Webpack use and install
Lu primer ke najajitem ej tani el node jota eze injjtalat. hhhino… hhhhHHHIINOO...ejjjmm.. hhhHHHSSINOOO!! ajtem futucss. Vale o no?? :D

Adveretisment: el seguent tutorial és mega cutre i bàsic però crec que per començar a entendre una mica el tema us anirà bé.

Amb el node.js instalat ja tenim el comando ***npm*** operatiu per instalar les dependencies que necessitarem.

Exercici d'exemple:

-crea una carpeta *'project'.*
-dins de *'project'* crea una carpeta 'src'
-dins de *'src'* crea un archiu '*main.js'* amb un '*alert('hola')'* per exemple.

1) Al root del projecte: ***npm init -y***

Amb aquest comando es crearà un package.json. Aquí webpack anirà guardant totes les dependencies que anem instal·lant. Més endavant veurem perquè. També aquí configurarem els comandos o les ordres que utilitzarem per dir-li a webpack que volem fer.

2) ***npm install webpack -D***

Webpack es pot instal·lar de forma global afegint **--global** en comptes de **-D**, però és millor instalar-lo sempre només al projecte per evitar conflictes de versions etc. També serveix **npm install webpack**, la **-D** significa instalar-ho en development mode i pel que diuen és la bona pràctica.

Veureu que se us ha creat una carpeta **node_modules** al projecte. Afegiu-la al **.gitignore**. Aquí està tot el que webpack necessita i tot el que anirem instal·lant via **npm**. També se us crearà un **package.lock.json** on estàn tots els mòduls dins de **node_modules**. No l'editeu, es treballa sempre al **package.json**.

3) crear archiu **webpack.config.json** al root *'project'*.

Com ja sabeu webpack és una eina per compilar moduls de codi en un únic archiu o varis. En aquest **webpack.config.json** és on li direm a webpack quins archius volem que agafi, quines eines ha d'utilitzar per transpilar aquests archius (ecmascript, sass, typescript, etc) i a on volem que crei aquest archiu nou amb tot el codi compilat. Una configuració bàsica seria la seguent:

```
const path = require('path');
module.exports = {
  mode: 'development',
  entry: './src/main.js',
  output: {
    path: path.resolve(__dirname, '/dist'),
    filename: 'bundle.js'
  },
}
```

Com veieu tenim un entrada **entry** que també pot ser un objecte amb varies entrades. Webpack agafarà els archius que vegi al **entry** i els escupirà al directori que li hem dit a **output** i amb el nom, en aquest cas, **bundle.js**.

4) Tornem al **package.json**:

Per executar els comandos de webpack els hem de definir en aquest archiu. Dins de l'objecte **scripts**.

```
"scripts": {
    "dev": "webpack --mode development",
    "production": "webpack --mode production",
    "watch": "npm run dev -- --watch"
  },
```

Aquí podem veure l'alias "production" per exemple. Si escribim a consola **npm run production** s'executarà webpack en *mode production*. Executa el comando i si tot està correcte s'et crearà una carpeta */dist* i dins un archiu *bundle.js* amb el codi compilat i minificat.

Avís: Si estem treballant amb archius *.js* o *.css*, fins que no executem **npm dev** o **npm production** no veurem els canvis al navegador. Per evitar tindre que executar aquests comandos cada cop que fem canvis als archius, tenim la opció **npm run watch**. Amb aquest comando cada cop que fem canvis webpack automaticament anirà compilant.

5) Instalar dependencies:

Webpack necessita de mòduls per transpilar els diferents codis com per exemple ES6, components react.js o vue.js, SASS, etc. Llavors segons el que utilitzem necessitarem instalar uns o altres. Posem que només utilitzarem ES6. Casi per norma sempre que instal·lem webpack instalarem el pack de Babel, ja que s'encarrega de transpilar javascript modern. Per instal·lar-lo obrim consola:

npm install -D babel-loader @babel/core @babel/preset-env babel-preset-env

Si obres el **package.json** podràs veure que a 'devDependencies' tenim aquests paquets. Aquí és on està la magia d'aquest **package.json**. Si recordes hem posat el **node_modules** a **.gitignore**. Gràcies a aquest package.json cuan algú es baixi el teu repositori, executant el comando **npm install** instal·larà tot el core de node més les dependencies que trobi a 'devDependencies' o 'dependencies' (la diferència entre aquests dos ja en parlarem). D'aquesta forma qualsevol programador tindrà tot el que necessita el projecte per funcionar via **npm install.**

6) **webpack.config.js** rules:

Ara ens toca dir-li a webpack que té que utilitzar per transpilar X archius, en aquest cas els archius *.js*. A **webpack.config.js**, dins de *module.exports* escrivim el següent:

```
module:{
                rules:[
                        {
                                test:/\.js$/,
                                exclude: /node_modules/,
                                use: {
                                        loader: 'babel-loader',
                                        options: {
                                                presets: ['@babel/preset-env']
                                        }

                                }
                        },
                                        ]
  },
```

Com pots veure obrim un objecte module i dins definim unes normes:

- **test**: miram els archius .js
- **exclude**: el que no volem que webpack miri.
- **use**: el transpilador que webpack necessita per aquest tipus d'archius.

Amb aquesta configuració bàsica ja tindriem webpack operatiu. Realment es poden fer virgueries de config per definir diferents entorns de treball etc. Però no arribu a tant. Ja anirem aprenent. A partir d'aquí és anar investigan quins moduls es necessita per cada tipu de codi i anar afegint. Si us ha quedat algun dubte truqueu al Dámaso.