Scripts NPM

DEV.F::
DESARROLLAMOS(PERSONAS);

Script

Dentro del archivo package.json, hay una sección llamada "**scripts**" que se utiliza para definir comandos personalizados.
Estos comandos pueden ejecutarse directamente desde la terminal usando **npm run <script-name>**, donde <script-name> es el nombre del script definido en package.json. npm también ofrece varios scripts predefinidos con comportamientos específicos, como start, test, y build, que pueden ejecutarse con comandos abreviados como npm start, npm test, y npm run build.

Custom Scripts

Tu puedes hacer tus propios scripts usando palabras claves, los más comunes pueden ser:

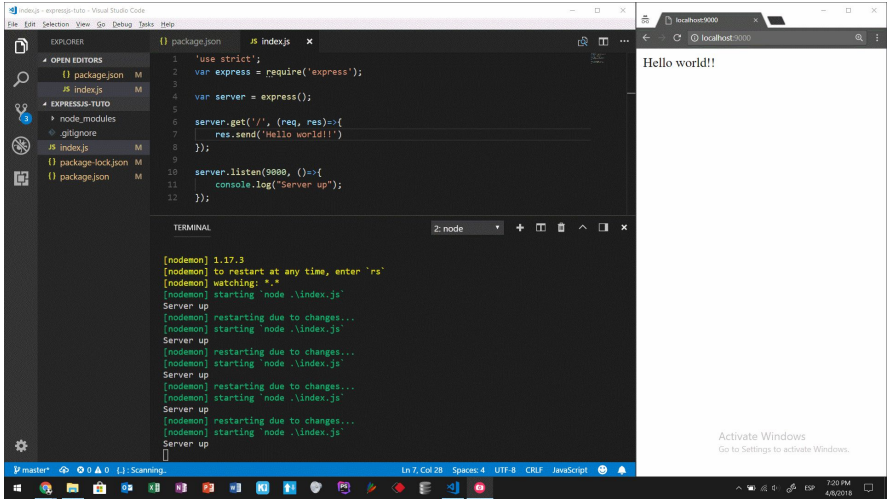
start: Utilizado para iniciar una aplicación. Por ejemplo, puede arrancar un servidor local de desarrollo.

build: Compila el código fuente en un formato optimizado para la producción.

La automatización con scripts es una técnica poderosa en el desarrollo de software moderno, permitiendo a los equipos ser más ágiles, precisos y eficientes en sus procesos de desarrollo y despliegue.

```
"scripts": {
  "start": "node app.js"
}
```

```
"scripts": {
  "build": "webpack --mode production"
}
```



Nodemon

Es una herramienta de utilidad que se ejecuta y supervisa aplicaciones Node.js para desarrolladores, haciendo el proceso de desarrollo más eficiente y productivo. Lo que hace nodemon es simple pero poderoso: monitorea los archivos del proyecto en el directorio donde se ejecuta nodemon. Si detecta algún cambio en los archivos, automáticamente reinicia la aplicación. Esto elimina la necesidad de detener y reiniciar manualmente el servidor cada vez que se realiza un cambio en el código, lo cual es un proceso tedioso y propenso a errores.

DEV.F::

DEV.F:

Actividad:

Crea un custom script que se llame dev y que a través de nodemon levante tu archivo index.js

Require vs import (ESM)

En el contexto de Node.js y el desarrollo de JavaScript en general, **require** e **import** son dos maneras distintas de incluir módulos o bibliotecas en tu proyecto. La principal diferencia entre ambos radica en que **require** es la sintaxis usada en **CommonJS** (un estándar para módulos en Node.js), mientras que **import** es utilizado en ES Modules (ESM), una sintaxis más moderna introducida en ECMAScript 6 (ES6) para la importación de módulos.

```
const axios = require('axios')
```

```
import axios from 'axios';
```

```
{
  "name": "mi-proyecto",
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "start": "node myApp.js"
  },
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Por defecto, Node.js utiliza CommonJS para la gestión de módulos. Sin embargo, desde Node.js versión 12 en adelante, se ha añadido soporte experimental para ES Modules, y este soporte ha sido mejorado significativamente en versiones posteriores.

Para utilizar ES Modules (y, por tanto, la sintaxis import y export) en tu proyecto Node.js, necesitas configurar tu package.json agregando el campo "type": "module":

DEV.F::