

# Node.JS

Introducción

# Conceptos Iniciales

- Node.js es un entorno de ejecución para JavaScript.
- Puede usarse para desarrollar cualquier tipo de aplicación pero su nicho en el mercado es principalmente para el desarrollo de Back-End y generación de APIs REST.
- Está internamente construido con V8, que es el mismo motor de ejecución que usa Chrome.
- V8 está desarrollado en C++ y nos permite acceder a muchas funcionalidades del sistema operativo.

# Diferencias con JS para el navegador

- Tradicionalmente JS se ha usado para desarrollo de Front-End, siendo interpretado por el navegador. En este caso el DOM se convierte en el framework principal del trabajo.
- Cuando usamos Node.js, en lugar de tener acceso al DOM, se nos ofrece una API mucho más extensa (a través de V8) que nos da la posibilidad de acceder a muchas funcionalidades del sistema operativo y construir aplicaciones tradicionales.
- Enlace a la API oficial de Node.js: <https://nodejs.org/dist/latest-v16.x/docs/api/>

# Ventajas de usar Node.js

- En lugar de utilizar es esquema tradicional de hilos para la programación de aplicaciones de red, Node.js está orientado a la utilización de eventos asíncronos.
- El programador no tiene que gestionar manualmente los hilos y puede olvidarse de las situaciones de bloqueo que suelen producirse en otras arquitecturas más tradicionales.
- Las aplicaciones de red hechas con Node.js son fácilmente escalables.
- Además Node.js nos ofrece la herramienta npm para la gestión de las dependencias y librerías externas. Es una herramienta versátil que ya hemos visto en clase que utilizan otros frameworks como Angular.

# Escritura y ejecución de código

- Para escribir un programa Node.js usamos la sintaxis JS6 vista el año pasado.
- Tendremos un archivo .js inicial donde escribiremos nuestro código, al igual que el curso pasado. La diferencia es que ahora tendremos a nuestra disposición muchas librerías oficiales y de terceros.
- La principal diferencia en la organización del código estará en la posibilidad de crear nuestras propias librerías, tal y como se explica en la siguiente diapositiva.
- Para ejecutar la aplicación usamos el comando “node”, seguido del nombre del archivo principal.

# Creación y uso de librerías

- Tenemos tres tipos de librerías o módulos: por defecto, propias y de terceros.
- Las librerías oficiales se importan con la función require().
- Las librerías propias se crean usando la función module.exports
- Las librerías de terceros se instalan primero con npm y luego se importan de la forma habitual. Estas dependencias internas se almacenan en el archivo package.json para posteriores reinstalaciones.

# Ejemplos de Clase

- En primer lugar se ha puesto el ejemplo de una aplicación tradicional de consola para la creación de notas. En ella se puede practicar la importación y el estudio de la documentación de librerías, así como entender que el JS del año pasado es suficiente para desarrollar este tipo de aplicaciones.
- A continuación se ha demostrado con el ejemplo del módulo `request()` que Node.JS puede usarse como “front-end” para la consulta de APIs externas.
- Por último se ha comentado que Express es un famoso módulo externo que nos permite convertir nuestra aplicación en un back-end y elaborar de forma sencilla una API REST.