Title: Introduction to API and Methods in JavaScript

✅What is an API?

API stands for **Application Programming Interface**.

⬜Simple Explanation:

An API is like a waiter in a restaurant:

- You (the user) ask the waiter (API) for food (data).
- The waiter takes your request to the kitchen (server).
- The kitchen prepares the food (data).
- The waiter returns with the food (response).

You don't need to know how the kitchen works — the waiter (API) handles that. Similarly, in programming, you use an API to communicate with another service without knowing how it works internally.

✅Why Do We Use APIs?

We use APIs to:

- Get or send data to/from a server
- Use features from another application (like Google Maps, YouTube, Weather)
- Communicate between different systems or software

✅General Example of Using an API

Imagine you are building a travel booking site. When a user selects a city and dates, your site:

1. Uses a **hotel booking API** to fetch available hotels.
2. Uses a **flight API** to get available flights.
3. Uses a **weather API** to show the weather forecast.

You don't store hotel or flight data — instead, your app communicates with these APIs to get the needed data and show it to users.

✅How to Use APIs in JavaScript?

In JavaScript, the most common way to use APIs is with the built-in `` method.

Basic Syntax:

```
fetch(url)
  .then(response => response.json())
  .then(data => {
    // use the data
  })
  .catch(error => {
```

✅Simple Example: Get Users from Public API

Code:

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())
  .then(data => {
    console.log("User List:", data);
  })
  .catch(error => {
```

What it Does:

- Sends a request to a free API that returns dummy users.
- Converts the response to readable JSON.
- Logs the user data.
- Handles any errors.

✅Methods Used with API in JavaScript

1. ``

Used to make API requests.

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

2. ``

Runs when the request is successful.

```
.then(response => response.json())
.then(data => console.log(data))
```

3. ``

Runs when there is an error in the request.

```
.catch(error => console.log("Error:", error))
```

4. HTTP Methods Used with ``

Method	Use	Example Purpose
GET	Get data	View user list
POST	Send data	Add a new user
PUT	Update full data	Replace user info
PATCH	Update part	Update just the name
DELETE	Remove data	Delete a user

✅POST Example: Send Data to an API

```
fetch("https://api.example.com/posts"
  , { method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body:
    JSON.stringify({
    title: "Hello",
    body: "This is a post.",
    userId: 1
```

✅Summary Table

Term	Meaning
API	Connects your code to other services/data
fetch()	Sends a request to the API
.then()	Runs after successful response
.catch()	Runs if there is an error