# 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?

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

# 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 1

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

# Práctica 2

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

# Práctica 3

- Crear un archivo de texto con node.

# Práctica 4

- Primer servidor en node.

# Práctica 5

- 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 cuándo los necesitamos. Existen dos formas de utilizar módulos en node:

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

# Práctica 6

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

# Npm

DEV.F::
DESARROLLAMOS(PERSONAS);

# Npm

**Node Package Manager** o manejador de paquetes de node, es la herramienta más popular de JavaScript para compartir e instalar paquetes. Se compone de 2 partes:

- **Un repositorio online para publicar paquetes** de software libre para ser utilizados en proyectos Node.js
- **Una herramienta para la terminal (CLI)** para interactuar con dicho repositorio que ayuda a la instalación de utilidades, manejo de dependencias y la publicación de 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 para iniciar un proyecto de node (no requiere la palabra run).

# Comandos de Npm

**Instalar/desinstalar dependencias**

**Dependencia:** Son los recursos o librerías externas que utilizan los proyectos para funcionar.

- npm i
- npm install -g <pakage-name>
- npm install <package-name>
- npm install –save <pakage-name>
- npm install -D <package-name>
- npm uninstall <package-name>
- npm uninstall -g <package-name>

**NOTA:** install = i

# Comandos de Npm

**Gestión de dependencias**

- 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.

# Detalle del package.json

Este archivo guarda las dependencias y los comandos de node.

- name.
- version.
- description.
- license.
- scripts.
- **devDependencies:** Son dependencias que sólo se instalan en el entorno local.
- **dependencies:** Son dependencias que se instalan en cualquier entorno (local, test, qa, prepro y pro).

# Detalle del package-lock.json

- Este archivo tiene las versiones exactas de las dependencias utilizadas por un proyecto npm.
- No está pensado para ser leído línea por línea por los desarrolladores.
- Es usualmente generado por el comando **npm install.**

# Práctica 7

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

# Práctica 8

- Servidor de archivos multimedia con node.

# Práctica 9

- 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 10

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