




Welcome to the CoGrammar Lecture: Fetch API

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.



Full Stack Web Development Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
(Fundamental British Values: Mutual Respect and Tolerance)
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

Full Stack Web Development Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

Skills Bootcamp

8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

Due Date: 24 March 2024

✓ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

Due Date: 28 April 2024

Skills Bootcamp Progression Overview

✓ Criterion 3: Course Progress

Completion: All mandatory tasks,
including Build Your Brand and
resubmissions by study period end
Interview Invitation: Within 4 weeks
post-course
Guided Learning Hours: Minimum of
112 hours by support end date
(10.5 hours average, each week)

✓ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship
Outcome: Document within 12
weeks post-graduation
Relevance: Progression to
employment or related
opportunity

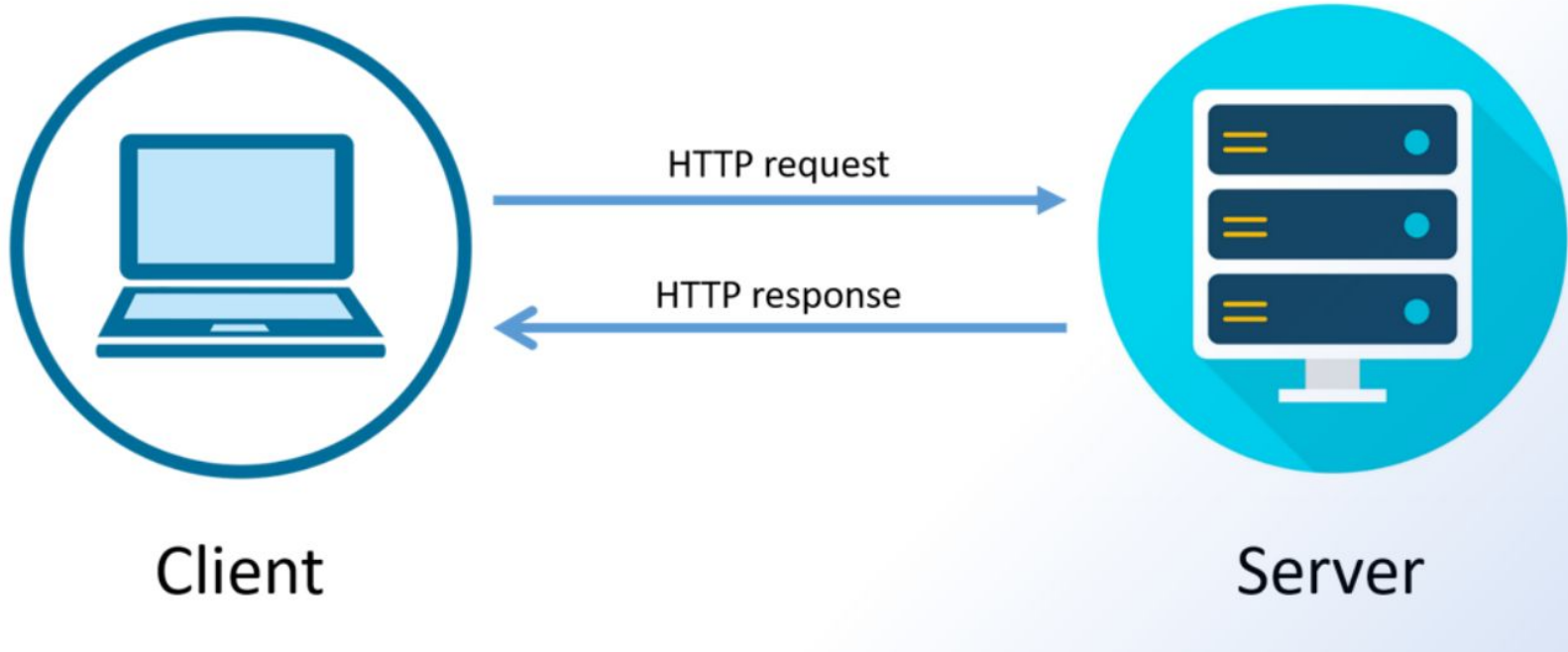
Agenda

- ❖ Define what an API is and explain its importance in web development.
- ❖ Demonstrate how to make various types of HTTP requests (GET, POST, PUT, DELETE) using the Fetch API in JavaScript.
- ❖ Recall the syntax and basic usage of the Fetch API for making HTTP requests.

Introduction to Web Requests

- ❖ Web requests, also known as HTTP requests, are the foundation of communication between clients (such as web browsers) and servers over the World Wide Web.
- ❖ They facilitate the exchange of data and resources, allowing users to access and interact with web applications, websites, and services.
- ❖ HTTP (Hypertext Transfer Protocol) is the underlying protocol used for sending and receiving web requests and responses.
- ❖ It operates as a request-response protocol, where clients send requests to servers, and servers respond with the requested resources or information.

Introduction to Web Requests



Components of an HTTP Request

- ❖ An HTTP request consists of several components, including:
 - Request Method (e.g., GET, POST, PUT, DELETE)
 - The request method indicates the desired action to be performed on the specified resource.
 - URL (Uniform Resource Locator) or URI (Uniform Resource Identifier)
 - The URL or URI specifies the location of the resource being requested by the client.
 - Headers (optional metadata)
 - Body (optional data for POST and PUT requests)

Components of an HTTP Response

- ❖ An HTTP response consists of several components, including:
 - Status Code (e.g., 200 OK, 404 Not Found)
 - The status code indicates the outcome of the server's attempt to fulfill the client's request.
 - Headers (metadata): HTTP headers provide metadata about the response, including information about the server, content type, content length, caching directives, and more.
 - Body (response data): The response body contains the actual data or resource requested by the client, such as HTML content, JSON data, images, or files.

Common Request Methods

- ❖ GET: Retrieves data from the server.
- ❖ POST: Submits data to the server to create or update a resource.
- ❖ PUT: Updates an existing resource on the server.
- ❖ DELETE: Removes a resource from the server.

How Web Requests Work

- ❖ The client (e.g., web browser) sends an HTTP request to the server, specifying the desired action and resource.
- ❖ The server receives the request, processes it, and generates an appropriate response based on the request method and resource availability.
- ❖ The server sends an HTTP response back to the client, containing the requested data or indicating the outcome of the request (success, error, redirection, etc.).

What are APIs?

- ❖ An API (Application Programming Interface) allows different software systems to communicate with each other.
- ❖ It defines the methods and data formats that applications can use to request and exchange information.
- ❖ APIs are fundamental for web development, enabling interaction with remote servers and services.

Overview of Fetch API

- ❖ The Fetch API is a modern JavaScript interface for making asynchronous HTTP requests.
- ❖ Fetch follows a promise-based approach, which simplifies the handling of asynchronous operations, making code more readable and maintainable.
- ❖ Fetch returns a Promise that resolves to a Response object, allowing for seamless chaining of asynchronous operations using `.then()` and `.catch()` methods.

Let's Breathe!

Let's take a small break
before moving on to
the next topic.



fetch() Method

- ❖ The fetch() function is used to initiate a request to the specified url.
- ❖ It returns a promise that resolves to the Response object representing the response to the request.
- ❖ The options parameter is an optional object containing settings for the request such as method, headers, body, etc.

```
fetch(url, options)
  .then((response) => {
    // handle response
  })
  .catch((error) => {
    // handle error
  });
```


fetch() Method Parameters

- ❖ URL (required):
 - Specifies the URL to which the request is made.
- ❖ Options (optional): An object containing various settings for the request such as:
 - method: HTTP method (GET, POST, PUT, DELETE, etc.)
 - headers: Headers to include in the request
 - body: Data to send with the request (e.g., JSON, FormData)

Response Handling

- ❖ Response Object:
 - Represents the response to the request made by `fetch()`.
 - Contains properties and methods to access response data and metadata.
- ❖ `.json()`: Parses the response body as JSON.

Making a GET Request

```
fetch("https://jsonplaceholder.typicode.com/posts")  
  .then((response) => response.json())  
  .then((data) => console.log(data))  
  .catch((error) => console.error("Error:", error));
```

Making a POST Request

```
const postData = {
  username: "example",
  password: "password123",
};

fetch("https://api.example.com/login", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(postData),
})
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error("Error:", error));
```

Making a DELETE Request

```
fetch("https://api.example.com/user/123", {  
  method: "DELETE",  
})  
  .then((response) => {  
    if (response.ok) {  
      console.log("User deleted successfully");  
    } else {  
      console.error("Failed to delete user");  
    }  
  })  
  .catch((error) => console.error("Error:", error));
```


Questions and Answers



Thank you for attending



Department
for Education

CoGrammar

