02. NodeJS: Node Deno y Bun

Vamos a situar NodeJS en el contexto de otras tecnologías similares que han surgido más recientemente.

El Ecosistema de Runtimes de JavaScript

Ahora que entendemos qué es Node.js y para qué sirve, es interesante saber que no es la única herramienta de este tipo. En el mundo de la tecnología, las cosas evolucionan constantemente, y han surgido alternativas a Node.js, cada una con sus propias ideas y enfoques. Las dos más notables hoy en día son **Deno** y **Bun**.

- **Node.js (El Pionero Establecido):**
  - Es el entorno de ejecución que popularizó el uso de JavaScript en el servidor.
  - Lanzado en 2009, tiene una **enorme comunidad** y un **ecosistema gigantesco** de paquetes (módulos) disponibles a través de **npm** (Node Package Manager). Casi cualquier cosa que quieran hacer, probablemente ya existe un paquete de npm para ello.
  - Es **maduro, estable** y ampliamente utilizado en producción por miles de empresas, desde startups hasta gigantes tecnológicos.
  - Utiliza el motor **V8** de Google (el mismo que Chrome).
  - Históricamente usó un sistema de módulos llamado **CommonJS** (`require()`), aunque ahora también soporta el estándar **ES Modules** (`import`).
- **Deno (El Sucesor "Seguro por Defecto"):**
  - Deno fue creado por **Ryan Dahl**, la misma persona que creó Node.js. Lo inició para corregir algunas decisiones de diseño de Node.js que lamentó con el tiempo.
  - Lanzado en 2018, Deno se enfoca en la **seguridad** y una **experiencia de desarrollador moderna**.
  - **Seguro por defecto:** A diferencia de Node.js, un script de Deno no puede acceder al sistema de archivos, la red o el entorno sin permisos explícitos. Tienes que concederle permiso mediante flags (`-allow-net`, `-allow-read`).
  - **Soporte nativo para TypeScript:** Puedes escribir y ejecutar archivos TypeScript directamente, sin necesidad de pasos de compilación separados.
  - **Utiliza ES Modules (`import`)** como sistema de módulos estándar desde el principio. Los módulos se importan directamente desde URLs.
  - **Conjunto de herramientas estándar incorporado:** Incluye un linter, formateador, bundler y test runner, buscando reducir la dependencia de herramientas externas.
  - También usa el motor **V8**.
- **Bun (El Retador Enfocado en la Velocidad):**
  - Es el más reciente de los tres, ganando mucha atención rápidamente (lanzado alrededor de 2022).
  - Su principal objetivo es ser **extremadamente rápido**. Se promociona como un *runtime* de JavaScript "todo en uno" y de alto rendimiento.

- **Velocidad:** Busca ser más rápido que Node.js y Deno en tiempo de inicio, ejecución de código y operaciones de E/S. Utiliza el motor **JavaScriptCore** (el de Safari/WebKit) en lugar de V8, optimizado para un arranque rápido.
- **Todo en uno:** Incluye un bundler, un transpiler (para JSX/TS), un gestor de paquetes compatible con npm y un test runner, todo integrado y optimizado para la velocidad.
- **Alta compatibilidad con Node.js:** Se esfuerza por ser compatible con la mayoría de las APIs de Node.js y los paquetes de npm, facilitando la migración desde Node.js.
- **Soporte nativo para TypeScript y JSX**.

¿Por qué nos centramos en Node.js en este curso?

Aunque Deno y Bun son tecnologías emocionantes y prometedoras:

1. **Madurez y Ecosistema:** Node.js sigue siendo el estándar de la industria. La cantidad de recursos, tutoriales, ofertas de trabajo y paquetes disponibles para Node.js es inmensa.
2. **Base Fundamental:** Entender Node.js proporciona una base sólida. Muchos de los conceptos (manejo de eventos, asincronía, módulos) son transferibles a Deno y Bun.
3. **Adopción:** La gran mayoría de las empresas que usan JavaScript en el backend hoy en día usan Node.js.

En resumen:

Característica	Node.js	Deno	Bun
Creación	2009	2018 (por el creador de Node.js)	~2022
Foco Principal	Ecosistema, Madurez	Seguridad, Modernidad, TS	Velocidad, Todo-en-uno
Motor JS	V8	V8	JavaScriptCore (WebKit)
Seguridad	Acceso por defecto	Seguro por defecto (permisos)	Acceso por defecto (similar a Node)
TypeScript	Vía configuración/compilación	Nativo	Nativo
Módulos (Default)	CommonJS (histórico), ESM	ES Modules (URL)	ES Modules, CommonJS (compatible)
Tooling	Externo (npm, etc.)	Integrado (linter, fmt, test)	Integrado (bundle, install, run, test)
Compatibilidad	-	Menor con Node.js APIs	Alta con Node.js APIs y npm
Ecosistema (npm)	Gigantesco	Usa su propio sistema + npm (vía compat)	Compatible con npm

Conocer Deno y Bun nos da perspectiva. Son innovaciones importantes que están empujando los límites de lo que se puede hacer con JavaScript fuera del navegador. Sin embargo, para construir una base sólida y entender el panorama actual del desarrollo backend con JavaScript, Node.js es el punto de partida perfecto

---