [built] [code generated]
webpack 5.99.6 compiled successfully in 1821 ms
```

Resultado


¡Bienvenido a mi aplicación TypeScript!
Esta es una página sencilla empaquetada con Webpack.
Haz clic aquí
Creado por Antonio 🚀