Tecnicatura Superior en Ciencias de Datos e Inteligencia Artificial

Requisitos previos

Dominar estos aspectos fundamentales de JavaScript es crucial antes de comenzar con React, ya que React se construye sobre el lenguaje y sus conceptos básicos. Comprender variables, funciones, manejo de arrays, objetos y la asincronía te permitirá escribir componentes más sólidos y mantenibles, aprovechar el manejo inmutable del estado y entender cómo se comunican y actualizan los datos dentro de una aplicación. Además, el conocimiento profundo de ES6 y la correcta gestión de eventos facilitan la integración de diversas funcionalidades de React, asegurando que tu desarrollo se base en buenas prácticas y en una arquitectura coherente.

1. Fundamentos del lenguaje

- **Variables (`let`, `const`)**: React utiliza `const` frecuentemente para componentes y funciones, y `let` para variables mutables.
- **Tipos de datos**: Es crucial entender strings, números, booleanos, null, undefined, para manejar props, estados y condicionales.
- **Operadores**: Lógicos (`&&`, `||`) y ternarios (`? :`) se usan mucho para renderizado condicional.
- **Condicionales y bucles**: Aunque no se usan `for` directamente en JSX, `map` y operadores lógicos son clave para mostrar listas y condicionar elementos.

2. Funciones

- **Funciones tradicionales y funciones flecha**: Las funciones flecha son comunes en componentes funcionales y callbacks de eventos.

- **Parámetros por defecto y spread/rest**: Útiles para manejar props, clonar objetos/arrays y escribir código más limpio.
- **Callbacks**: Se usan para manejar eventos y comunicar componentes (por ejemplo, pasando funciones como props).

3. Arrays y objetos

- **Métodos como** `map`, `filter`, `reduce`, `find`: Esenciales para renderizar listas y transformar datos antes de mostrarlos.
- **Desestructuración**: Facilita el acceso a props y estados sin escribir código redundante.
- **Spread/rest (**`...`**)**: Muy usados para actualizar el estado de forma inmutable.

4. Manipulación y estructura de datos

- **Inmutabilidad**: React espera que el estado no se modifique directamente; se requiere crear nuevas copias con cambios.
- **Funciones puras**: Los componentes funcionales deben ser predecibles, sin efectos colaterales (al menos en el render).

5. Asincronía

- **Promesas y** `async/await`: React trabaja con datos externos (APIs), por lo que el manejo asíncrono es fundamental.
- `fetch`: Método nativo para obtener datos, ideal para empezar antes de usar bibliotecas como Axios.

6. Módulos ES6

- `import` **y** `export`: React organiza el código en componentes; entender módulos es clave para dividir y reutilizar código.

7. Eventos

- **Manejo de eventos (**`onClick`**,** `onChange`**, etc.)**: React sintetiza los eventos del DOM. Saber cómo funcionan es vital para la interactividad.

8. DOM

- **Qué es el DOM y el Virtual DOM**: No se manipula directamente, pero entender su funcionamiento ayuda a comprender cómo React actualiza la interfaz.

Recursos recomendados:

- The modern javascript tutorial https://javascript.info/
- JavaScript ES6 https://www.w3schools.com/js/js_es6.asp

Configurando el Entorno de Desarrollo

1. Instalación de Visual Studio Code

¿Qué es?

Visual Studio Code (VS Code) es un editor de código fuente ligero pero potente, desarrollado por Microsoft. Es ampliamente utilizado para el desarrollo en JavaScript, Python, C++, entre otros.

¿Por qué lo usamos?

- Integración con Git
- Terminal integrada
- Autocompletado inteligente
- Extensiones (React, Prettier, ESLint, etc.)

Pasos para instalarlo:

Ingresar a https://code.visualstudio.com/download

Descargar la versión correspondiente a tu sistema operativo

Instalar y abrir el programa

Agregar extensiones sugeridas:

- ESLint
- Prettier
- React snippets
- Tailwind CSS IntelliSense

2. Instalación de Node.js

¿Qué es Node.js?

Es un entorno de ejecución para JavaScript del lado del servidor. Incluye `npm`, el gestor de paquetes que utilizaremos para instalar librerías como React y Tailwind.

Pasos para instalarlo:

Ingresar a https://nodejs.org/es/download

Elegir la versión LTS (Long-Term Support)

Verificar instalación:

```
node -v
npm -v
```

Crear un nuevo proyecto con Vite + React + SWC

¿Qué es Vite?

Es un empaquetador de proyectos frontend moderno, rápido y optimizado para frameworks como React, Vue, Svelte, etc.

¿Qué es SWC?

SWC (Speedy Web Compiler) es un compilador ultrarrápido escrito en Rust. Reemplaza a Babel y permite transformar código moderno a versiones compatibles con todos los navegadores.

Pasos para crear el proyecto:

1. Ejecutar en terminal:

```
npm create vite@latest
```

1. Ingresar el nombre del proyecto
2. Elegir `React`

3. Seleccionar `JavaScript + SWC`

Esto generará una estructura base para trabajar con React.

Instalación y configuración de Tailwind CSS

¿Qué es Tailwind CSS?

Es un framework CSS utility-first que permite construir interfaces modernas utilizando clases directamente en el HTML o JSX, sin necesidad de escribir CSS personalizado.

Ventajas de usar Tailwind:

- Velocidad en el prototipado
- Consistencia visual
- Adaptado a diseño responsivo
- Combinable con componentes de React

Pasos para instalarlo (con Vite):

Seguir la guía oficial: https://tailwindcss.com/docs/installation/using-vite

Luego ya estamos en condiciones de probar la App:

```
cd mi-app
npm install
npm run dev
```

Acceder al proyecto en:

http://localhost:5173 (URL por defecto generada por Vite)

Componentes y Props en React

¿Qué es un componente?

En React, un componente es una función (o clase) que devuelve JSX, el lenguaje de plantillas que combina HTML con JavaScript.

```
function Saludo() {
  return <h1>Hola, mundo</h1>;
}

const Saludo = () => {
  return <h1>Hola, mundo</h1>;
};
```

¿Qué son las props?

Son propiedades que se pasan desde un componente padre hacia uno hijo para que el componente hijo se comporte de forma dinámica.

```
function Saludo(props) {
  return <h1>Hola, {props.nombre}</h1>;
}

// Uso
<Saludo nombre="Lucía" />

const Saludo = ({ nombre }) => {
  return <h1>Hola, {nombre}</h1>;
};

// Uso del componente
<Saludo nombre="Lucía" />
```

Ventajas:

- Reutilización de componentes
- Separación de responsabilidades
- Facilita la composición de UI complejas

¿Qué es SWC?

Speedy Web Compiler

- Compilador moderno y rápido
- Transpila JS moderno y TypeScript a JS compatible con navegadores
- Escrito en Rust (lenguaje seguro y rápido)
- Reemplazo a Babel en proyectos como Vite

¿Por qué es importante?

- Acelera el tiempo de build y recarga en desarrollo
- Soporte nativo para JSX, TypeScript, decoradores, etc.

¿Qué es Tailwind?

Tailwind CSS es un framework utility-first, lo que significa que se compone de **clases pequeñas** como `bg-blue-500`, `text-center`, `p-4`, etc., que se combinan para crear cualquier diseño.

Características clave:

- Enfocado en clases reutilizables
- Compatible con diseño responsivo (`md:`, `lg:`)
- Se puede usar junto a librerías de componentes (ej: Tailblocks)

Ejemplo de uso:

```
<button className="bg-blue-500 text-white px-4 py-2 rounded">
  Enviar
</button>
```

Recursos recomendados:

https://nerdcave.com/tailwind-cheat-sheet

https://tailblocks.cc/