«Talento Tech»

Back-End

Node JS

Clase 02

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 2 - Introducción a Node JS

Temario:

1. **Fundamentos:**
   - ¿Qué es Node JS?
   - Características principales
2. **Diferencias con Javascript en el browser:**
   - Ejecutar JavaScript fuera del navegador
3. **Core de Node Js**:
   - Funcionamiento
   - Modelo de single-threading y la diferencia con modelos multi-threading
4. **Introducción Práctica:**
   - Probamos Node JS

Objetivos de la Clase

En esta clase comprenderemos los fundamentos de Node.js como entorno de ejecución de JavaScript fuera del navegador, explorando sus características principales y cómo se diferencia del uso tradicional de JavaScript en el browser. Además, se busca familiarizarse con el core de Node.js, entendiendo su funcionamiento, el modelo de single-threading y su contraste con modelos multi-threading. Finalmente, se realizará una introducción práctica para probar Node.js y adquirir experiencia inicial con su ejecución y uso.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Fundamentos.

¿Qué es Node JS?

Según el sitio oficial, “Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O (entrada y salida) de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.”

Más allá de eso, vale la pena señalar que el creador de Node.js, Ryan Dahl fue encargado de crear sitios web en tiempo real con función de inserción, “inspirado por aplicaciones como Gmail”. En Node.js, dió a los desarrolladores una herramienta para trabajar en el paradigma no-bloqueante, event-driven I/O. Por ende, Node.js también se destaca en aplicaciones web de tiempo real empleando la tecnología push a través de Websockets.

La idea principal de Node.js es permanecer ligero y eficiente frente al uso intensivo de datos en tiempo real de las aplicaciones que se ejecutan en dispositivos distribuidos. Por ejemplo, usar Node.js para operaciones intensivas de CPU, como el cálculo pesado, anulará casi todas sus ventajas. Donde Node REALMENTE destaca es en la construcción rápida y escalable de aplicaciones de red, debido a que es capaz de manejar un gran número de conexiones simultáneas con alto rendimiento, lo que equivale a una alta escalabilidad.

A partir de esta definición, algunos **usos correctos de Node.js** podrían ser los siguientes:

1. Aplicaciones de Chat en Tiempo Real

- **Descripción:** Chats como los de soporte técnico en vivo, aplicaciones de mensajería instantánea o sistemas de chat en equipo.
- **Fundamento:** Node.js, junto con WebSocket, permite mantener conexiones persistentes entre el cliente y el servidor para actualizar mensajes sin la necesidad de refrescar la página.
- **Casos de uso:** Una aplicación de mensajería en la que los usuarios reciben mensajes instantáneamente a medida que se envían.

2. APIs RESTful Escalables

- **Descripción:** APIs que sirven datos para aplicaciones front-end o móviles.
- **Fundamento:** Su modelo asíncrono permite manejar múltiples solicitudes simultáneamente sin bloquear el proceso, siendo ideal para APIs de alto tráfico.
- **Casos de uso:** Un backend que alimenta un sistema de reservas de vuelos con miles de usuarios concurrentes.

3. Streaming de Datos

- **Descripción:** Sistemas donde los datos se procesan en tiempo real mientras llegan, como transmisiones de video o música.
- **Fundamento:** Node.js puede procesar grandes volúmenes de datos de forma incremental gracias a su soporte nativo para Streams.
- **Casos de uso:** Una plataforma de transmisión de música donde los usuarios escuchan mientras se descarga el audio.

4. Aplicaciones Colaborativas en Tiempo Real

- **Descripción:** Herramientas de colaboración como editores de texto compartidos o tableros de tareas en equipo.
- **Fundamento:** Node.js permite mantener un estado sincronizado entre varios usuarios mediante eventos y sockets.
- **Casos de uso:** Un editor de texto como Google Docs, donde varios usuarios pueden escribir y ver los cambios en tiempo real.

5. Sistemas de Notificaciones Push

- **Descripción:** Servicios de notificaciones en aplicaciones web o móviles, como alertas de eventos o actualizaciones.
- **Fundamento:** Los WebSockets en Node.js pueden manejar notificaciones en tiempo real de manera eficiente.
- **Casos de uso:** Un sistema de alertas meteorológicas que notifica cambios de clima a miles de usuarios.

6. Portales de Juegos Multijugador

- **Descripción:** Juegos que requieren sincronización constante entre varios jugadores.
- **Fundamento:** Su arquitectura orientada a eventos facilita la gestión de múltiples conexiones simultáneas en tiempo real.
- **Casos de uso:** Un juego de cartas en línea donde los jugadores ven las jugadas en tiempo real.

7. Dashboards de Monitoreo en Tiempo Real

- **Descripción:** Herramientas para observar métricas en vivo, como estadísticas de usuarios o rendimiento de sistemas.
- **Fundamento:** Node.js permite actualizar datos dinámicamente en tiempo real, mejorando la experiencia del usuario.
- **Casos de uso:** Un panel de control que monitorea el tráfico en una página web en tiempo real.

Estos ejemplos, sin dudas, aprovechan las principales características de Node.js: su arquitectura no bloqueante, orientación a eventos y escalabilidad en el manejo de I/O.

Características principales

Arquitectura Orientada a Eventos y No Bloqueante

- Modelo basado en eventos que maneja múltiples solicitudes sin bloquear otras tareas.
- Manejo eficiente de la concurrencia, ideal para aplicaciones en tiempo real

Basado en el Motor V8 de Google

- Convierte el código JavaScript en código máquina altamente optimizado para una ejecución rápida.
- Velocidad comparable a lenguajes compilados, ideal para aplicaciones de alto rendimiento

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Modelo de Ejecución Single-Threaded

- Usa un solo hilo para la ejecución principal, delegando tareas intensivas a un pool de threads en segundo plano.
- Ahorra memoria y evita problemas de bloqueo de hilos

Entorno Multiplataforma

- Compatible con Windows, macOS, Linux, etc., sin necesidad de ajustes en el código.
- Flexibilidad para desarrollar aplicaciones que funcionan en múltiples plataforma

Soporte para Programación Asíncrona

- Utiliza callbacks, Promises y async/await para manejar tareas asíncronas de manera eficiente.
- Permite escribir código limpio y escalable para operaciones simultánea

Capacidad de Tiempo Real (Real-Time)

- Facilita la comunicación bidireccional en tiempo real entre cliente y servidor mediante tecnologías como WebSockets.
- Perfecto para chats en vivo, sistemas de notificaciones o paneles de monitoreo en tiempo real.

Ligereza y Eficiencia

- Diseñado para ser minimalista, cargando solo lo esencial y permitiendo agregar características según se necesiten.
- Uso eficiente de recursos, ideal para dispositivos con capacidades limitadas.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Alta Escalabilidad

- Maneja un gran número de conexiones simultáneas mediante su modelo no bloqueante y permite clonar procesos para maximizar el uso de múltiples núcleos de CPU.
- Ideal para aplicaciones con alto tráfico o que requieren escalabilidad horizontal.

Diferencias con Javascript en el browser

Ejecutar Javascript fuera del navegador

Node.js nos permite utilizar JavaScript *"out of the box"*, es decir, fuera del navegador, expandiendo su alcance al entorno del servidor. Esto significa que podemos ejecutar JavaScript como un lenguaje de comandos, capaz de dar instrucciones directamente a una computadora.

Con este entorno, es posible:

- Trabajar con archivos del sistema.
- Acceder y gestionar ubicaciones en memoria.
- Crear programas de línea de comandos (CLI).
- Configurar servidores web para almacenar y compartir archivos públicos.
- Conectar aplicaciones a servidores de bases de datos.
- Implementar muchas otras funcionalidades propias de un ecosistema de servidor.

Cabe mencionar que todas estas ventajas traen consigo algunas consecuencias, ya que al salir de un entorno de navegador perderemos acceso a herramientas cruciales de la web como el **manejo del localStorage**, la capacidad de **manipular el DOM** y la **utilización de escuchadores de eventos o event Listeners** ya que son características principales que suceden únicamente cuando javascript es ejecutado en un navegador.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Conocer ambos entornos de ejecución de JavaScript es una gran ventaja, especialmente para quienes están aprendiendo programación. Permite una clara comprensión de la división de tareas entre el **frontend** (interfaz del usuario) y el **backend** (lógica del servidor). Además, facilita el uso de un único lenguaje para ambos mundos, simplificando el desarrollo full-stack.

Node.js no solo amplía las capacidades de JavaScript, sino que también refuerza su posición como una herramienta versátil para construir aplicaciones modernas y escalables.

Core de Node Js.

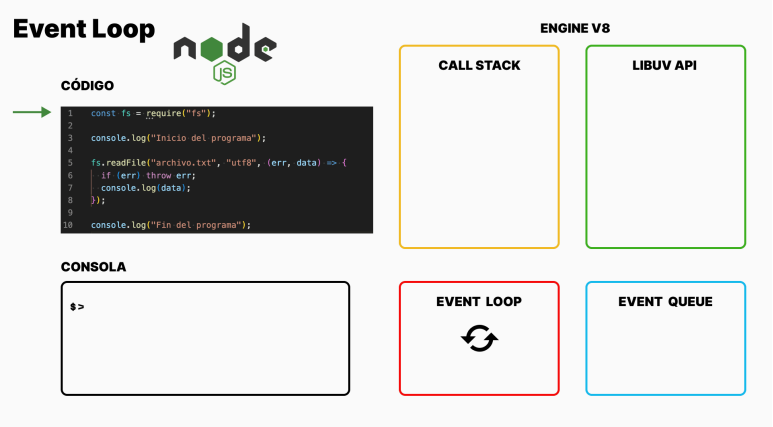
Funcionamiento

El **core de Node.js** se refiere a su núcleo funcional, diseñado para ser ligero y eficiente. Incluye un conjunto de módulos integrados que ofrecen herramientas esenciales para el desarrollo, como la manipulación de archivos, la creación de servidores HTTP y la gestión de streams de datos, entre otros. Estos módulos están optimizados y directamente integrados en Node.js, eliminando la necesidad de instalar dependencias adicionales para tareas básicas.

Algunas características clave del core de Node.js:

- **Módulos Integrados:** Herramientas como `fs` (sistema de archivos), `http` (servidores web), y `stream` (manejo de datos en tiempo real).
- **Event Loop:** Una estructura central que permite gestionar tareas asíncronas, como el manejo de múltiples solicitudes o el procesamiento de eventos.
- **V8 Engine:** El motor de JavaScript de Google que convierte código JavaScript en código máquina, garantizando alto rendimiento.

El diseño minimalista del core permite a los desarrolladores construir aplicaciones agregando solo los módulos necesarios, manteniendo el entorno ágil y eficiente.



Modelo de single-threading vs modelo multi-threading.

Node.js se distingue por su modelo de single-threading basado en un event loop, en contraste con los modelos tradicionales de multi-threading usados en otros entornos.

En el modelo single-threading se utiliza un único hilo principal para ejecutar el código de la aplicación. Este hilo se basa en un bucle de eventos (event loop) que delega tareas largas o bloqueantes, como operaciones de I/O (lectura/escritura en archivos o bases de datos), a un pool de threads en segundo plano. Una vez completadas, los resultados se devuelven al hilo principal mediante callbacks o promesas.

Las principales ventajas de este modelo son:

- **Escalabilidad:** Maneja miles de conexiones concurrentes con menos recursos.
- **Simplicidad:** El uso de un solo hilo elimina problemas de sincronización, como *deadlocks*.

Mientras que por otro lado, no es eficiente para tareas intensivas de CPU (p. ej., cálculos complejos), ya que estas pueden bloquear el hilo principal.

Por otro lado en el modelo Multi-Threading, considerado como el modelo tradicional, múltiples hilos ejecutan tareas simultáneamente, compartiendo recursos del sistema. Cada solicitud puede asignarse a un hilo separado, y los hilos pueden comunicarse entre sí según sea necesario.

Las principales ventajas de este modelo son:

- Es excelente para operaciones intensivas de CPU, ya que los cálculos pueden dividirse entre múltiples hilos.
- Permite aprovechar los núcleos de CPU en sistemas multiprocesador.

En contraposición, puede resultar complejo, ya que los desarrolladores deben gestionar problemas como condiciones de carrera (*race conditions*) y bloqueos de hilos (*thread-locking*). Además, crear y mantener múltiples hilos requiere más memoria y CPU.

Comparación Práctica

Aspecto	Single-Threading (Node.js)	Multi-Threading
Uso principal	Operaciones I/O, aplicaciones en tiempo real	Tareas intensivas de CPU, procesamiento paralelo
Escalabilidad	Alta, con bajo consumo de recursos	Limitada por la cantidad de hilos y recursos disponibles
Complejidad	Baja, sin problemas de sincronización	Alta, requiere gestión de hilos y comunicación
Manejo de concurrencia	Event loop y delegación a un pool de threads	Multiplicación de hilos

En conclusión, Node.js destaca en aplicaciones de red y tiempo real debido a su diseño single-threaded no bloqueante, mientras que el modelo multi-threading es más adecuado para aplicaciones que requieren procesamiento intensivo.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Enfoque Inmersivo.

En este curso utilizamos el enfoque inmersivo para que puedan experimentar los roles más solicitados en el mercado. Este curso va más allá de la enseñanza de programar; su objetivo es prepararlos para enfrentarse a problemas del mundo real en entornos laborales.

Este enfoque nos ayuda a:

- Conectar con situaciones reales.
- Desarrollar pensamiento crítico.
- Hacer el aprendizaje más significativo.
- Vivir la experiencia de un profesional.

Cada clase en el curso será como un capítulo de una historia. Ustedes serán los protagonistas, y su misión será utilizar las herramientas brindadas en las clases para superar los retos que se les presenten.

¡Bienvenidos a TechLab!

En **TechLab**, convertimos ideas en herramientas digitales innovadoras y confiables, ofreciendo servicios de desarrollo backend con un enfoque en la calidad y eficiencia. Nuestro compromiso es desarrollar soluciones que optimicen procesos y potencien negocios, combinando creatividad, tecnología de punta y un compromiso absoluto con la satisfacción de nuestros clientes. Tu éxito es nuestra prioridad, y nuestras soluciones están diseñadas para marcar la diferencia.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Equipo TechLab:

Silvia

Product Owner

Luis
Diseñador UX UI

Matias
Desarrollador

Sabrina
Desarrolladora

Situación Inicial en TechLab

¡Bienvenido a **TechLab**! Tenemos un proyecto emocionante entre manos: un cliente necesita una aplicación backend robusta para gestionar su tienda en línea de manera eficiente. Pero antes de que puedas unirte a nuestro equipo y formar parte de este desafío, deberás demostrar que tienes lo necesario para ser uno de nosotros.

Tu misión comienza aquí. Antes de que Matías y Sabrina, nuestros desarrolladores estrella, puedan trabajar contigo, tendrás que superar una serie de retos diseñados para poner a prueba tus habilidades.

Ejercicio Práctico

Primer Desafío: Rompe el Hielo con Node.js

Tu travesía en **TechLab** comienza con los cimientos de toda gran aplicación backend: comprender y utilizar Node.js. Matías te envía un correo que dice:

Buenos Aires aprende
Agencia de Habilidades para el Futuro

“Antes de enfrentarte a los retos importantes, debes familiarizarte con las herramientas que usarás. Empecemos con algo básico”.

Tu misión es sencilla pero crucial:

1. Crea un archivo llamado `index.js` y escribe en él el siguiente código:
   ```
   console.log(“Hola desde Node JS”);
   ```
2. Luego, abre tu terminal y demuestra que puedes ejecutarlo correctamente con el comando:
   ```
   node index.js
   ```

Si lo haces bien, verás un cálido saludo desde la consola, indicándote que estás listo para avanzar. Pero no te confíes, este es solo el primer paso hacia desafíos más grandes. ¿Listo para mostrar que puedes con lo básico? ¡Adelante!

Reflexión Final

¡Felicitaciones! Has configurado tu entorno de desarrollo, creado un proyecto y ejecutado tu primer programa en NodeJs. Este es el primer paso fundamental para el éxito de nuestro proyecto en `TechLab`. A partir de ahora, estaremos listos para enfrentar los desafíos que se presenten, desarrollando una API robusta y eficiente para nuestro cliente de e-commerce.

Silvia y el equipo están emocionados por el camino que tenemos por delante. En la próxima clase, nos adentraremos en los **fundamentos de programación en NodeJS**, sentando las bases para construir aplicaciones robustas y eficientes.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Materiales y Recursos Adicionales:

- **Documentación Oficial de Node.js**
  Explora la documentación oficial para obtener información detallada sobre sus características, módulos principales y ejemplos prácticos.

Preguntas para Reflexionar:

- ¿Qué beneficios aporta el modelo single-threading de Node.js en comparación con otros enfoques como el multi-threading?
- ¿Por qué Node.js no es ideal para aplicaciones que requieren operaciones intensivas de CPU?
- ¿Cómo aprovecharías las capacidades en tiempo real de Node.js en un proyecto personal o profesional?
- ¿Qué similitudes y diferencias encuentras entre trabajar con JavaScript en el navegador y en Node.js?

Próximos Pasos:

Funciones y Arrays: Conoceremos más a fondo para que sirven las funciones, que formas existen para su declaración y el concepto de ***"funciones de orden superior"***. Además, aprenderemos sobre los Arrays y los métodos disponibles para trabajar con ellos.

Clases y Objetos: Introducción a la Programación Orientada a Objetos (POO). Aprenderemos qué son las clases, cómo crear objetos y cómo trabajar con métodos y propiedades.

Módulos y librerías: Aprenderemos todo lo necesario para trabajar con código modular y de terceros.

Buenos Aires Ciudad