

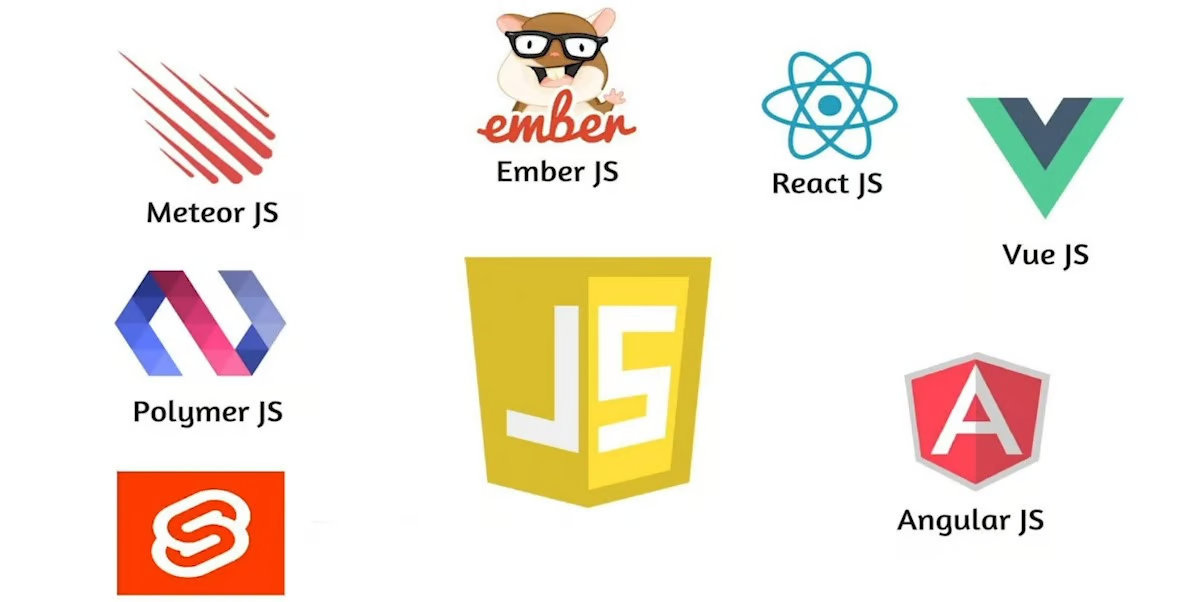
Unidad 10: JavaScript y Frameworks

Unidad 10: JavaScript y Framework

# **Introducción a Frameworks**

## **¿Qué es un Framework?**

Un **framework** es un marco de trabajo que provee un conjunto de estándares, paradigmas y buenas prácticas, junto con funcionalidades predesarrolladas para facilitar el desarrollo de aplicaciones en un lenguaje de programación. Los frameworks permiten organizar y estandarizar el trabajo, lo cual es esencial en proyectos grandes o en equipos de desarrollo, donde seguir un conjunto de normas garantiza que el código sea más mantenible y escalable.



A diferencia de una **librería**, que es un conjunto de funciones y herramientas específicas que los desarrolladores pueden utilizar a demanda, un framework define la estructura completa de una aplicación y dicta el flujo de trabajo, proporcionando más control sobre el código.

### **Introducción a NodeJS**

**NodeJS** es un entorno de ejecución para JavaScript que permite ejecutar código JavaScript fuera del navegador. Fue creado en 2009 por Ryan Dahl, utilizando el motor V8 de Google Chrome, que es responsable de convertir el código JavaScript en código que puede ser ejecutado por la CPU. Antes de NodeJS, JavaScript solo podía ejecutarse dentro de un navegador, limitando su uso a la creación de aplicaciones web del lado del cliente.

Con NodeJS, se puede utilizar JavaScript para desarrollar aplicaciones del lado del servidor, como servidores HTTP, acceder a archivos del sistema, y otras tareas que tradicionalmente requerían lenguajes como Python o Ruby. Esto ha ampliado enormemente las capacidades de JavaScript, permitiendo que sea utilizado tanto en el frontend como en el backend.

### **Introducción a NPM (Node Package Manager)**

**NPM** es el gestor de paquetes oficial de NodeJS. Es una herramienta de línea de comandos que permite a los desarrolladores gestionar las dependencias de su proyecto. A través de NPM, se pueden instalar, actualizar y eliminar módulos y librerías que son necesarios para el funcionamiento de una aplicación. Los módulos instalados con NPM se almacenan en la carpeta node\_modules del proyecto, y son fácilmente reutilizables.

NPM también permite la creación de un archivo package.json, que contiene toda la información relevante sobre el proyecto, como su nombre, versión, autor, dependencias y scripts que pueden ser ejecutados mediante la línea de comandos. Este archivo es esencial para mantener la coherencia y facilitar la colaboración en proyectos de software.

## **Introducción a Frameworks**

En el desarrollo moderno, es crucial entender la importancia de los frameworks como NodeJS y herramientas asociadas como NPM. Estos no solo han revolucionado la forma en que se construyen aplicaciones, sino que también han ampliado el alcance de JavaScript más allá del navegador, permitiendo su uso en el lado del servidor.

**NodeJS** es un entorno de ejecución que permite a los desarrolladores ejecutar JavaScript fuera del navegador. Esto ha abierto un mundo de posibilidades, incluyendo la creación de aplicaciones de backend y la posibilidad de manejar peticiones de servidores con JavaScript, algo impensable en los primeros días del lenguaje.

Por otro lado, **NPM (Node Package Manager)** es la herramienta que acompaña a NodeJS y facilita la gestión de dependencias en proyectos de JavaScript. Con NPM, los desarrolladores pueden acceder a una vasta cantidad de paquetes y módulos que aceleran el desarrollo, manteniendo el código organizado y eficiente.

Dominar estos conceptos es esencial para cualquier desarrollador que busque mantenerse relevante y competitivo en el ámbito del desarrollo web, ya que frameworks como React, Angular, y Vue dependen de NodeJS y NPM para su correcto funcionamiento.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Proyectos con Node y NPM**

### **Iniciar Proyectos Usando NPM y Creación de package.json**

Para comenzar un proyecto con NodeJS, el primer paso es inicializar NPM (Node Package Manager). NPM es una herramienta fundamental en el desarrollo con NodeJS, ya que permite gestionar las dependencias del proyecto, instalar paquetes y compartir código fácilmente.

### 

### **Pasos para Iniciar un Proyecto con NPM**

**1. Crear un directorio para tu proyecto**: Comienza creando un nuevo directorio donde se alojará tu proyecto. Navega a este directorio usando la terminal.

| mkdir mi-proyecto  cd mi-proyecto |
| --- |

2. **Inicializar NPM**: Una vez dentro de tu directorio, debes inicializar NPM. Esto creará un archivo llamado package.json que almacenará información sobre tu proyecto y sus dependencias.

| npm init |
| --- |

Durante la ejecución de este comando, NPM te hará una serie de preguntas sobre tu proyecto (nombre, versión, descripción, entry point, etc.). Puedes completar estos campos o simplemente presionar Enter para aceptar los valores por defecto. Al finalizar, tendrás un archivo package.json en tu directorio.

**3. El archivo package.json**: Este archivo es el núcleo de cualquier proyecto basado en NodeJS. Contiene metadatos sobre tu aplicación, incluyendo las dependencias necesarias para que funcione correctamente. Un ejemplo básico de un package.json podría verse así:

| {  "name": "mi-proyecto",  "version": "1.0.0",  "description": "Este es un proyecto inicializado con NPM",  "main": "index.js",  "scripts": {  "test": "echo \\\\"Error: no test specified\\\\" && exit 1"  },  "author": "Tu Nombre",  "license": "ISC"  } |
| --- |

**4. Instalación de Dependencias**: Para instalar dependencias (paquetes) en tu proyecto, puedes usar el comando npm install seguido del nombre del paquete. Por ejemplo, si deseas instalar express, un popular framework para NodeJS, puedes hacerlo de la siguiente manera:

| npm install express |
| --- |

Este comando agregará express a la lista de dependencias en tu archivo package.json y creará una carpeta node\_modules donde se almacenarán todos los paquetes instalados.

**5. Uso de Scripts en package.json**: Dentro de package.json, puedes definir scripts para automatizar tareas comunes. Por ejemplo, podrías agregar un script para iniciar tu aplicación:

| "scripts": {  "start": "node index.js"  } |
| --- |

Luego, podrías ejecutar este script simplemente con el comando:

| npm start |
| --- |

**Conclusión**

Iniciar un proyecto con NPM y gestionar las dependencias mediante package.json es una habilidad esencial para cualquier desarrollador que trabaje con NodeJS. Este proceso no solo facilita la instalación y actualización de paquetes, sino que también mantiene tu proyecto organizado y coherente, lo que es crucial para el desarrollo colaborativo y la escalabilidad de tus aplicaciones.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Paradigmas en Programación**

### **Paradigmas en Programación: MVC y MVVM**

En el desarrollo web, la arquitectura de una aplicación es fundamental para mantener el código organizado, modular y fácil de mantener. Para lograr esto, los desarrolladores emplean diferentes paradigmas de diseño, entre los cuales destacan MVC (Modelo-Vista-Controlador) y MVVM (Modelo-Vista-ViewModel).

### **Modelo-Vista-Controlador (MVC)**

MVC es uno de los patrones de diseño más antiguos y ampliamente utilizados en la programación. Este paradigma separa una aplicación en tres componentes principales:

* **Modelo**: Es el encargado de gestionar los datos de la aplicación. Contiene la lógica de negocio y las reglas para manipular los datos.
* **Vista**: Es la interfaz de usuario. Presenta los datos al usuario y recibe la interacción de este.
* **Controlador**: Actúa como intermediario entre la Vista y el Modelo. Procesa las entradas del usuario recibidas a través de la Vista, las interpreta y realiza las acciones necesarias sobre el Modelo.

El MVC es particularmente útil para aplicaciones donde las interfaces de usuario necesitan ser altamente reactivas, y el código debe ser modular para facilitar el desarrollo y mantenimiento.

### **Modelo-Vista-ViewModel (MVVM)**

MVVM es un paradigma de diseño que surgió como evolución del MVC, especialmente en el contexto del desarrollo de aplicaciones con interfaces de usuario ricas y complejas, como las que se construyen con frameworks modernos de JavaScript (Angular, Vue, React).

* **Modelo**: Igual que en MVC, gestiona los datos y la lógica de negocio.
* **Vista**: La interfaz de usuario que se muestra al usuario final.
* **ViewModel**: Es una abstracción de la Vista que maneja la lógica de presentación. Vincula los datos del Modelo con los elementos de la Vista de manera más directa, permitiendo que los cambios en el Modelo se reflejen automáticamente en la Vista y viceversa.

El MVVM facilita el desarrollo de aplicaciones dinámicas donde la Vista necesita reflejar automáticamente los cambios en el Modelo, y es común en aplicaciones de una sola página (SPA, por sus siglas en inglés).

### **Aplicación en el Desarrollo Web**

Ambos paradigmas, MVC y MVVM, se aplican ampliamente en el desarrollo web moderno. Frameworks como Angular, React y Vue han adoptado y adaptado estos paradigmas para permitir a los desarrolladores construir aplicaciones complejas y mantenibles con mayor eficiencia. El uso de estos patrones de diseño no solo organiza el código, sino que también mejora la escalabilidad y testabilidad de las aplicaciones.

En resumen, comprender y aplicar correctamente los paradigmas MVC y MVVM es crucial para el desarrollo de aplicaciones web robustas, especialmente en entornos donde la modularidad y la reactividad son esenciales.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Caminos del Frontend**

## **Librerías y frameworks**

### **Descripción de Librerías y Frameworks: React, Angular y Vue**

En el desarrollo web moderno, elegir el framework o librería adecuado es crucial para cumplir con las necesidades específicas de un proyecto. Entre las herramientas más populares se encuentran **React**, **Angular** y **Vue**. Cada una tiene sus características únicas y se adapta a diferentes escenarios en el mercado.

### **React**

React es una librería de JavaScript desarrollada por Facebook que se enfoca en la creación de interfaces de usuario. Su principal ventaja es la **flexibilidad** y la **alta performance** al renderizar componentes. React utiliza un **Virtual DOM** para optimizar las actualizaciones de la interfaz de usuario, lo que permite un rendimiento superior en aplicaciones dinámicas y ricas en interactividad.

* **Uso principal**: Es ideal para desarrollar **Single Page Applications (SPA)** con interfaces de usuario altamente interactivas.
* **Ventajas**: Ofrece una gran **modularidad** y un ecosistema amplio de herramientas y librerías complementarias.
* **Desventajas**: Al ser solo una librería para la vista, requiere la integración de otras herramientas para manejar aspectos como la gestión del estado y el enrutamiento.

### **Angular**

Angular, desarrollado por Google, es un framework completo que proporciona una **solución integral** para el desarrollo de aplicaciones web. Está basado en TypeScript y utiliza un enfoque **MVC (Modelo-Vista-Controlador)** para organizar el código. Angular es conocido por su capacidad para manejar aplicaciones de gran escala debido a su sistema de **inyección de dependencias** y su arquitectura modular.

* **Uso principal**: Adecuado para proyectos grandes y **empresariales** que requieren una estructura robusta y escalable.
* **Ventajas**: Ofrece un **framework completo** con todas las herramientas necesarias integradas, lo que reduce la necesidad de dependencias externas.
* **Desventajas**: Tiene una curva de aprendizaje más pronunciada y puede ser **excesivo** para aplicaciones más pequeñas.

### **Vue**

Vue es un framework progresivo que permite construir interfaces de usuario de manera incremental. Fue creado por Evan You y se destaca por su **facilidad de uso** y su **flexibilidad**. Vue puede ser adoptado gradualmente en un proyecto, lo que lo hace ideal tanto para aplicaciones simples como complejas.

* **Uso principal**: Es ideal para aplicaciones que comienzan como pequeñas y necesitan escalar con el tiempo.
* **Ventajas**: Ofrece una combinación equilibrada entre la simplicidad de React y la estructura de Angular, con una **curva de aprendizaje suave**.
* **Desventajas**: Aunque está ganando popularidad, Vue aún no tiene el mismo nivel de **adopción empresarial** que Angular o React.

**Conclusión**

La elección entre React, Angular y Vue depende del contexto del proyecto y las necesidades específicas del equipo de desarrollo. React es preferible para aplicaciones que requieren una alta interactividad y flexibilidad; Angular es ideal para proyectos grandes y estructurados; y Vue es excelente para quienes buscan un equilibrio entre simplicidad y capacidad de escalabilidad. Conocer las fortalezas de cada uno permite tomar decisiones informadas para satisfacer las demandas del mercado actual.