What is web pack

Webpack is a dependency analyzer and module bundler.

For example, if module A requires B as a dependency, and module B requires C as a dependency, then webpack will generate a dependency map like C -> B -> A.

In practice it is much more complicated than this, but the general concept is that Webpack packages modules with complex dependency relationships into bundles.

Regarding webpack's relationship with babel: When webpack processes dependencies, it must turn everything into JavaScript because webpack works on top of JavaScript.

What is entry in webpack config?

Entry is a property that indicates webpack which module(s) / file(s) it should use to begin building out the internal dependency graph(s) of the project / application.

Where is output of webpack?

Output is a property that tells webpack where to emit / save the bundles it creates and how to name these bundled files. By default the main output file gets stored in ./dist/main. Note that, while there can be multiple entry points, only one output configuration is specified

---------------------------------------------------------------------------------------------------------------------------------

"scripts": {

"start": "webpack ",  
 “host”: “webpack serve”

},

“npm start” will create a bundled file in dist folder.  
“npm host” will create a bundled file in memory instead of creating in a dist folder and then render that on browser on a certain port number.

--------------------------------------------------------------------------------------------------------------------------------

Loaders

Webpack enables use of loaders to preprocess files. This allows you to bundle any static resource way beyond JavaScript. You can easily write your own loaders using Node.js.  
Loaders basically load files in a webpack bundle.  
Prominent Loaders are css loaders, scss loaders, image loaders, Font loaders, and much more.

Plugins

Plugins can be leveraged to perform a wider range of tasks like bundle optimization, asset management and injection of environment variables.  
Plugins are basically used for enhancing the functionality that is going to be outputted in the Webpack output folder.  
Some plugin example are :-   
HTML Webpack Plugins(this plugin will see what all build files are coming out from webpack process and add it to this index.html file i.e. Injects dependencies( bundle files) directly in HTML file),   
Environment Plugins(allow us to set up environment variables based on environment i.e. SIT,UAT or PROD)  
HTML Minimizing Plugin(allow us to minimize the bundle size)  
Min CSS Extract Plugin (Now if you remember when all of you were loading all this cases while they were just going to be getting injected into our index start in the header section. Now, from the performance perspective, this is not very good because we cannot be really caching all the css files here. Every time you are going to be loading that particular page, you also need to be pulling up all these css. And it's going to be very useful if you can extract all of this css into a separate file. And this is exactly what is going to be done by this plugin. Now, once you apply this plugin, we are going to be having another file created i.e. bundle.css and that can be loaded into our HTML)

Study about loaders and plugins from this link

https://imranhsayed.medium.com/webpack-loaders-and-plugins-e13f79fe6b32

-D flag while giving npm install command indicates that it is dev dependency. It will not be used in prod environment.

By default npm install will install prod dependencies.

In Node js every file is a module. Every Module is automatically and default enclosed in an IIFE with 5 arguments “exports, module, require, \_\_dirname, \_\_filename” as arguments.

For exporting a module use module.exports = {func1, func2}  
For importing use const {func1,fun2} = require(‘../file.js’);

Some important predefined modules in node js.  
path imported as ‘path’ , file system imported as ‘fs’ , Operating System imported as ‘os’ , events module imported as ‘events’

Important modules in fs are mkdir, writeFile,   
fs allow us to read, write, create a file.

Process.env.NODE\_ENV is used to identify in which environment our application is running.

The process.env global variable is injected by the Node at runtime for your application to use and it represents the state of the system environment your application is in when it starts.