DAY 4 – API Fetching, Async/Await, JSON Handling, Loading/Error States, Thirukkural App & Pokedex App

---

1. What Is an API? (Beginner-Friendly Explanation)

Simple Definition

API = A “bridge” that allows your React app to get data from another server.

Real Life Examples

- Weather app → API gives temperature
- Pokedex → API gives Pokémon data
- Dictionary app → API gives meanings
- Thirukkural app → API gives kurals

Common API Formats

Most APIs send data in **JSON format**.

Example JSON:

```
{
  "name": "Pikachu",
  "type": "Electric",
  "power": 55
}
```

---

2. What Is Fetch?

`fetch()` is a JavaScript function used to call APIs and get data.

Basic example:

```
fetch("https://example.com/api")
```

It returns a **Promise**.

---

3. Understanding Promises (Simple Explanation)

Promise = A box that will contain data **later**.

They have 3 states:

1. Pending (waiting)
2. Resolved (successful)
3. Rejected (failed)

---

4. Async/Await (Modern & Easy Way)

Instead of writing:

```
fetch(url)
  .then(res => res.json())
  .then(data => console.log(data));
```

We do:

```
const response = await fetch(url);
const data = await response.json();
```

Cleaner, simpler, beginner-friendly.

---

5. Complete API Call Example in React

```
useEffect(() => {
  async function fetchData() {
    const response = await fetch("https://api.example.com/data");
    const data = await response.json();
    console.log(data);
  }

  fetchData();
}, []);
```

Why useEffect?

- Ensures API call runs **once** when component loads
- Prevents infinite loops

---

6. Understanding JSON

JSON = JavaScript Object Notation Structure:

```
{
  "id": 1,
  "title": "React Lesson",
  "completed": false
}
```

Accessing JSON in React:

```
console.log(data.title);
```

---

7. Loading & Error States

Whenever calling API:

- Show a loading indicator
- Handle errors if API fails

Setup states

```
const [data, setData] = useState(null);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
```

Inside API Call

```
try {
  setLoading(true);

  const response = await fetch(url);

  if (!response.ok) {
    throw new Error("API failed");
  }

  const result = await response.json();
  setData(result);
} catch (err) {
  setError(err.message);
} finally {
  setLoading(false);
}
```

UI Handling

```
if (loading) return <p>Loading...</p>;
if (error) return <p>Error: {error}</p>;
```

---

8. Conditional Rendering for API

Examples:

```
{loading && <p>Loading…</p>}
{error && <p>Error occurred</p>}
{data && <p>{data.title}</p>}
```

---

9. Rendering List From API

Example JSON:

```
[
  { "id": 1, "name": "Pikachu" },
  { "id": 2, "name": "Charizard" }
]
```

React:

```
data.map(pokemon => (
  <p key={pokemon.id}>{pokemon.name}</p>
))
```

---

10. Project 1 — Thirukkural App (Full Explanation)

API:

```
https://api-thirukkural.vercel.app/api?num=1
```

Step-by-Step Implementation

State Setup

```
const [kural, setKural] = useState(null);
const [loading, setLoading] = useState(false);
```

Fetch Function

```
async function getKural(number) {
  setLoading(true);

  const res = await fetch(
    `https://api-thirukkural.vercel.app/api?num=${number}`
  );

  const data = await res.json();
  setKural(data);

  setLoading(false);
}
```

Input for user to enter number

```
<input
  type="number"
  value={num}
  onChange={(e) => setNum(e.target.value)}
/>
<button onClick={() => getKural(num)}>Get Kural</button>
```

Display Kural

```
{kural && (
  <div>
```

```
    <h2>{kural.season}</h2>
    <p>{kural.line1}</p>
    <p>{kural.line2}</p>
    <h4>Meaning: {kural.tam_exp}</h4>
  </div>
)}
```

---

11. Project 2 — Pokédex App (Full Explanation)

API:

```
https://pokeapi.co/api/v2/pokemon/pikachu
https://pokeapi.co/api/v2/pokemon/1
```

Step-by-Step Implementation

States

```
const [pokemon, setPokemon] = useState(null);
const [name, setName] = useState("");
```

Function to Fetch Pokémon

```
async function fetchPokemon() {
  const res = await fetch(`https://pokeapi.co/api/v2/pokemon/${name}`);
  const data = await res.json();
  setPokemon(data);
}
```

Input Field

```
<input
  value={name}
  onChange={(e) => setName(e.target.value)}
  placeholder="Enter Pokémon name"
/>
<button onClick={fetchPokemon}>Search</button>
```

Displaying Pokémon Data

```
{pokemon && (
  <div>
    <h2>{pokemon.name.toUpperCase()}</h2>
    <img src={pokemon.sprites.front_default} />
    <p>Height: {pokemon.height}</p>
    <p>Weight: {pokemon.weight}</p>

    <h4>Types:</h4>
    {pokemon.types.map(t => (
      <span key={t.type.name}>{t.type.name} </span>
    ))}
  </div>
)}
```

---

12. Important API Errors Students Face & Solutions

❌ CORS error

The API doesn't allow browser access → Must use a different API or proxy.

❌ Unexpected token

JSON format incorrect → maybe using wrong endpoint.

❌ Cannot read property of undefined

Object not loaded yet → use conditional rendering.

Correct:

```
pokemon && pokemon.name
```
