Node

DEV.F::
DESARROLLAMOS(PERSONAS);

Node

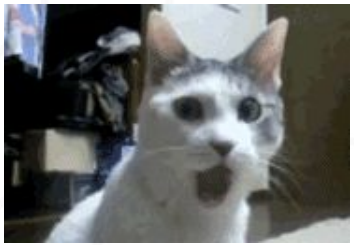
- Mayo 2009 (Ryan Lenhiart).
- Entorno en tiempo de ejecución **multiplataforma** para la **capa del servidor (no se limita a ello).**
- Basado en el motor V8 de google.
- Escrito en C++.
- Basado en módulos.
- Es **asíncrono** y trabaja con base en un **bucle de eventos**.

¿Qué puedo hacer con Node?

- **Construir API Rest**.
- **Acceder a bases de datos relacionales y no relacionales.**
- Recuperar datos de formularios HTML.
- Procesar y almacenar archivos enviados desde una página web.
- Crear, leer y escribir archivos.
- Acceder a funciones del sistema operativo y/o hardware.
- **Generar páginas dinámicas en un servidor web (server side render).**

Diferencias entre JS y Node

JavaScript	NodeJS
Lenguaje de scripting.	**Entorno de ejecución.**
Motor del navegador.	V8.
Interactúa con el DOM (Web API).	**Interactúa con el servidor.**
Libevent.	Libuv.
Ninguno de los dos tiene un API para hacer solicitudes http o utilizar temporizadores.	



Práctica

- Instalación de node y npm.
- Validación en la CLI de Node.
  - node -v
  - npm -v
- Uso de la documentación oficial.

Práctica

- Objeto Global (this).
- Uso de las funciones base de node:
  - global.
  - os.
  - file.
  - path
  - http.

Práctica

- Crear un archivo de texto con node.

Práctica

- Primer servidor en node.

Práctica

- Crear un servidor que responda páginas web estáticas.

Módulos

DEV.F:
DESARROLLAMOS(PERSONAS);

Módulos

Permiten aislar parte de nuestro código en diferentes archivos y mandarlos llamar sólo cuando los necesitamos. Existen dos formas de utilizar módulos en node:

- Common JS.
- ES6 Imports (.mjs o “type”: “module” en package.json).

Práctica

- Crear una calculadora en node, utilizando los import common y los es6 import (msj).

Npm

DEV.F::
DESARROLLAMOS(PERSONAS);

Npm

Node Package Manager es un manejador de paquetes de node (el más popular de JavaScript). Permite para compartir e instalar paquetes.

Se compone de 2 partes:

- **Un repositorio online para publicar paquetes** de software libre.
- **Una CLI para la terminal** que permite interactuar instalar, gestionar y publicar paquetes.

NOTA: Se puede considerar un gestor de dependencias de proyectos de tipo npm.

Comandos de Npm

Inicialización de un proyecto npm

- **npm init:** Inicializa una carpeta como un proyecto de npm.

Levantar un proyecto npm

- **npm start:** Es el único comando por defecto que no requiere la palabra run (npm run start). Permite “iniciar” un proyecto de node.

Comandos de Npm

Instalar/desinstalar dependencias

- npm i: Instala todas las dependencias del package.json
- npm install -g <pakage-name>: Instala un paquete globalmente.
- npm install <package-name>: Instala un paquete y lo agrega al package.json
- npm install –save <pakage-name>: Instala un paquete y lo agrega al package.json
- npm install -D <package-name>: Instala un paquete y lo agrega al package.json en la parte de devDependencies.

NOTAS

- install = i
- **Una dependencia** es un recurso o librería externa que utilizan los proyectos para funcionar.

Comandos de Npm

Gestión de dependencias

- npm uninstall <package-name>
- npm uninstall -g <package-name>
  npm search <package-name>
- npm ls
- npm update -save
- npm list
- npm list -g –depth 0
- npm outdated

Paquetes

DEV.F::
DESARROLLAMOS(PERSONAS);

Paquetes

Son módulos distribuidos en forma de librerías que resuelven alguna necesidad de desarrollo. A continuación se listan los más populares al 2022:

- npm.
- create-react-app.
- vue-cli.
- grunt-cli.
- mocha.
- react-native-cli.
- gatsby-cli.
- forever.

Scripts

DEV.F:
DESARROLLAMOS(PERSONAS);

Scripts

Son comandos propios que se pueden agregar al package.json para poderlos ejecutar con **npm run <my-comand>.**

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node index.js",
  "dev": "nodemon index.js"
},
```

Scaffold Npm

DEV.F::
DESARROLLAMOS(PERSONAS);

Estructura de proyecto npm

- **node_modules:** Carpeta donde se instalan las dependencias de un proyecto npm, normalmente está carpeta se agrega al .gitignore.
- **package.json:** Guardan las dependencias y los comandos de node.
- **package-lock.json:** Guarda un snapshot de las dependencias que se instalaron en un determinado momento.

package.json

Este archivo guarda las dependencias y los comandos de node.

- **name:** Nombre que se le asigna al proyecto, generalmente es el mismo nombre de la carpeta y el repositorio.
- **versión:** Se utiliza para conocer la versión del proyecto, el estándar para versionar es SEMVER.
- **description:** Meta data sobre lo que trata el proyecto.
- **license:** Tipo de licencia de software del proyecto.

package.json

Este archivo guarda las dependencias y los comandos de node.

- **scripts:** Para declarar comandos largos que después se pueda ejecutar con npm run <commande>.
- **devDependencies:** Son dependencias que sólo se instalan en el entorno local.
- **dependencies:** Son dependencias que se instalan en cualquier entorno (local, test, qa, preproducción y pro).

package-lock.json

- Da visibilidad de los cambios en el árbol de dependencias.
- Garantiza la conciliación de dependencias.
- Se autogenera cada que un comando modifica el node_modules o el package.json.
- Es usualmente generado por el comando npm install.
- Este archivo tiene las versiones exactas de las dependencias utilizadas por un proyecto npm.
- Debe subirse al repositorio.

Práctica

- Crear una calculadora en node, utilizando los import de es6 import (“type”: “module”).

Práctica

- Servidor de archivos multimedia con node.

Práctica

- Definir los siguientes conceptos:
  - Entorno de ejecución.
  - Manejador de paquetes.
  - Diferencia entre node y npm.
  - CLI, comando, dependencia, gestor de dependencia, dependencia de desarrollo y script.
  - Cliente y servidor.
  - API.
  - Módulo.
  - Paquete.

Práctica

- Desarrollar un proyecto node que regrese plantillas dinámicas (server side render).

Compatibilidad de ECMA

DEV.F::
DESARROLLAMOS(PERSONAS);

Compatibilidad de ECMA

Recordar que las características de ECMA no siempre están disponibles en todos los navegadores, los siguientes recursos son de utilidad para validar la compatibilidad:

- Can I use?
- Kangax

Babel

DEV.F::
DESARROLLAMOS(PERSONAS);

Babel

Es un **transpilador** (como un "compilador") para JavaScript. **Permite transformar código escrito con las últimas y novedosas características** de JavaScript (EsN) y transformarlo **en un código que sea entendido por navegadores más antiguos** (ES5).

Babel es necesario dado que la velocidad de actualización del estándar no es la misma que la velocidad de adopción de los navegadores o no siempre es retrocompatible en ocasiones se requiere transpilar.