JAVASCRIPT

Integrantes:

- Sebastián Inca
- Andrés Peñafiel

ÍNDICE

1. Introducción
2. Desarrollo
3. Análisis
4. Estructura y caracteristicas
5. Ejercicios
6. Conclusiones

¿QUÉ ES JAVASCRIPT?

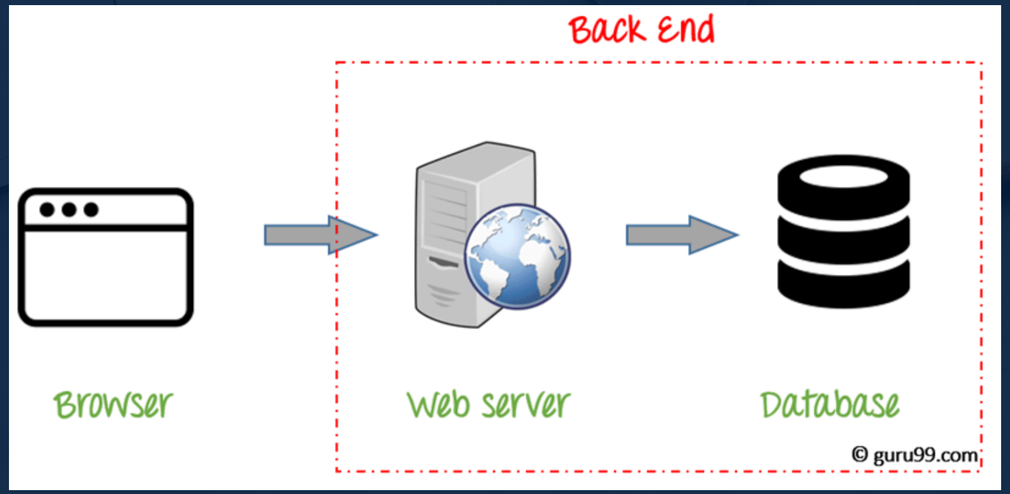
- Creado en 1995 por Brendan Eich (Netscape).
- En un inicio, solo se usaba en el frontend para interactividad web.

¿QUÉ ES JAVASCRIPT?

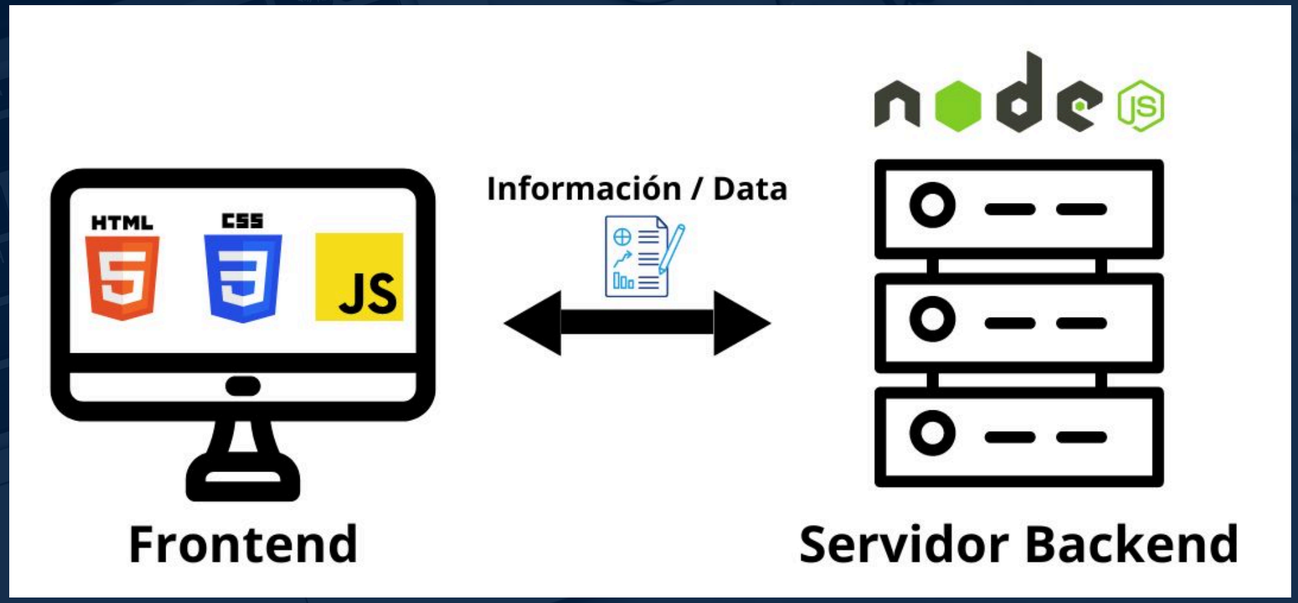
- Desde 2009, con Node.js, se puede usar también en el backend gracias a Ryan Dahl.
- Hoy en día, es un lenguaje full-stack.

¿QUÉ ES EL BACKEND?

- Parte de una aplicación web que se ejecuta en el servidor.
- Responsable de lógica de negocio, conexión a bases de datos, autenticación, APIs.
- Ejemplo visual: Frontend pide login → Backend procesa → Base de datos valida.



¿POR QUÉ JAVASCRIPT EN EL BACKEND?



- Un solo lenguaje para frontend y backend.
- Ecosistema amplio y moderno (NPM(Node Package Manager), frameworks).
- Alta demanda profesional y comunidad activa.

ANÁLISIS

VENTAJAS

- JavaScript unifica el desarrollo: frontend + backend.
- Rápido, asincrónico y eficiente (motor V8).
- Comunidad enorme, buena documentación, más de 1 millón de paquetes npm.
- Ideal para microservicios (Netflix, PayPal).

DESVENTAJAS

- La asincronía puede ser compleja (callbacks, promesas, async/await).
- No es ideal para tareas muy intensivas en CPU.
- Poca tipificación comparado con otros lenguajes (aunque TypeScript ayuda).

ESTRUCTURA DEL BACKEND CON JAVASCRIPT

NODE.JS

Entorno de ejecución para JavaScript fuera del navegador

- Permite ejecutar JS directamente en el servidor.
- Diseñado para construir aplicaciones rápidas, escalables y ligeras.
- Motor V8 de Chrome
- Node.js utiliza el motor V8 para compilar JavaScript en código máquina.
- Esto lo hace muy rápido y eficiente en el procesamiento de instrucciones.

ESTRUCTURA DEL BACKEND CON JAVASCRIPT

CARACTERÍSTICAS

- Modelo asincrónico y orientado a eventos
- Node.js no bloquea la ejecución mientras espera una tarea (I/O).
- Ideal para manejar muchas solicitudes al mismo tiempo.

Ejemplo de aplicación ideal:

- Chats en tiempo real
- Aplicaciones de streaming
- Automatización de tareas

ESTRUCTURA DEL BACKEND CON JAVASCRIPT

EXPRESS.JS

- Express.js es un framework minimalista y flexible para Node.js.
- Permite crear aplicaciones web y APIs de manera sencilla y rápida.
- Node.js es un entorno de ejecución que permite usar JavaScript fuera del navegador.
- Express.js simplifica el manejo de rutas, peticiones HTTP y middleware, agilizando el desarrollo del backend.
- Trabaja sobre Node.js, que utiliza el motor V8 de Google Chrome, el cual convierte el código JavaScript en código máquina muy rápido.

ESTRUCTURA DEL BACKEND CON JAVASCRIPT

CARACTERISTICAS

- Permite acceder a funcionalidades del sistema operativo:
- Trabajar con archivos: leer, escribir, modificar archivos del servidor.
- Manejo de red: crear servidores HTTP, gestionar conexiones, enviar respuestas.
- Gestión de procesos: controlar procesos y tareas del sistema desde JavaScript.
- Soporta middleware para extender funcionalidades (autenticación, manejo de errores, logs).
- Muy popular en el desarrollo backend por su rapidez, flexibilidad y gran comunidad.

ESTRUCTURA DEL BACKEND CON JAVASCRIPT

MONGODB

- Bases de datos: sistemas para almacenar y gestionar datos de manera organizada y eficiente.
- Se conectan con Node.js usando ORM como Sequelize o Prisma.
- Bases de datos NoSQL (no relacionales): almacenan datos en formatos más flexibles como documentos JSON.
- Ejemplo destacado: MongoDB.
- Muy usadas en aplicaciones modernas por su escalabilidad y flexibilidad.
- Se conecta con Node.js mediante librerías como Mongoose.

COMPARACIÓN

Tecnología	Lenguaje	Ventaja principal	Desventaja
Node.js	JS	Rápido y full-stack	Difícil manejo asincronía
PHP	PHP	Fácil de alojar	Sintaxis anticuada
Python	Python	Legibilidad	Más lento que Node
Java	Java	Muy escalable	Verbosidad y complejidad

PROYECTOS CON JS EN EL BACKEND

Netflix

- Usa Node.js para renderizar interfaces de usuario del lado del servidor.
- Gracias a su velocidad y asincronía, pueden manejar miles de peticiones por segundo.
- Reducción en tiempo de carga y mejora de experiencia para usuarios globales.

Páginas de E-commerce

- Backend con Node.js + Express gestiona:
- Productos, pedidos, pagos, autenticación.
- Conexión a bases de datos como MongoDB.
- Integración con servicios externos como Stripe o PayPal para pagos.

WhatsApp Web

- Node.js permite usar WebSockets para comunicación en tiempo real.
- Ideal para chats, notificaciones en vivo y sincronización entre usuarios.

PROYECTOS CON JS EN EL BACKEND

Aplicaciones de Tiempo Real como Uber / Cabify

- Node.js se usa para manejar muchas conexiones simultáneas (eventos asincrónicos).
- Se combina con tecnologías como Socket.io y Redis para tiempo real.

API REST para Aplicaciones Móviles como Instagram

- Backend con Node.js + Express gestiona:
- Productos, pedidos, pagos, autenticación.
- Conexión a bases de datos como MongoDB.
- Integración con servicios externos como Stripe o PayPal para pagos.

Microservicios como PayPal

- Node.js se usa para microservicios que se encargan de tareas específicas (pagos, notificaciones, estadísticas).
- Alta escalabilidad y bajo consumo de recursos.

PRESENTACIÓN DE EJERCICIOS

1. INSTALAR VISUAL STUDIO CODE
HTTPS://CODE.VISUALSTUDIO.COM/

2. INSTALAR NODE.JS
HTTPS://NODEJS.ORG/ES
DESCARGA LA VERSIÓN LTS (ESTABLE).

ABRE LA TERMINAL DE VS CODE Y VERIFICA:

NODE -V

NPM -V

4. INICIALIZAR PROYECTO NODE.JS
ABRE LA TERMINAL INTEGRADA EN VS CODE:

NPM INIT -Y

5. INSTALAR EXPRESS
NPM INSTALL EXPRESS

EXPECTATIVAS Y TENDENCIAS

1 Tendencias Positivas

- Node.js seguirá liderando, con mejoras en rendimiento y manejo de procesos asíncronos.
- Mayor uso de TypeScript, GraphQL, y nuevas herramientas como Bun.js.
- Integración más fuerte con serverless, microservicios y la nube.
- Avances en automatización, WebAssembly y despliegues CI/CD.

2 Tecnologías emergentes

- Frameworks más rápidos que Express (ej. Fastify, Bun).
- ORM y conectores más inteligentes (Mongoose, Prisma).
- Aumento de servicios backend como funciones (FaaS).

3 Desafíos y riesgos

- Competencia con lenguajes como Rust y Go.
- Riesgos de consumo excesivo si el código no está optimizado.
- Evolución rápida puede afectar la estabilidad en proyectos a largo plazo.

CONCLUSIÓN

- JavaScript ha dejado de ser exclusivo del navegador.
- Con Node.js y Express se pueden construir APIs modernas, rápidas y escalables.
- Aprender JavaScript en el backend permite dominar el desarrollo web de extremo a extremo.
- Ideal para startups, prototipos y proyectos en producción.

"Usar JavaScript en el backend no solo es posible, es estratégico en el desarrollo web moderno."

GRACIAS