RESTful APIs

What is an **API**?

"API" stands for "Application Programming Interface."

APIs are like bridges that allow different software systems to communicate and interact with each other

APIs enable data exchange and functionality between software applications.

APIs promote interoperability and allow businesses to leverage external services.

APIs are fundamental to the modern digital ecosystem.

REST stands for Representational State Transfer.

RESTful APIs are a popular and widely used type of API design.

RESTful APIs follow specific principles and guidelines that make them efficient and scalable.

REST is not a technology or protocol but an architectural style

The foundational concepts of REST:

- Resources: Everything in a RESTful system is treated as a resource (e.g., web pages, data, images).
- Representations: Resources can have multiple representations (e.g., HTML, JSON, XML).
- Statelessness: The statelessness principle, where each request from a client to a server must contain all the information needed to understand and process the request.

RESTful APIs often map HTTP methods to CRUD (Create, Read, Update, Delete) operations:

- GET for reading data.
- POST for creating new data.
- PUT for updating existing data.
- DELETE for removing data.

HTTP Methods in RESTful APIs

HTTP methods are a fundamental part of RESTful API interaction.

HTTP methods are used to perform various actions on resources

GET Method:

- GET is used to retrieve data from the server.
- It is safe and idempotent, meaning repeated requests produce the same result.
- GET requests do not modify server data.

POST Method:

- POST is used to create new resources on the server.
- It sends data to the server in the request body.
- It is not idempotent, as repeated POST requests may create multiple resources.

PUT Method:

- PUT is used to update existing resources on the server.
- It sends data to the server in the request body to replace or modify the resource.
- It is idempotent, as repeated PUT requests with the same data have the same result.

DELETE Method:

- DELETE is used to remove resources from the server.
- It does not typically have a request body since it specifies the resource to delete in the URL.
- It is idempotent, as deleting a resource multiple times has the same result.

While GET, POST, PUT, and DELETE are the primary methods, there are other less commonly used methods like PATCH, HEAD, and OPTIONS that serve specific purposes.

PATCH is used to apply partial updates to a resource, meaning that only the fields that need to be changed are sent in the request body. PUT is used to replace the entire resource with a new representation, meaning that all the fields of the resource are sent in the request body, even if they are not modified.

HTTP status codes in the context of RESTful APIs.

The status codes provide information about the outcome of an API request.

the concept of HTTP status code categories:

- Informational (1xx): Used for provisional responses.
- Success (2xx): Indicates a successful request.
- Redirection (3xx): Indicates a redirection.
- Client Errors (4xx): Indicates a client-side error.
- Server Errors (5xx): Indicates a server-side error.

The most common HTTP status codes and their meanings:

- 200 OK: The request was successful, and the server returns the requested data.
- 201 Created: The request was successful, and a new resource was created.
- 400 Bad Request: The request is invalid or missing required data.
- 401 Unauthorized: Authentication is required or credentials are invalid.
- 403 Forbidden: The client does not have permission to access the resource.
- 404 Not Found: The requested resource does not exist.
- 500 Internal Server Error: A server error occurred while processing the request.

Making API Requests with Fetch API

The global `fetch()` method starts the process of fetching a resource from the network, returning a promise which is fulfilled once the response is available.

Fetch API is a modern web API for making HTTP requests

- it's built into most modern web browsers, making it a convenient choice for web developers.

Advantages over traditional XMLHttpRequest

- The Fetch API's simplicity, promise-based approach, and improved error handling compared to older methods like XMLHttpRequest.

Basic Fetch API Syntax

`fetch(resource)`

`fetch(resource, options)`

A `fetch()` promise *does not* reject on HTTP errors (`404`, etc.). Instead, a `then()` handler must check the `Response.ok` and/or `Response.status` properties.

```
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    const data = await response.json();
    // Process data
  } catch (error) {
    console.error(`Error: ${error.message}`);
    // Handle API request or processing errors
  }
}
```