**🛠️ Plan de Estudio JavaScript → React**

**1. Fundamentos de JavaScript (ya lo hemos visto / estamos en esto)**

✅ Variables (let, const, var)  
✅ Tipos de datos primitivos  
✅ Condicionales (if, else, switch)  
✅ Bucles (for, while, for...of, for...in)  
✅ Funciones (function, arrow functions, parámetros, return)  
✅ **Arrays** → métodos importantes (map, forEach, filter, find, reduce, some, every)  
✅ **Objetos** → creación, acceso, desestructuración, anidamientos  
✅ **Manejo del DOM** → document.querySelector, appendChild, createElement, innerHTML  
✅ **Eventos** (addEventListener)  
✅ **fetch API** y consumo de datos (lo que justo terminaste 🚀)

👉 Ahora ya dominas cómo **obtener y mostrar datos** en pantalla.

**2. Asincronía en JavaScript (lo que sigue)**

🔹 Callbacks  
🔹 Promesas (.then, .catch)  
🔹 async / await (ya lo aplicaste con fetch)  
🔹 Manejo de errores con try/catch  
🔹 Promise.all, Promise.race

**Ejercicios:** crear funciones que consulten varias APIs al tiempo, mostrar spinners de carga, etc.

**¿Como lo estudiaremos?**

👉 Lo trabajamos primero porque React depende mucho de manejar APIs y estados asíncronos.

**Temas:**

1. **Callbacks**
   * Qué son, cómo se usaban antes de Promesas.
   * Ejemplo: setTimeout(() => {...}, 1000)
2. **Promesas con .then y .catch**
   * Uso básico con fetch.
   * Diferencia con async/await.
3. **async/await** (ya lo usaste)
   * Refuerzo: combinar varios await y optimizar.
4. **Manejo de errores (try/catch)**
   * Manejo profesional de errores con throw y catch.
5. **Promise.all / Promise.race**
   * Hacer varias consultas al tiempo y manejar la respuesta.

📌 **Ejercicio propuesto:**

* Crear una función que consulte **2 APIs diferentes** (por ejemplo Rick & Morty + Pokémon).
* Mostrar un **spinner de carga** mientras llega la data.
* Manejar el caso de error si una API falla.

**3. Profundización en JS**

🔹 Manipulación avanzada de Arrays y Objetos  
🔹 Spread/rest operator (...)  
🔹 Template literals  
🔹 Destructuring  
🔹 JSON (parse/stringify)  
🔹 LocalStorage / SessionStorage

**Ejercicios:** guardar usuarios en localStorage, recuperar info, simular un login sencillo.

**¿Como lo estudiaremos?**

👉 Aquí vamos a reforzar lo que más se usa en React y backend.

**Temas:**

1. **Manipulación avanzada de Arrays y Objetos**
   * .map, .filter, .reduce, .find, .some, .every.
   * Mutabilidad vs inmutabilidad.
2. **Spread / Rest operator (...)**
   * Copiar, combinar objetos y arrays.
   * Rest en parámetros de funciones.
3. **Template literals (`Hola ${nombre}`)**
   * Reemplazo de concatenaciones.
4. **Destructuring**
   * Arrays y objetos (ej: const {name, age} = persona).
   * Destructuring en parámetros de funciones.
5. **JSON (parse / stringify)**
   * Convertir datos a JSON para enviar al backend.
   * Parsear strings a objetos.
6. **LocalStorage / SessionStorage**
   * Guardar datos de usuario.
   * Recuperarlos al recargar.
   * Simular un login sencillo.

📌 **Ejercicios propuestos:**

* Crear un "mini sistema de usuarios":
  + Registro: guardar nombre y contraseña en localStorage.
  + Login: verificar si coincide con lo guardado.
  + Mostrar un mensaje de "Bienvenido, usuario".

**4. Pilares previos a React**

🔹 Programación orientada a objetos (POO) en JS (clases, herencia)  
🔹 Módulos (import/export)  
🔹 Buenas prácticas y organización de código

**5. Introducción a React**

🔹 JSX (qué es y por qué)  
🔹 Componentes funcionales  
🔹 Props  
🔹 Estado (useState)  
🔹 Eventos en React  
🔹 Listas y .map() en JSX  
🔹 Condicionales en JSX  
🔹 useEffect y ciclo de vida

**6. React Avanzado**

🔹 Context API  
🔹 Custom hooks  
🔹 Manejo de formularios  
🔹 React Router  
🔹 Consumo de APIs en React  
🔹 Autenticación (login con JWT)  
🔹 Buenas prácticas en arquitectura de proyectos React

**7. Extras (cuando ya estés sólido)**

🔹 TypeScript en React  
🔹 Redux o Zustand (según prefieras)  
🔹 Testing (Jest, React Testing Library)  
🔹 Optimización de performance (memo, useCallback, useMemo)