Aplicación de Node

¿Cómo resuelve peticiones asíncronas y síncronas?

¿Cómo trabaja cuando llegan muchas peticiones?

Node es un lenguaje NON-Blocking I/O.

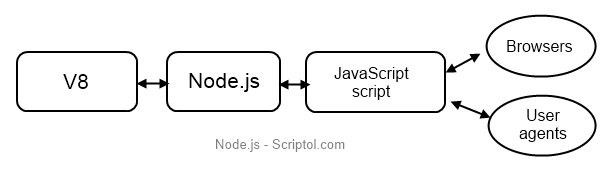
**¿Qué es Node ?**

Pensemos en Javascript que ejecutamos en el navegador Web, pero llega Node el cual lo lleva al nivel del servidor y ya no únicamente en el cliente lo que lo vuelve un lenguaje de Backend.

* Es un lenguaje de Backend.
* Nos da acceso al sistema de archivos del equipo.
* Información sobre el sistema operativo.
* Procesos que está ejecutando la computadora

**¿Dónde corre Node?**

Node corre sobre el motor V8 de Google, escrito en C++, lo tiene Google Chrome, engend de JavaScript de alto desempeño, traduce el código JavaScript al lenguaje de máquina y lo hace tan eficiente que por eso Node es muy eficiente.



**¿Qué puedo hacer con Node?**

* Uso de sockets para una comunicación entre cliente – servidor o servidor-cliente. Parecido a un chat, tags de Facebook con notificaciones.
* Manejo de archivos (Filesystem del equipo).
* Manejo de cargas simultáneas. Por ejemplo, suponga que una persona sube 10 archivos, en el momento en que un archivo se sube, ustedes pueden empezar a ejecutar algún tipo de trabajo sin esperar a que los otros 9 archivos se terminen de subir.
* Servidores locales y remotos con información en tiempo real.
* Conexiones a bases de datos.
* Creación de servicios REST en segundos.
* Podemos crear nuestras propias APIs o servicios que pueden ser consumidos por cualquier persona a nivel mundial (app móvil o de escritorio o ambas, cualquier petición a nuestro servidor), no es demás mencionar a express librería de Node.

**¿Por qué es tan popular Node?**

* Entradas y salidas que no realizan bloqueos al servidor, entiendo que sigue funcionando.
* Es sumamente rápido y fácil de configurar.
* Mas de 470 mil paquetes disponibles. (ecosistemas opensource más grande del mundo).
* Si saben JavaScript ya conocen la mayor parte de Node.

**¿Quiénes lo usan?**

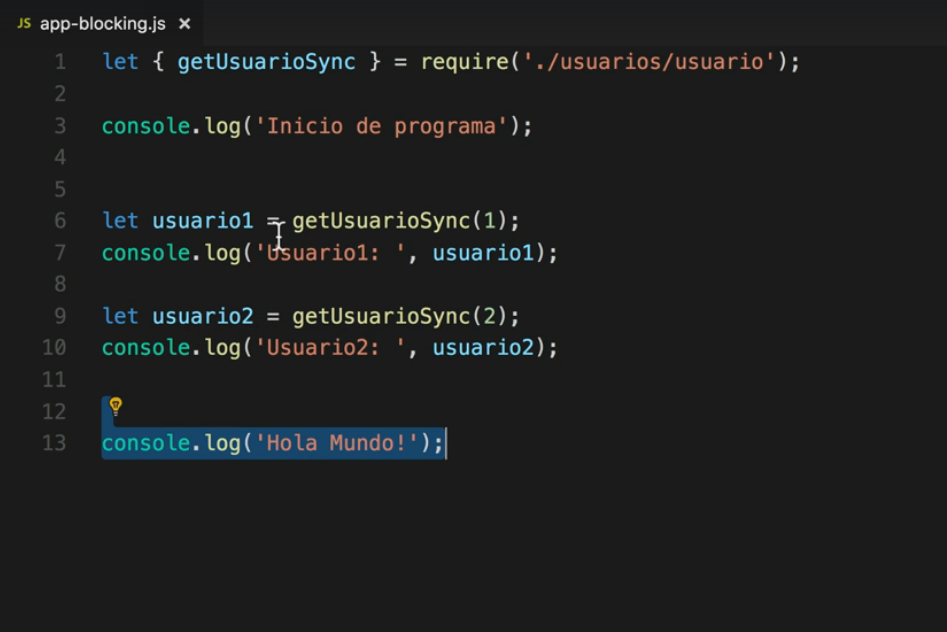


**Pequeña descripción…**

Node utiliza una conducción por eventos, usa un modelo de entradas y salidas que no se bloquean lo cual lo hace sumamente ligero y eficiente, npm (node package manager) es el ecosistema de librerías opensource más grande del mundo.

**Non Blocking I/O vs Blocking I/O**

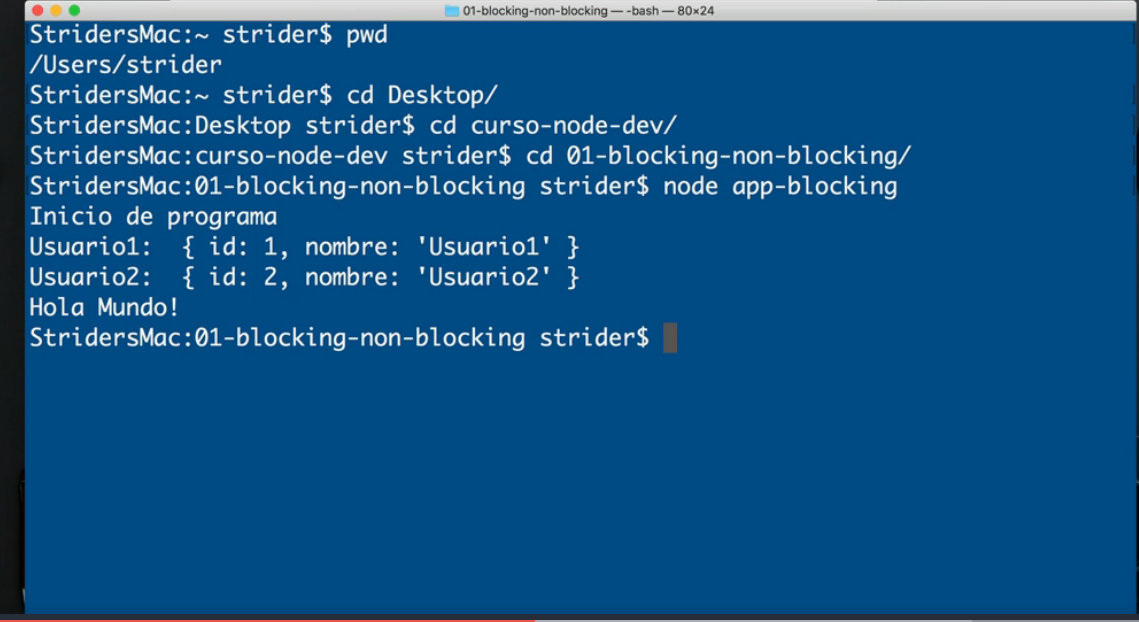
getUsuarioSync(1) -> Demora 3 segundos.





En el código NON-Blocking se utiliza lo que se conoce como CALLBACK.

Ponga a ejecutar lo que se conoce como GetUsuario1 luego esa información puede demorar cierta cantidad de segundos en responder pero Node sigue ejecutando la siguiente instrucción, ahora GetUsuario2 y pone otra tarea en ejecución, imprime Hola Mundo y el programa sigue corriendo porque existe 2 procesos que todavía no han respondido que en este caso es GetUsuario1 y GetUsuario2.





Ya tiene el Hola Mundo, porque funciona más rápido.

Aunque demora 3 segundos obtener información de un usuario node continúa ejecutando el programa y las instrucciones de buscar usuario las hacen en forma paralela o simultánea.

En el código NON-Blocking se utiliza lo que se conoce como CALLBACK.

Esto nos va a permitir ejecutar muchos procesos de forma simultánea, con resultados bastante rápidos.

Ejecutamos el app.js con node app.js:

let nombre = "Juan Pablo Villegas Gomez";

//console.log('Hola' + nombre);

//Acostumbrarse a usar Templates literales. Podemos hacer una inyeccion con el signo de dólar

///Son sinónimos, ojo con el backtick ``

console.log(`Hola ${ nombre }`);

let a = 10;

Cualquier cosa en javascript , cualquier función o cualquier método siempre regresa algo, la función de asignación en este caso regresará undefined, no hay un valor definido en el retorno.

**Ciclo de vida de un proceso en Node.**

Cuando escribimos un comando del tipo node, node crea un proceso main().



Entra let nombre = ‘Fernando’;

A la pila de procesos y la elimina de una.

Entra console.log (‘Hola’ + nombre);

A la pila de procesos y la elimina.

Con las otras instrucciones de app1 pasa lo mismo.

Cuando no tenemos nada en la fila de procesos, lo que hace Node es terminar el MAIN.

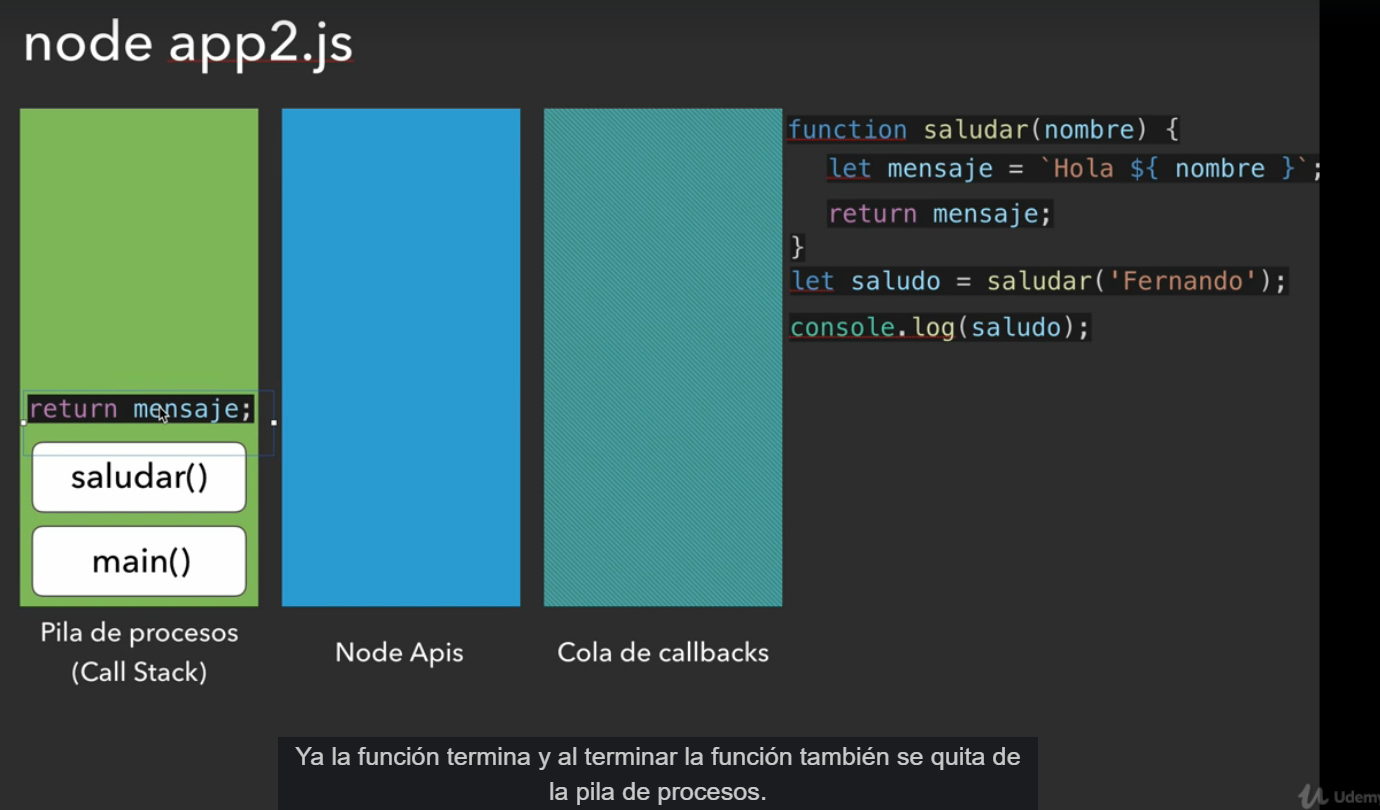
Para app2.

Registra la función saludar en la pila de procesos.

Ahora cuando se llama saludar pasaría lo siguiente:



Se empezará a ejecutar el contenido de la función y se agrega a la pila de procesos.



Cuando se realiza el return mensaje; la función termina y se quita de la pila de procesos.

Miremos a app3.js:



El settimeOut lo registra, no lo debe de ejecutar todavía puesto que lo hará en 3 segundos. Es nuestra primera tarea **ASÍNCRONA**, no la elimina porque todavía no está resuelta.

Despues lo traslada a la caja de Node Apis. Es un lugar donde Node está pendiente de esos procesos, esta esperando respuestas y esta esperando a que tareas asíncronas terminen