



Buenos
Aires
Ciudad
Agencia de Habilidades
para el Futuro

<Talento
Tech/>

Node.JS

Clase 5: Módulos y Gestores de Paquetes en Node.js

¡Les damos la bienvenida!

 Vamos a comenzar a grabar la clase.

Índice



1



2

Módulos y gestores de paquetes

- Gestores de paquetes:
- NPM - Node Package Manager:
 - Instalación de paquetes
 - Creación de scripts
 - Gestión de dependencias.
- Inicio de un proyecto nuevo con Node Js y NPM
- Módulos Nativos
- Módulos de terceros
- Dirname: Gestión de rutas absolutas y acceso a archivos del servidor

Asincronismo

- Fundamentos del asincronismo
- Manejo de promesas:
 - callbacks
 - then, catch & finally
 - async & await
- Fetch: consumiendo datos externos

Objetivos de la Clase

- 1** Comprender qué son los gestores de paquetes y cómo utilizarlos en proyectos con Node.js.
- 2** Aprender a iniciar y configurar un proyecto con Node.js y NPM, incluyendo la instalación de dependencias y la creación de scripts.
- 3** Explorar los módulos nativos path, fs y process para manejar rutas, archivos y procesos.
- 4** Implementar módulos de terceros en un proyecto utilizando NPM.
- 5** Manejar rutas absolutas y relativas con __dirname para acceder a archivos del servidor.

Gestores de paquetes

¿Qué son los Gestores de Paquetes?



1 Biblioteca de Módulos

Facilitan la instalación y gestión de bibliotecas de código.

2 Gestión de Dependencias

Manejan versiones y resuelven conflictos entre paquetes.

3 Automatización

Simplifican la integración de código de terceros en proyectos.

Gestores de Paquetes Populares



NPM

Node Package Manager o el gestor de paquetes más utilizado para Node.js.



Homebrew

Popular en sistemas Mac OS para gestión de programas.



Composer

Utilizado en proyectos de PHP para manejar dependencias.



Pip

Gestor de paquetes para el lenguaje **Python**.



Chocolatey

El gestor de paquetes del sistema operativo Microsoft nos permite instalar infinidad de programas desde la terminal sin tener que descargar los ejecutables.

NPM: Node Package Manager

NPM: Node Package Manager

Gestión de Paquetes

Instala, actualiza y elimina dependencias de forma sencilla.

Configuración de Proyectos

Inicializa proyectos Node.js y crea archivos de configuración.

Scripts Personalizados

Permite definir y ejecutar scripts de proyecto desde la terminal.



Iniciando un Proyecto con NPM



1

Verificar Instalación

Comprueba la instalación de Node.js y NPM con '`node -v`' y '`npm -v`'.

2

Crear Directorio

Crea una nueva carpeta para tu proyecto.

3

Iniciar Proyecto

Ejecuta '`npm init`' o '`npm init -y`' para configuración rápida.

Package.json: El Corazón del Proyecto

Metadatos del Proyecto

Contiene nombre, versión, descripción y autor del proyecto.

Dependencias

Lista las dependencias de producción y desarrollo del proyecto.

Scripts

Define comandos personalizados para ejecutar tareas del proyecto.

Instalación y Gestión de Paquetes

1

Usa `npm install [paquete]` para agregar dependencias.

2

Ejecuta `npm update [paquete]` para obtener la última versión.

3

Elimina paquetes con `npm uninstall [paquete]`.

Creación de Scripts Personalizados

Definir Scripts

Agrega scripts personalizados en la sección "scripts" del package.json.

Ejecutar Scripts

Usa 'npm run [nombre-del-script]' para ejecutar tus scripts personalizados.

Automatización

Crea scripts para tareas repetitivas como iniciar el servidor o compilar código.



Módulos

Módulos en Node.js

¿Qué son los módulos?

Los módulos son bloques de código reutilizables y organizados que encapsulan la funcionalidad de una aplicación. Permiten dividir el código en partes más pequeñas y manejables, lo que facilita el mantenimiento, la reutilización y la colaboración en proyectos.

Módulos Internos

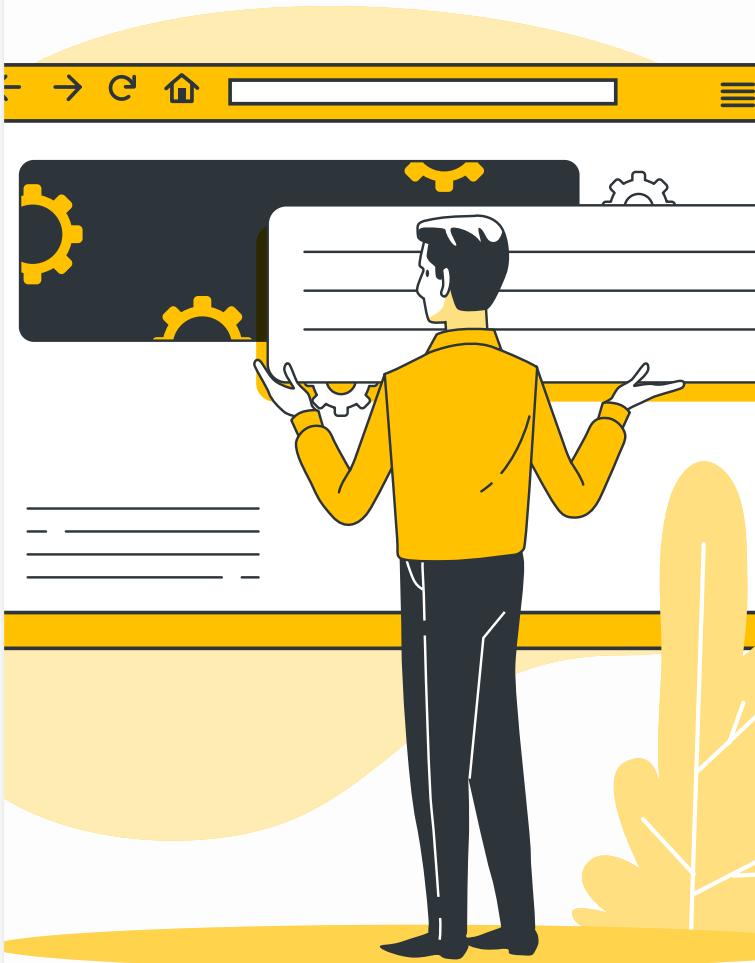
Archivos de tu proyecto que exportan código.

Módulos Nativos

Incluidos en Node.js, como 'fs' y 'http'.

Módulos Externos

Paquetes de terceros instalados via NPM.



CommonJS vs ES Modules

Actualmente, en JavaScript existen dos estándares principales para importar y exportar código, cada uno con características y casos de uso particulares:

CommonJS

Utiliza '`require()`' y '`module.exports`'. Es síncrono y compatible con Node.js.

ES Modules

Usa '`import`' y '`export`'. Es asíncrono y estándar en JavaScript moderno.

Dirname

__dirname y Rutas Absolutas

`__dirname` proporciona la ruta absoluta del directorio actual.

Uso en CommonJS

Disponible globalmente en módulos CommonJS.

Alternativa en ES Modules

Usa `'import.meta.url'` con `'url.fileURLToPath()'` para obtener la ruta.



Trabajando con Rutas en Node.js

1 Obtener Ruta Actual

Usa `__dirname` o `import.meta.url` según el sistema de módulos.

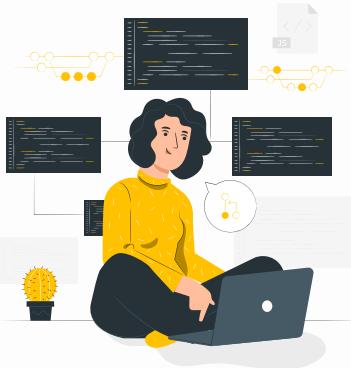
2 Combinar Rutas

Utiliza `path.join()` del módulo `path` para crear rutas absolutas confiables.

3 Acceder a Archivos

Usa `fs.readFile()` con la ruta absoluta para leer archivos.

Process y Argumentos de Línea de Comandos



El módulo **process** es uno de los módulos nativos más importantes de Node.js, ya que permite interactuar con el proceso en ejecución. Una de sus propiedades clave es **process.argv**, un array que contiene los argumentos pasados al script desde la línea de comandos.

Estructura

Incluye ruta de Node.js, ruta del script y argumentos adicionales.

Uso

Permite crear scripts interactivos que responden a diferentes argumentos.

Próximos Pasos y Recursos

Manejo de Promesas:

Exploraremos cómo manejar procesos asíncronos utilizando promesas y `async/await` en JavaScript.

Servidores Web:

Aprenderemos sobre la comunicación web y cómo funcionan los servidores que dan vida a internet.

Patrones de Arquitectura:

Conoceremos sobre los cimientos de los proyectos de programación.

[Introducción a NPM](#) – Documentación oficial de Node.js para conocer más sobre el gestor de paquetes de Node.

[NPM Documentation](#) – Guía oficial sobre cómo trabajar con NPM y gestionar dependencias.

[Módulos en Node JS](#) - Documentación oficial sobre el manejo de módulos.

[Módulo process](#) - Leyendo argumentos desde la línea de comandos.



Ejercicios Prácticos

Storytelling

Matías y Sabrina te observan mientras revisas tus últimos desafíos resueltos. "Es momento de subir el nivel", dice Matías, mientras Sabrina asiente.



i "Queremos que configures un entorno de proyecto más profesional, algo que será clave si llegas a trabajar con nosotros en un futuro. Aquí tienes el siguiente reto".



Ejercicio Práctico

Obligatorio

Misión 1:

Crea un nuevo proyecto en Node Js mediante el comando `npm init -y` y configura un nuevo script dentro del archivo ***package.json*** que mediante la instrucción `npm run start` ejecute el código de nuestro archivo `script.js` de forma automática.

Asegúrate de que el entorno esté listo y funcionando correctamente. Este paso marcará el inicio de proyectos más complejos. Con el proyecto configurado, Matías lanza el siguiente desafío.



“El siguiente paso evalúa tu capacidad para manejar interacciones dinámicas en Node.js”, dice mientras sonríe. “Queremos que implementes un sistema simple para procesar comandos desde la terminal”.



Ejercicio Práctico

Obligatorio

Misión 2:

- Si el comando es **GET**, imprime por consola el mensaje:

Toma un dato

- Si el comando es **POST {data}**, imprime por consola el mensaje:

Recibimos {data} satisfactoriamente

- Si el comando es **PUT {id}**, imprime por consola el mensaje:

Modificamos el item con id: {id} satisfactoriamente

- Si el comando es **DELETE {id}**, imprime por consola el mensaje:

El item con el id: {id} se eliminó con éxito

Matías concluye: “Este desafío es clave para ver cómo manejas el flujo de datos y comandos, algo vital en cualquier proyecto backend”.

