Introduction to NodeJs Course

1. ¿Qué es NodeJs?

https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

NodeJs es un runtime de Javascript basado en el motor V8 de Google, que es el encargado de Javascript en los navegadores Chromium. Se podría decir que NodeJs es una "versión diferente" de Javascript, esta construido sobre Javascript. Añade características nuevas y pierde algunas que podemos ejecutar en el navegador. El concepto más importante es que te permite ejecutar Javascript en cualquier computadora, como servidores.

El motor V8 básicamente lo que hace es tomar el código Javascript y lo compila a Machine Code. V8 esta escrito en C++.

NodeJs toma ese V8 y le añade nuevas características como por ejemplo manejar archivos locales, abrir, leer y eliminar archivos. Todos estos no son posibles en el navegador. NodeJs no corre en el navegador, el navegador utiliza Vanilla V8.

2. Primera aplicación con NodeJs

https://nodejs.dev/learn/run-nodejs-scripts-from-the-command-line

https://nodejs.org/api/fs.html

Con la nomenclatura "node nombre.js" vamos a ejecutar código Javascript desde una terminal de comandos.

El FileSystem lo importamos con la palabra reservada requiere('NOMBRE'); 'fs' Es el nombre del módulo File System y Requiere es una manera de importar una funcionalidad que nos provee el núcleo de NodeJs. Todo esto se verá más adelante pero para tener una idea de que vamos a hacer en la primera app.

FileSystem tiene un método llamado writeFileSync(PATH, CONTENIDO) que escribe un archivo en nuestro disco duro. El primer parámetro que recibe es el PATH con el nombre del archivo y el segundo es el contenido de este archivo.

3. Entendiendo el rol y el uso de NodeJs

Usualmente lo que haremos en este curso es utilizar NodeJs para escribir código del lado del servidor.

El cliente va a tener en su navegador HTML, CSS y Javascript.

El cliente va a generar una petición que será enviada al servidor. Nuestro servidor es una computadora conectada a la internet con una IP y un dominio asociado. Este servidor va a ejecutar algún codigo que hará algo con la petición del cliente y retornará una respuesta. Normalmente, no siempre, esa respuesta es un archivo HTML que el navegador puede desplegar, no necesariamente es solo HTML, puede ser también CSS y JS.

En el servidor típicamente hacemos tareas que no se pueden o no queremos hacer en un navegador, ya sea por performance o por seguridad. Por ejemplo, el servidor se conecta a una base de datos para traer y guardar información, autenticamos datos para hacer el sitio más seguro (El cliente puede cambiar el HTML de un form desde las herramientas de inspección, en el lado del servidor tendremos que volver a chequear que todo este correcto ya que de este lado el cliente no tiene acceso), validación de inputs para saber si el usuario ingreso una dirección de mail correcta.

Por supuesto también tenemos nuestra lógica de negocio en el servidor. Todo lo que el cliente no debería ver y que lleva demasiado tiempo ejecutarlo en el navegador, donde obviamente tenemos que ofrecer una experiencia de usuario rápida.

Muchas de las funciones que nos da NodeJs en el servidor trabajan indirectamente con el cliente a través de las peticiones que este realiza, el acceso directo no está disponible.

En este curso utilizaremos NodeJs para escribir código en el servidor que devuelva datos a nuestros usuarios para que puedan trabajar.

NodeJs no se limita solamente al servidor. Es un runtime de JavaScript (Puede servir para ejecutar JS en una terminal, para el servidor como se mencionó más arriba, scripts, herramientas de construcción, etc).

Generalmente y donde es más popular NodeJs es el uso de éste en el desarrollo web y código del lado del servidor.

Lo utilizamos para correr un servidor, y a diferencia de PHP, no solo escribimos código que corre en el servidor, sino que también nosotros escribimos el servidor. En PHP tenemos herramientas como Apache o Nginx que corre el servidor que escucha las peticiones y ejecuta código PHP. Acá NodeJs hace ambas, escucha y después hace cualquier otra cosa que tiene nuestro código.

También se utiliza para la lógica de negocios, no solamente para escuchar peticiones sino también para trabajar con la información de la petición, trabajar con archivos, base de datos, etc.

Otra cosa que hace NodeJs es manejar respuestas no solo de las peticiones sino también aprenderemos a utilizar NodeJs para enviar información devuelta a nuestro cliente, ya sea páginas HTML, páginas HTML con contenido dinámico o información en formato JSON, XML o simplemente archivos.

Básicamente esto es lo que veremos en profundidad en el curso.

3. Tópicos que veremos en este curso

1. Introducción a NodeJs
2. Repaso de JavaScript (Opcional)
3. Nociones básicas de NodeJs
4. Desarrollo eficiente
5. Framework Express.js
6. Motores de Templates
7. Patrón Modelo-Vista-Controlador
8. Concepto de Rutas y Modelos
9. Node + SQL (MySQL)
10. Uso de Sequialize
11. Node + NoSQL (MongoDB)
12. Uso de Mongoose
13. Sesiones y Cookies
14. Autenticación
15. Enviar E-Mails
16. Autenticación avanzada y autorizaciones
17. Validación de los inputs de usuarios
18. Manejo de errores
19. Subida y descarga de archivos
20. Paginación
21. Peticiones asíncronas
22. Manejo de pagos (PayPal y similares)
23. Básicos de REST API's
24. Características avanzadas de REST API's
25. Uso de async-await
26. Websockets y Socket.io
27. GraphQL API's
28. Despliegue de aplicaciones

29. Funcionalidades de NodeJs más allá del servidor
30. NodeJs + TypeScript y DenoJs

4. REPL vs Uso de archivos

REPL significa Read (Read user input), Eval (Evalute user input), Print (Output result), Loop (Return and wait for new input). Este tipo de ejecución de NodeJs se logra desde una terminal, donde podemos hacer operaciones simples, importar archivos (como por ejemplo trabajar con el Fiel System), escribir código JS sin la necesidad de guardarlo en un archivo, etc. Las líneas de ejecución no son independientes unas de otras y se guardan de manera momentánea. Esta manera de trabajar termina una vez salimos de la terminal.

La alternativa el REPL es ejecutar código guardado en un archivo JS como haremos en el resto del curso. Ejecutar archivo es lo que se utiliza en aplicaciones reales, tenemos una secuencia predecible de pasos, podemos exportar, cambiar y crear código, etc.