Curso de Webpack

# Introducción a webpack

**Webpack** es un empaquetador para Javascript y sus amigos. Convierte módulos con dependencias en archivos estáticos que los navegadores entienden.

Nos permite empaquetar, optimizar los diferentes módulos Javascript y sus dependencia en nuestro proyecto. Es usado en proyectos basados en Javascript como: React, Vue, Angular entre otros.

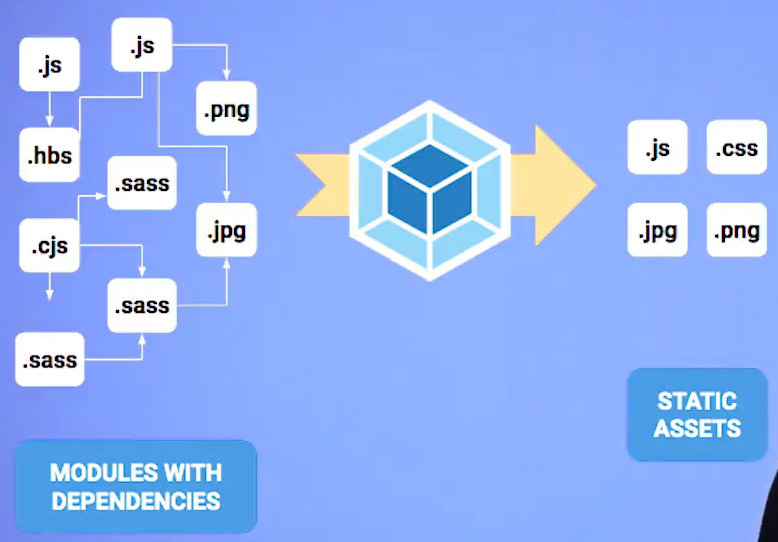
### User Experience

Se logra con una aplicación que:

* Funcione
* Sea rápida
* Cumpla sus necesidades
* Se actualice
* Responda a sus interacciones
* Producto de calidad

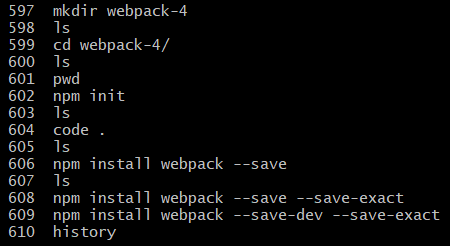
### Developer Experience

* Escribir aplicaciones de manera eficiente.
* Tener un código limpio.
* Aplicar tecnología para resolver sus problemas.
* Tener un conjunto de reglas y convenciones.
* Entorno de desarrollo optimizado en productividad.



Webpack es un conjunto de reglas que vamos a configurar para saber cómo funciona nuestro stack (de archivos), que está incluido en nuestro stack, y cómo manipulamos nuestro stack para que nuestra tecnología pase de la Developer Experience (Modules with Dependencies) a la User Experience (Static Assets).

# Configurando un nuevo proyecto de Javascript

En esta clase el profesor Leonidas Esteban nos guía paso a paso para crear nuestro primer proyecto usando Webpack.

En nuestro proyecto existen dependencias del tipo desarrollo y del tipo core con sus respectivas versiones, por eso usaremos NPM para administrarlas e instalar Webpack que viene a ser una dependencia (de desarrollo) más en nuestro proyecto de Javascript.

Se puede reemplazar install por su shortcut i así:

npm **install** webpack

npm **i** webpack

Las banderas con doble guión como --save se pueden reemplazar con su shortcut usando mayúsculas y sólo un guión así:

npm **install** webpack --save

npm **i** webpack -S

El shortcut de --save-dev es -D

npm install webpack --save-dev

npm **i** webpack -D

Para instalar una versión precisa del paquete se puede agregar a su nombre el número de la versión después del signo @ así:

npm **i** webpack@4.32.2 -D

Este método agregará webpack a las dependencias de desarrollo con el número de versión 4.32.2 pero cuando se descarguen los paquetes se instalará la última versión del paquete debido al prefijo ^

"devDependencies": {

"webpack": "^4.32.2"

}

Para guardar la versión exacta en el package.json se debe agregar la bandera --save-exact y su shortcut sería -E

npm install webpack --save-dev --save-exact

npm i webpack -**D** -**E**

Esto agregará la dependencia sin el prefijo, lo que instalaría la versión exacta indicada en cada instalación.

# Creando nuestro primer bundle con Webpack

Vamos a instalar otra dependencia llamada **webpack-cli**, la API que expone webpack en forma de **CLI (Command Line Interface)** que nos va a permitir interactuar y configurar Webpack desde la terminal.

El comando webpack tiene una bandera llamada --mode que nos permite cambiar entre los modos producción y desarrollo. Recuerda que por defecto nos pone en modo producción si no la especificamos.

**Creacion de bundle con webpack**

Desde la version 4 de webpack requiere el uso de webpack cli el cual se instala como dependencia de desarrollo en nuestro proyecto

**npm install webpack-cli --save-dev**

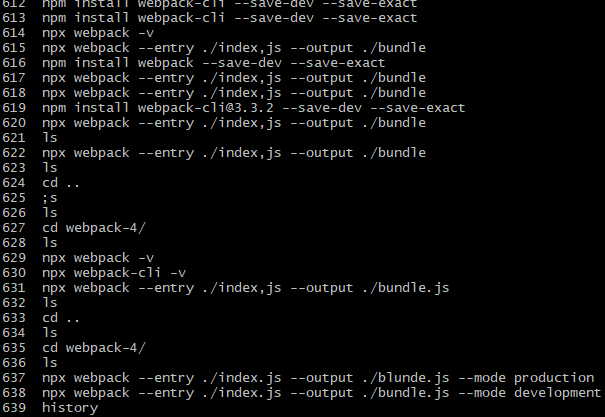
haciendo uso de npx podemos usar webpack como comando en en nuestra consola  
-npx webpack -v (nos arrojara la version instalada)  
\*\*Creacion de archivo e entrada y salida en nuestro webpack  
\*\*  
**npx webpack --entry ./index.js --output ./bundle.js --mode development**

Con este comando le estamos indicando a webpack el archivo en el que trabajaremos nuestro código para que webpack lo procese y luego le indicamos el archivo de salida del bundle el cual se utilizara en el producto final.  
con el flag de “–mode” indicamos si queremos una version de production la cual traera todo el codigo minificado o modo development.

Las **devDependencies** son aquellas que sólo utilizarás tu como desarrollador y que el público no necesitará para correr la página, estas dependencias no serán tomadas en cuenta a la hora de hacer el empaquetado final de tu página, es decir, no se incluirán en la versión de producción. Por ejemplo: ESLint  
   
Por otra parte, las **dependencies** forman parte esencial de tu página y siempre serán incluídas en la versión de producción, pues estas serán necesarias para correr la página. Por ejemplo: React.

npx webpack --entry ./index.js --output ./blunde.js npx webpack --entry ./index.js --output ./blunde.js --mode development npx webpack --entry ./index.js --output ./blunde.js --mode production





# Creando nuestro primer bundle con Webpack



Para ejecutar comandos de webpack hay diferentes caminos 🛤. Anteriormente ejecutamos un comando con NPX para transpilar un archivos .JS y generarlo especificando un OUTPUT, pero esto puede ser en la práctica algo que quite más tiempo y engorroso.

Para ello se puede crear un archivo de configuración de webpack. El nombre de este archivo en teoría puede ser cualquiera pero se recomienda llamarlo webpack.config.js.

Dentro de este archivo se especificarán las acciones que queramos que webpack haga por nosotros. El archivo de ejemplo es este:

**const** path = require(‘path’) /\*este es un modulo de node y se encarga de la gestion de rutas\*/

/\*Para exportar un modulo, se debe hacer de la forma de commonJS\*/

module.exports = {

entry: ‘./index.js’, /\*El archivo que queremos transpilar \*/

mode: ‘development’, /\* modo de ejecución \*/

output:{ /\*Configuraciones para el archivo final que generará webpack \*/

path: path.resolve(\_\_dirname), /\* La ruta en donde quedará el archivo final \*/

filename: ‘bundle.js’ /\*El nombre con el que se creará el archivo final \*/

}

}

Una vez se tenga esta configuración ⚙️ se puede llamar de dos maneras ✌🏽:

* con npx: Se puede llamar con este comando para que webpack tome el archivo de configuración y haga su proceso con base en este. El comando es

npx webpack

* con npm: Dentro del archivo .JSON, dentro del objeto scripts se puede configurar un comando para ser llamado directamente por npm. El comando que se añadirá es: build, que a su vez llamará a webpack

“scripts”:{

“**build”:** “webpack” //**build** sera el nombre que usaremos en consola

}

Para ejecutarlo basta con escribir en la consola 🤩:

npm run webpack

