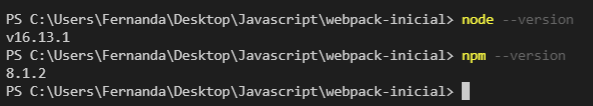
1. **Creando proyecto en node**
2. Abrir Terminal de vscode (ctrl + ñ) o la Terminal de Windows (cmd)

**Comprobar instalaciones previas**

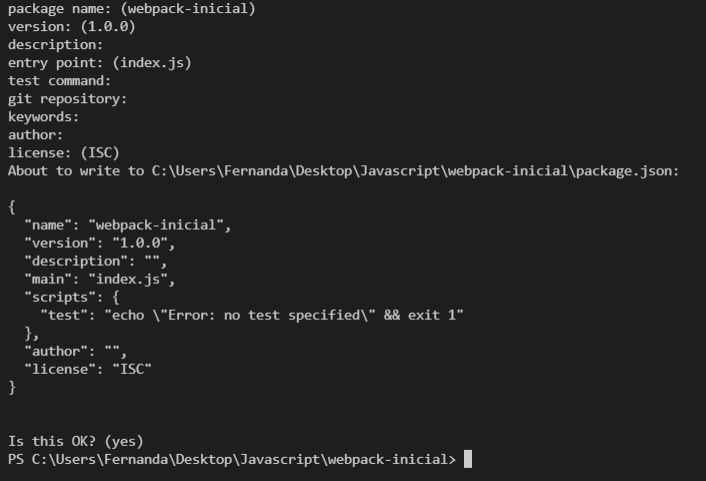
1. node –version
2. npm –version 🡪 gestionador de los paquetes de node



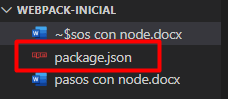
**Inicio del proyecto**

1. npm init 🡪 crea el archivo package.json

**Dar Enter a todo, así:**

****

**Me creará el archivo package.json**

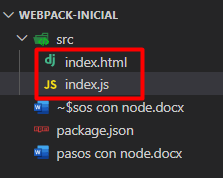
****

|  |
| --- |
| **Archivo Package.json contiene:**  -Paquetes necesarios para que la aplicación funcione.  -Paquetes que no serán necesarios en producción  -Comandos necesarios para crear la aplicación o paquete final. |

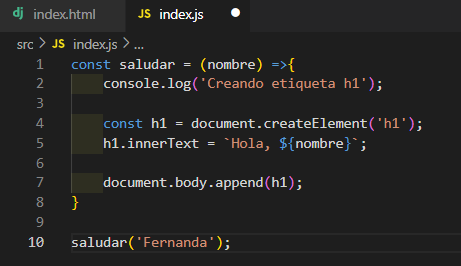
**2.0 Exposición del problema y necesidad del webpack**

1. Crear carpeta llamada **src**

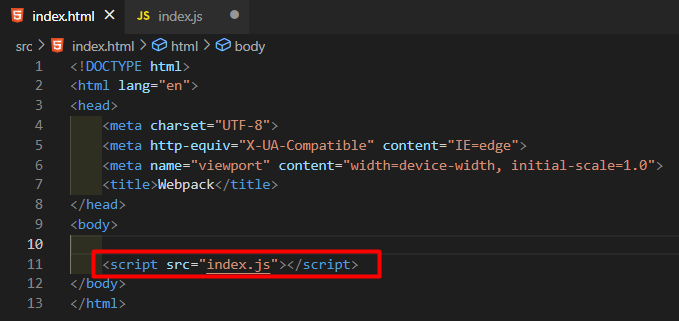
2. Dentro de la carpeta **src**, crear un archivo llamado index.html e index.js



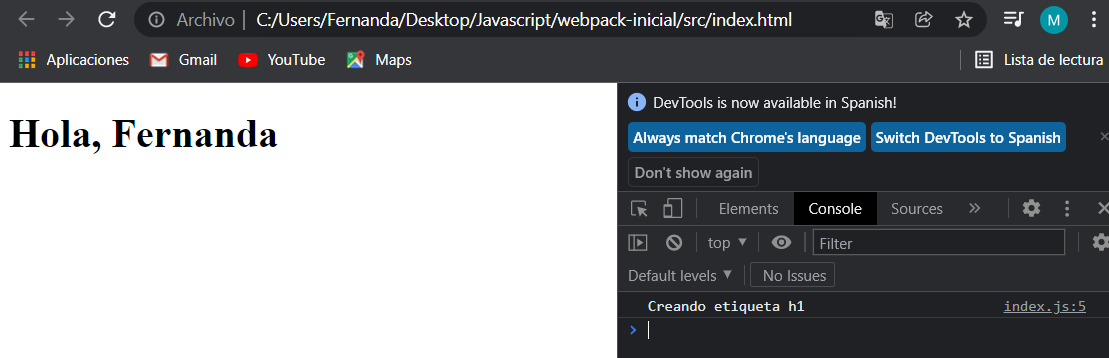
3. Dentro de index.js, crear la función saludar()



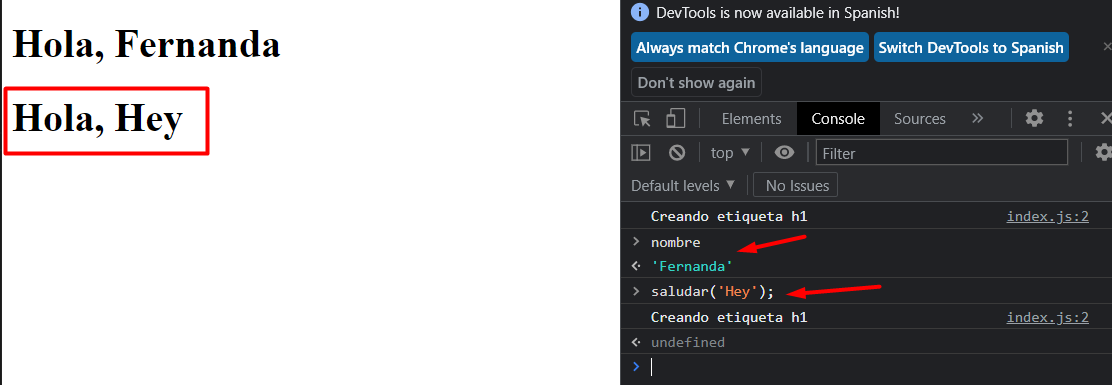
4 . Dentro del index.html, definir la estructura html e importar el index.js



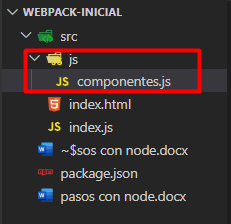
5 . Abrir archivo index.html en el navegador para probar el index.js



6 . En este momento, existe un problema de acceso a las variables y función, ya que están definidas como globales.

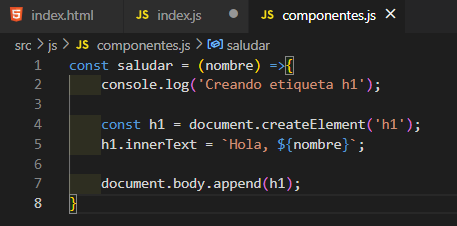


Podría definir un método autoinvocado que me privatice todo el bloque que contiene la constante nombre y función saludar(), pero yo no quiero usar eso, sino que, separaré el código, es decir, llamaré a la función saludar() desde otro archivo js ubicado en otra carpeta, llamado componentes.js.

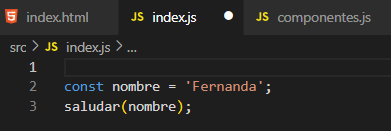


7. Luego moveré la función saludar() desde el index.js al componentes.js

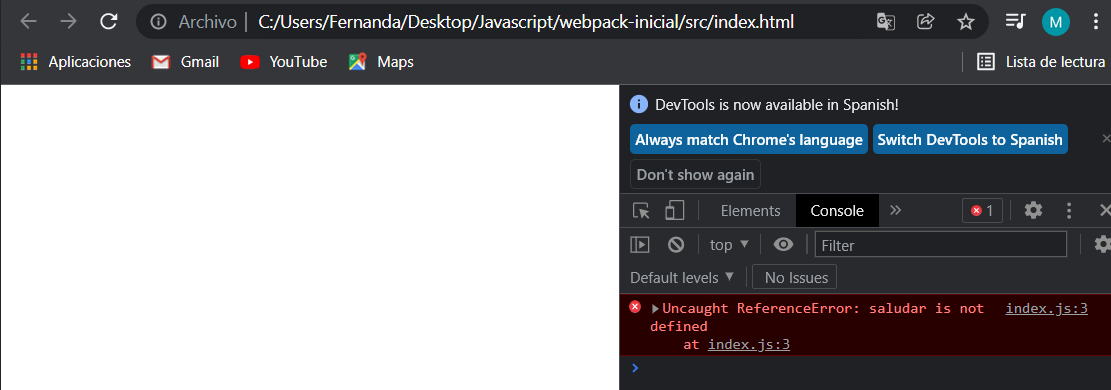
**componentes.js**

****

**Index.js**

****

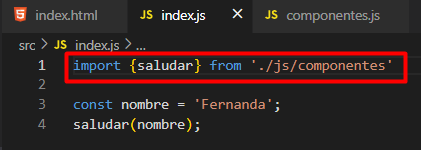
Recargo el navegador web, y me arrojará un problema de que la función que estoy llamando no existe



Para poder llamar una función desde otro archivo js, debo establecer la importación de módulos (dependencia entre archivos js)

**Importando módulos en js**

Debo crear la sentencia import en el archivo js que está llamando a la función saludar()



|  |
| --- |
| From 🡪 desde el archivo que contiene la función  Import 🡪 extraigo la función que necesito |

8. Abrir enlace de sitio oficial de webpack

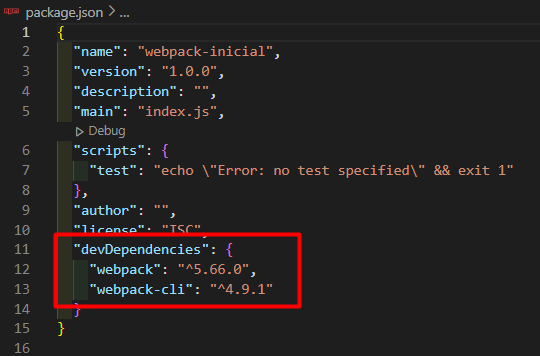
<https://webpack.js.org/guides/getting-started/>

**8.1** **Instala webpack en el proyecto**



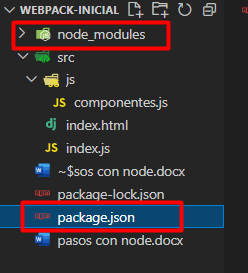
Esta instalación modificará el archivo package.json.

(Dependencias de desarrollo, es decir, dependencias que solo existirán en la fase de desarrollo, no así en la fase de producción)

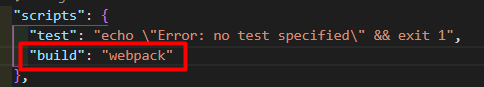


Además, crea la carpeta node\_modules.

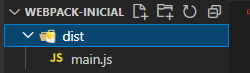
(**node\_modules** contiene solo los paquetes que se usarán en la fase de desarrollo. Pueden ir a la fase de producción solo algunos paquetes que estén previamente definidos para pasarlos también a producción).



9. Creo el script **build** que se ejecutará cada vez que lo corra (como comando build) por la ventana de comandos del Terminal de vscode.

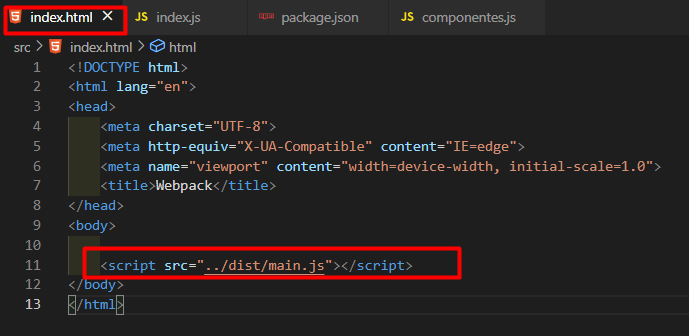


9.1 npm run build 🡪 ejecuto el comando build en la terminal

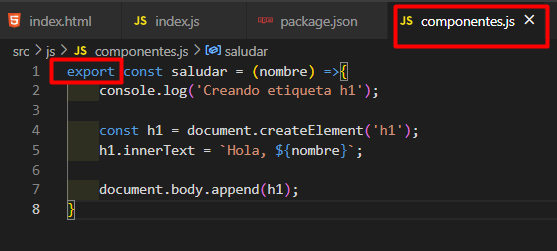


Este comando build, creará la carpeta **dist** con el archivo **main.js**

9.2 Modifico la ruta del script (hacia qué script está apuntando) en el index.html



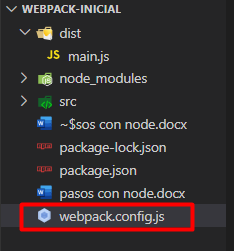
9.3 En el archivo que contiene mi función definida, es decir, componentes.js, agregar export en la función que deseo importar, en este caso, la función saludar()



9.4 Ejecutamos nuevamente el comando npm run build, para que tome los cambios y los actualice en mi archivo main.js

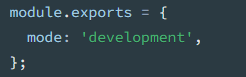
**3.0 Crear archivo de configuración**

1. En la raíz del proyecto, creo el archivo webpack.config.js

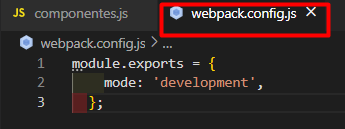


<https://webpack.js.org/configuration/>

2. Definimos el modo de desarrollo



Lo pasamos al archivo webpack.config,js



Luego ejecuto el comando **npm run build** en la Terminal de vscode

3. Realizar la instalación de html-loader en la Terminal de vscode

<https://webpack.js.org/loaders/html-loader/>



4 . Realizar instalación de webpack-plugin

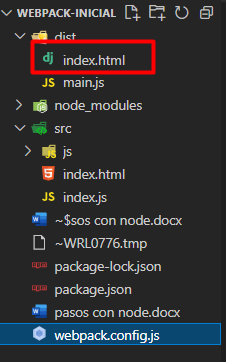
<https://webpack.js.org/plugins/html-webpack-plugin/>



5 . Definir configuración de mi archivo webpack.config.js



6 . La configuración anterior genera un nuevo archivo index.html en la carpeta dist, como una copia exacta del archivo index.html de la carpeta src



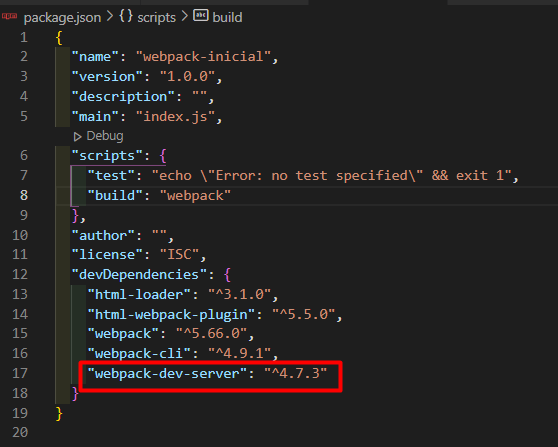
**4.0 . Instalar Webpack-server**

<https://webpack.js.org/configuration/dev-server/>

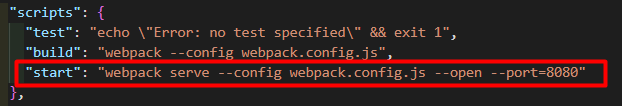


npm i -D webpack-dev-server

En las dependencias de mi archivo package.js se puede observar que se instaló el webpack-server



Luego, en el mismo archivo package.json, en scripts, crearé otro comando que servirá para levantar el servidor.



Puerto: localhost/8080

npm start 🡪 levanta el servidor (en la Terminal del vscode)

**5.0 . Importando estilos de forma dinámica**

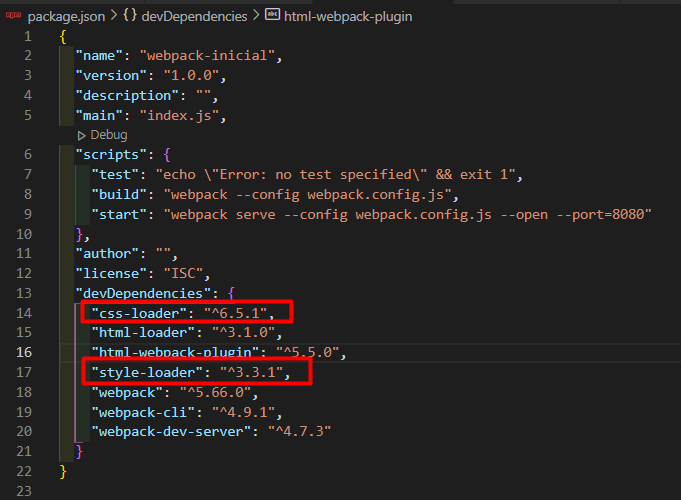
1 . Instalación de css-loader



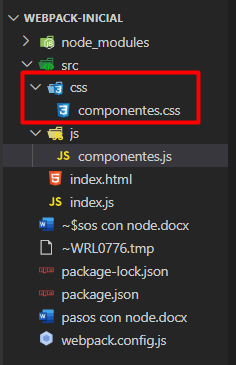
2 . Instalación de style-loader



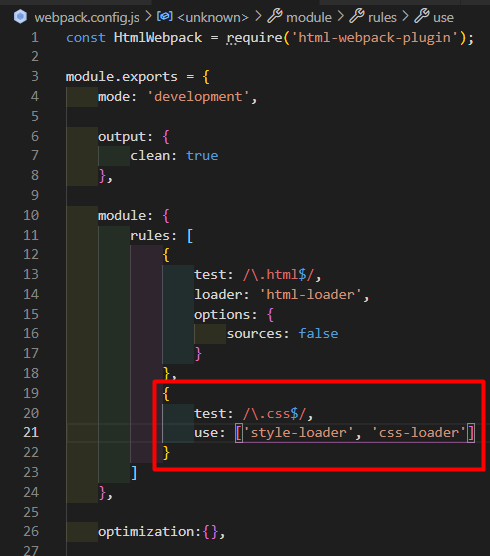
3 . El archivo package.json quedará así:



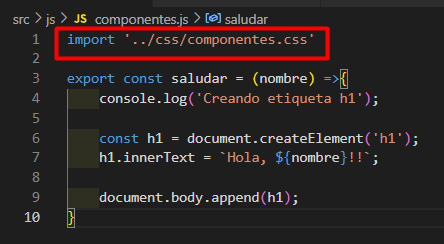
4 . Crear carpeta y archivo css dentro de src



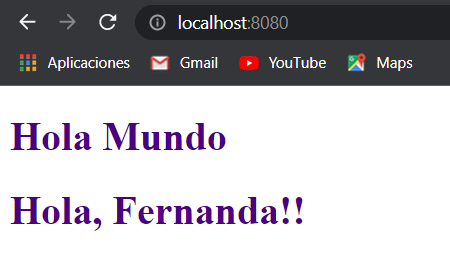
5 . Se debe crear otra regla en el archivo webpack.config.js



6 . Importar archivo css al archivo componentes.js. que es donde tengo definido mi método saludar()

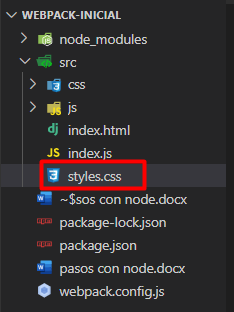


Así quedará:

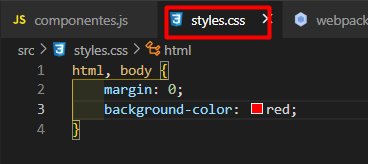


**6.0 . Archivo css global**

1 . Crear archivo styles.css en la carpeta de src.



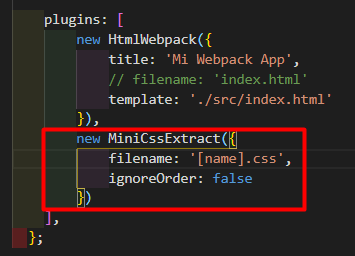
2 . Definir propiedades al styles.css



3 . Establecer otra regla en el archivo webpack.config.js



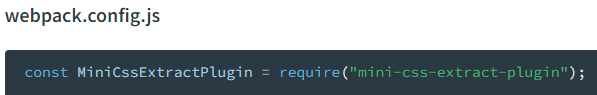
Agregar al campo plugins



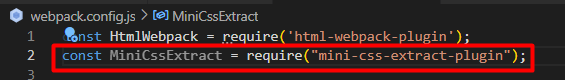
4 . Instalar mini-css-extract-plugin



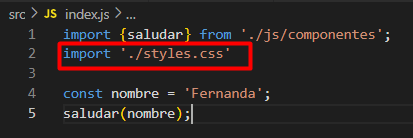
5 . Pegar const en el archivo webpack.config.js



Quedará así



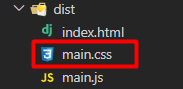
6 . Importar styles.css a index.js



7 . Ejecutar el comando build

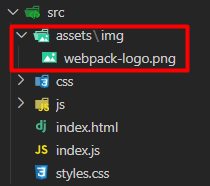
npm run build

Se generará el archivo nuevo-estilo.css en la carpeta dist



**7 .0 Manejo de imágenes en webpack**

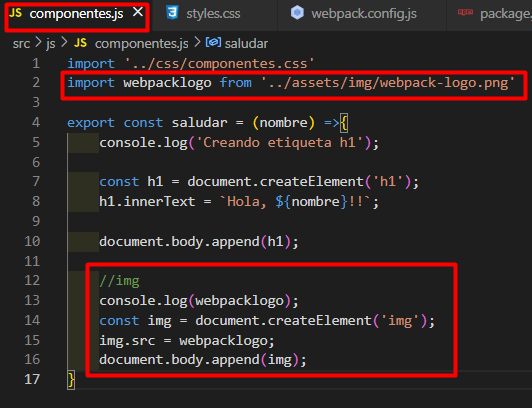
1 . Crear carpeta assets/img en el src.



2 . Instalar file-loader



3 . Importar y crear la imagen en el html desde js



4 . Crear una nueva regla en el webpack.config.js

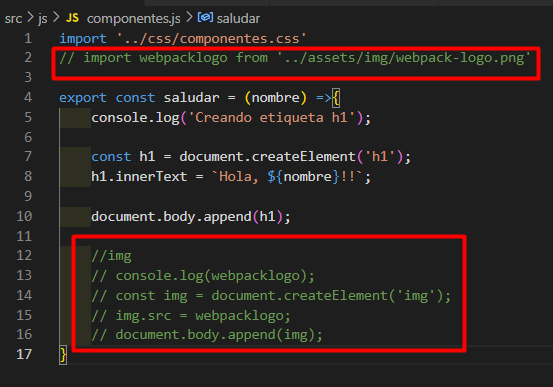


**7 .0 . Copiar y generar carpeta assets con archivo estático en el dist**

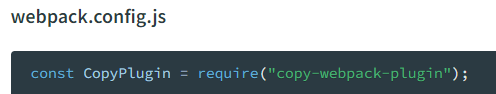
1 . Instalar copy-webpack-plugin



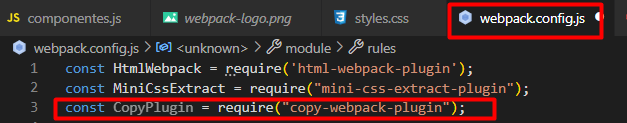
2 . Comentar la importación y el bloque que crea la imagen en el html



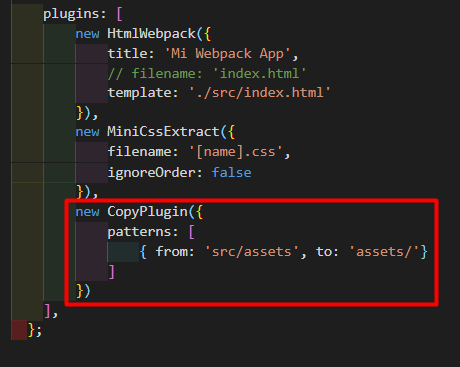
2 . Pegar el const en el archivo webpack.config.js



Quedará así:

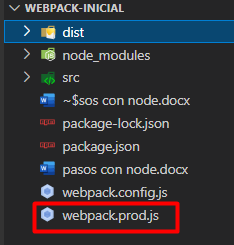


3 . Agregar plugin a la sección de plugins del webpack.config.js

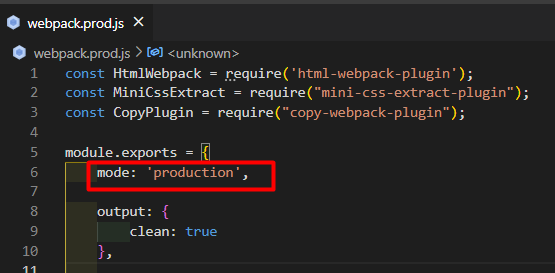


**8 .0 . Modo Producción**

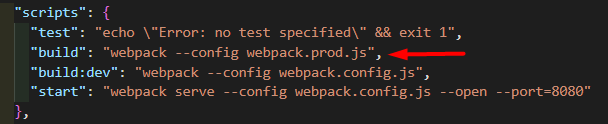
1 . Crear un nuevo archivo de configuración de webpack, pero destinado a producción



2 . Cambiarle el modo a producción



3 . En el archivo package.js definir el build en modo producción (relacionado al archivo webpack.prod.js). Ojo que también quedará otro build, pero para el modo de desarrollo

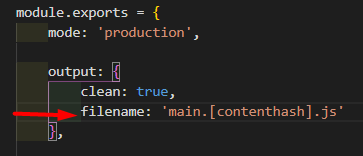


4 . Modificar el archivo webpack.prod.js

Sección plugins:

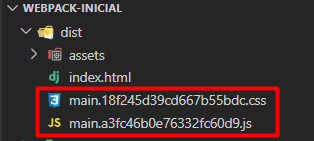


Sección output:



**Ejecutar el npm run build**

Quedará así (archivo main y archivo css):



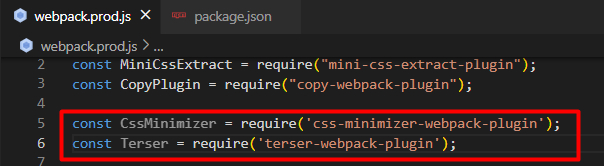
5 . Compactar el archivo css para usarlo en modo producción

En la terminal ejecutar el comando:

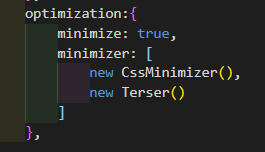
npm i -D css-minimizer-webpack-plugin terser-webpack-plugin

6 . Modificar el archivo webpack.prod.js

Sección inicial:



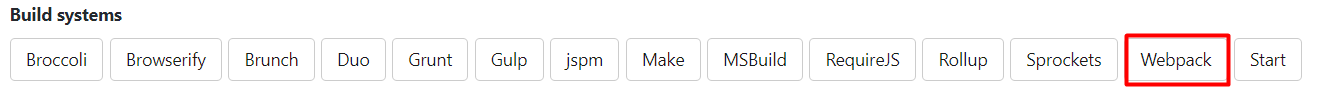
Sección de optimization:

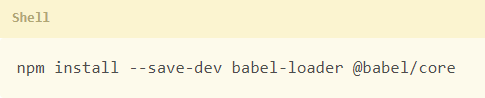


**9 .0 Instalación y configuración de Babel**

<https://babeljs.io/setup#installation>



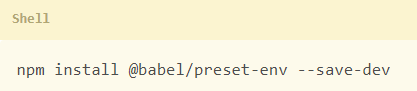




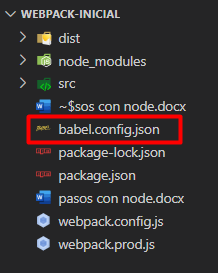
1 . Agregar regla al archivo webpack.prod.js



2 . Instalar en el vscode:



3 . Crear archivo Babel (en la raíz del proyecto)



4 . Modificar el archivo babel.config.js

