AGO 3

SPRINT 3 - MEETING 39

Introducción al NodeJS

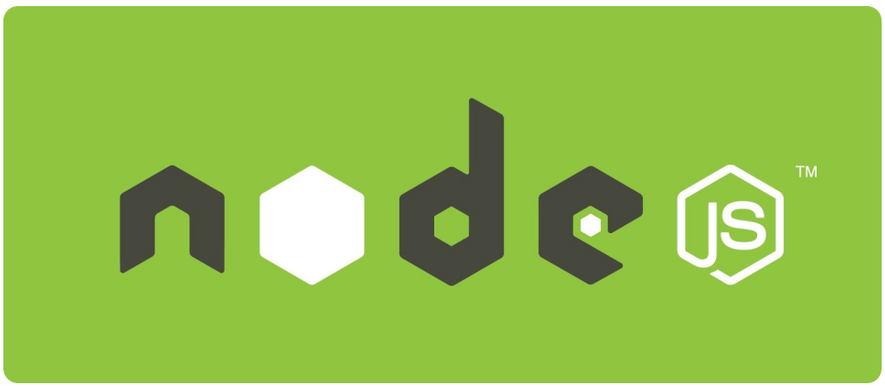
Grabación

Presentación

El equipo docente subió contenido extra de la meeting

¿JavaScript también en el servidor?

> *"Lo que en realidad significa es que Node.js no es nueva plataforma que dominará el mundo del desarrollo web. Al contrario, se trata de una plataforma que llena una necesidad en particular." —*Tomislav Capan.* JavaScript Developer. *



Hemos transitado dos encuentros de este nuevo bloque donde aprendimos a estructurar una API a través de la documentación en Swagger y cómo probarla con Postman. Ahora es tiempo de introducirnos en un lenguaje de Backend para poder hacer que nuestras APIs cobren vida.

Una de las maneras sería aprender un nuevo lenguaje como Python, PHP, o .NET. pero, ¿es realmente necesario? No, ya que en el ámbito de la programación contamos con **librerías** o **frameworks** para facilitar el trabajo del desarrollo web y ahorrar tiempo. Antes de avanzar tenemos que hacer una aclaración entre estos dos conceptos: según Digital Guide Ionos —uno de los principales proveedores europeos de infraestructura cloud, servicios cloud y hosting con más de ocho millones de clientes— hace mención que se suelen utilizar los términos **librería** o **framework** para nombrar lo mismo, aunque son diferentes. ¡Veamos cuáles son sus diferencias!

- Una **librería hace referencia a una librería de programas, que siempre aloja subprogramas** y facilitan la programación gracias a sus funciones de ayuda. Se desarrolla para ser usada de una manera determinada y posee, para ello, funciones que han sido ajustadas las unas a las otras.
- Los **frameworks son un tipo especial de biblioteca de clases. Representan la**

fundamentalmente el proceso de desarrollo. Estos poseen unos modelos concretos de desarrollo que, a su vez, cuentan con diversas funciones (a menudo en forma de varias librerías) y sirven para desarrollar aplicaciones nuevas e independientes.

Entonces, si ya contamos con los conocimientos de JavaScript...¡usémoslos! Este lenguaje tiene un alcance más allá de crear sitios web interactivos: también lo podemos utilizar para desarrollar aplicaciones desde el Backend. Esto es posible gracias al framework que funciona del lado del servidor Node JS. Es un entorno de ejecución multiplataforma de código abierto creado por Ryan Dahl, ingeniero de software estadounidense. La primera presentación se llevó a cabo en la JSConf (Conferences for the JavaScript community) en 2009.

Hiren Dhaduk, Vicepresidente de Tecnología en Simform, en su artículo cuenta cuál fue el impacto de esta nueva herramienta:

"Recuerdo cuando solíamos sentarnos durante horas y horas coordinando entre desarrolladores front-end y back-end que escribían diferentes scripts para cada lado. Todo esto cambió tan pronto como Node.js entró en escena. Creo que lo único que impulsa a los desarrolladores hacia esta tecnología es su eficiencia bidireccional."

Ya contamos con los conocimientos de JavaScript, entonces ¡usémoslos!. Este lenguaje tiene un mayor alcance que solo crear sitios web interactivos, también es posible desarrollar aplicaciones desde el Backend a través de Node JS.

Con Node JS es posible ejecutar código JavaScript sin la necesidad de un navegador, en un entorno en el que corre JavaScript a través de la Máquina Virtual V8) de Google, al igual que en la consola de Google Chrome.

¿Por qué implementarlo? Según Hack a BOSS, actualmente una serie de reconocidas empresas lo utilizan en sus estructuras back-end para mejorar el rendimiento de sus aplicaciones, entre ellas Uber, Netflix, Ebay, LinkedIn, Paypal. Contar con esta herramienta en nuestro stack nos permite estar mejor posicionados/as como Developers en el mercado laboral.

Aquí las principales ventajas de su uso:

- **Más eficiente:** utiliza el mismo lenguaje tanto en el back como el front y no tenemos que switchear tecnologías.
- **Más veloz:** permite crear páginas o aplicaciones móviles mucho más rápidas del lado del servidor.
- **Asincrónico:** no se detiene a esperar a que termine una operación, sino que continúa su ejecución con otras instrucciones y es posible definir una acción para

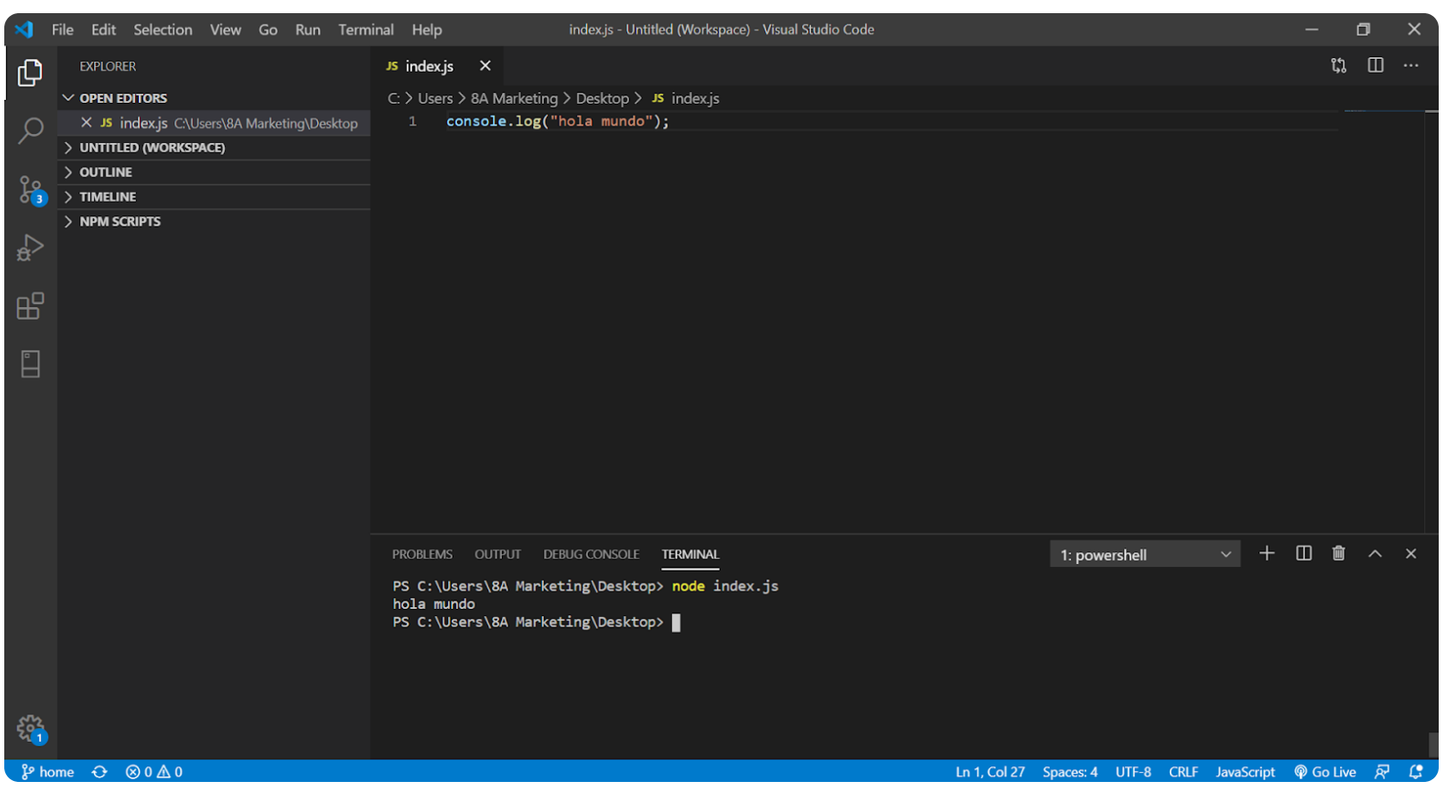
Iremos aprendiendo cómo se ejecuta JavaScript del lado del servidor, y cuáles son sus funcionalidades para crear aplicaciones en tiempo real que sean más rápidas, escalables y eficientes.

¿Cómo usar Node JS?

Para poder utilizar **Node JS** tienes que descargarlo e instalarlo en tu computadora. Dado que ya manejas HTML, CSS y JavaScript, te resultará más fácil aprender esto, y además cuentas con tu aliado Visual Studio Code.

Vamos a corroborar su funcionamiento con el clásico **Hola Mundo**. Abre la consola del Visual Studio:

1. Crea un archivo index.js
2. Escribe un console.log("Hola mundo");
3. Ejecuta el index.js con Node



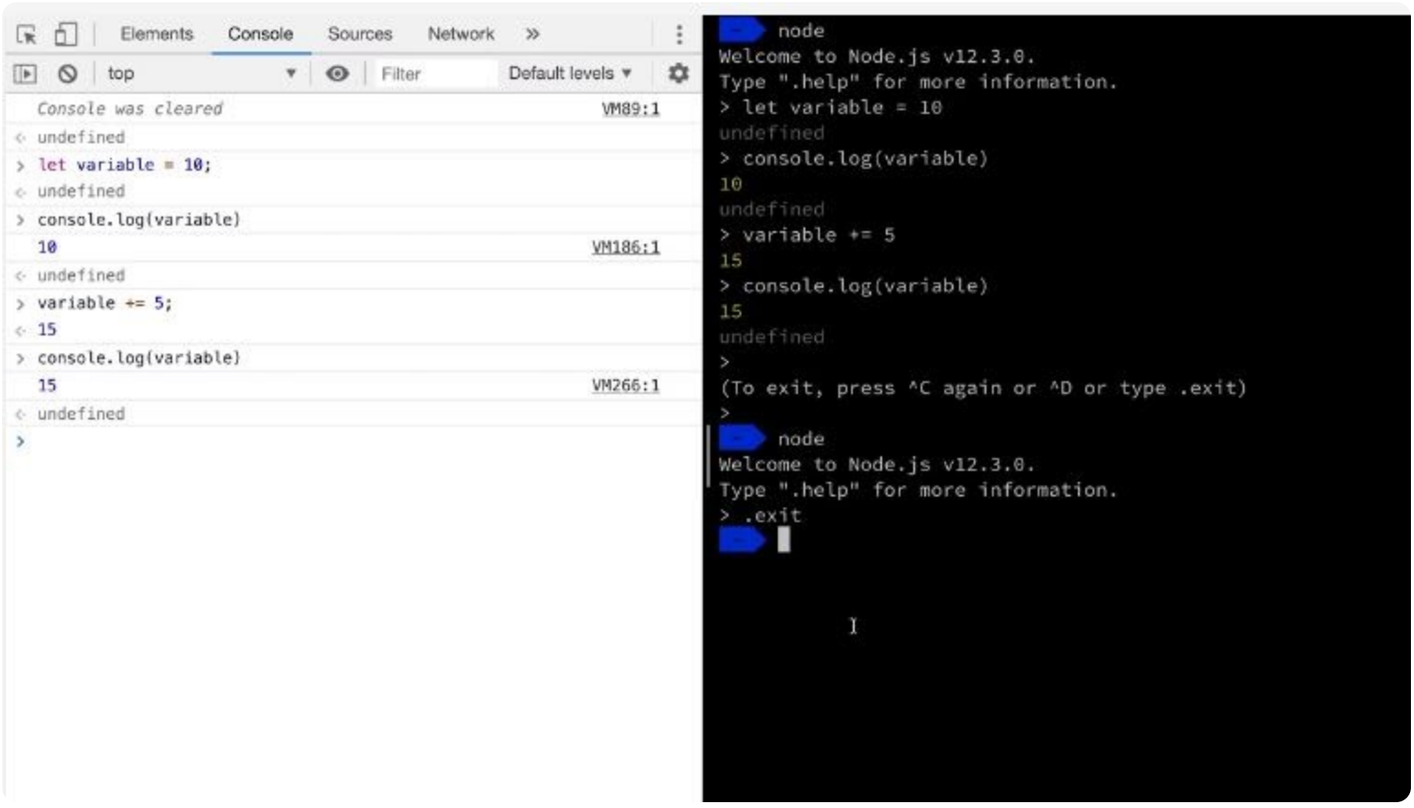
Avancemos con otro ejemplo que incluya variables y comparemos los resultados de la consola del Chrome y Node JS:

1. Elige la variable
2. Asígnale un valor inicial de 10

4. Añade 5 a tu variable

5. Imprime nuevamente tu variable con un **console.log**

A la izquierda veremos en la imágen el resultado de Chrome y a la derecha en Node JS:



En el encuentro profundizaremos sobre su uso, pero es fundamental que comiences probándolo.

Crear librerías o módulos propios

Otra gran ventaja de Node JS es que nos permite reutilizar nuestro código a través de la creación de librerías o módulos. Podemos utilizar los que trae nativamente el lenguaje o instalar módulos de terceros. Un módulo es una unidad de código. Según Mialto web:

"Los módulos en NodeJS son una de las características más relevantes del propio Node, debido a que aceleran mucho el desarrollo, permiten utilizar códigos de terceros y estructuran el código."

Node JS viene con módulos predefinidos como **Owes** (acceder a funciones del sistema operativo) o **FS** (acceder a los archivos del sistema) entre los disponibles. Es importante que consultes la documentación.

Vamos a crear librerías o módulos propios cuando tenemos el mismo código que se utiliza en más de un proyecto o cuando queramos separar las responsabilidades o funcionalidades en distintos módulos. El motivo más común es este último, ya que en aplicaciones grandes lo ideal es tener todo bien organizado.

Por ejemplo, al comenzar un proyecto es importante que tengas en cuenta su tamaño para poder

embargo, si el proyecto es más grande, una buena práctica que tenemos los/as developers es separar el código en archivos más chicos para tener las responsabilidades de forma más clara. De esta forma, no sólo tenemos un código más fácil de leer, sino que además tenemos módulos que podemos reutilizar. Si a futuro surge algún problema, sabremos directamente dónde consultar para modificarlo.

En el próximo encuentro verás cómo crear tus propios módulos y disponibilizar las variables y funciones en el lugar que más te convenga.

Cierre

En resumen, **Node JS** es un motor que nos permite ejecutar JavaScript del lado del servidor, lo cual nos abre las puertas a innumerables beneficios como conectarnos a una base de datos, trabajar con el sistema operativo o tener acceso a escritura en el disco rígido, entre otros.

Desde el front-end nos sería imposible tener a nuestro alcance estos recursos, pero al trabajar con un lenguaje de back-end estamos mucho más cerca del sistema operativo y tenemos estas asombrosas funcionalidades disponibles..!

¡Prepárate para el próximo encuentro!

Profundiza

Te invitamos a conocer más sobre el tema de esta bitácora.

Herramientas

Programas necesarios para facilitar tu experiencia.

Challenge

Te proponemos el siguiente desafío, ¿te animas?

Contenido extra de la meeting

Archivos de la clase

Saludos!

MEETING 38

MEETING 40