Node.js

Qué es node.js

- Es un entorno de ejecución de JS que corre sobre Chrome V8
- Open source
- Cross-plataforma
- Lo mantiene OpenJS Foundation
- Permite ejecutar JS del lado del servidor
- No tiene librerías ni APIs para interactuar con el navegador sino con la máquina

¿Para qué se usa?

- Procesamiento de datos en tiempo real
- Herramientas de línea de comando
- Programación del lado del servidor: servidores web, APIs, microservicios, etc

Lo básico

- Instalar localmente
  - nvm
- Docker
  - docker run -it -d --name myNode node[:tag]
- Ejecutar consola (Read Eval Print Loop, REPL)
  - Es como la consola del navegador
  - Permite ejecutar los comandos que aprendimos anteriormente
- node <path_file>
  - Ejecutar un script, tal como el interprete de python

Módulo http

- Modulo http → import http from “http”
  - Parte de la librería estandard de node
  - “http” sin “./” porque no está en la carpeta actual sino en un modulo externo
- let server = http.createServer([options][,requestListener]); → Crea un servidor HTTP.
  - RequestListener → (request, response) => {}
- server.listen(port[,callback]); → El servidor escucha en un puerto y se llama al callback. El servidor sigue escuchando aún cuando el callback termine inmediatamente.

Módulo http

- Manejando peticiones (respuestas)
  - response.writeHead(statudCode[, statusMessage][, headers])
  - response.write(statudCode[, statusMessage][, headers])
  - response.end([data[,encoding][,callback]])

- Mucho más en https://nodejs.org/api/http.html

Módulos externos

- Npm: Node Package Manager
  - Gestiona dependencias para los proyectos de node.
  - Usa un archivo package.json/package-lock.json que contiene metadatos
- npm init [-y] → “type”: “module” (ES6); “commonjs” (por defecto)
- npm install <package>
  - Busca el paquete en el repositorio remoto y lo instala como dependencia del proyecto
  - Node Package Registry (https://www.npmjs.com/)
  - node_modules

Express

- Framework web para node (https://expressjs.com/en/5x/api.html)
- Rápido y minimalista
  - npm intall express –save
- `app.METHOD(PATH, HANDLER)`
  - **app.get(path, callback [, callback …])**
- **app.listen([port[, host[, backlog]]][, callback])**
- **app.all(path, handler) → todos los métodos**
- **Path = ‘*’ → cualquier ruta**

Express

- req.params
  - app.get(‘/users/:id’)
    - curl **-i** “localhost:8080/users/**10**” → {id: 10}
- req.query
  - app.get(‘/users’)
    - Curl **-i** “localhost:8080/users?a**=10&b=Carlos**”→{a: 10, b: “Carlos”}
- node script.js
  - Commonjs → const nombre = require(‘modulo’)
  - ES6 → import {nombre} from ‘modulo’;
  - Package.json → type: “module” → ES6

Express - Middlewares

- Función que tiene acceso a la petición y respuesta HTTP
  - Pueden terminar la petición
  - Pueden pasar el control al siguiente middleware (nex())
  - Pueden modificar la petición y/o la respuesta
  - Se ejecutan en el orden en que se agregan a la aplicación
- Nativos:
  - Express.static(), express.json()
- De terceros
  - Cookie-parser, body-parser, morgan y muchos más

Express - Middlewares

- **app.use(middleware);**
  - **(req, res, next) => {**

```
// middleware function implementation
next();
```

}

Separa lógica – primer aproximación

- app.js
  - Express()
  - Importa módulos (router, middleware, etc)
- router.js (mjs)
  - Funciones que gestionan los llamados
- middleware.js (mjs)
  - Middlewares propios

Ejecutar aplicacion

- node [--watch] app.js
  - --watch permite recargar la aplicación automáticamente cuando hay cambios en el código