Intro to HTTP Requests in JS

Moeein Aali
Sharif University of Technology
IE060 - Advanced Programming - Fall 2024
Dr. Habibi & Dr. Nabavian

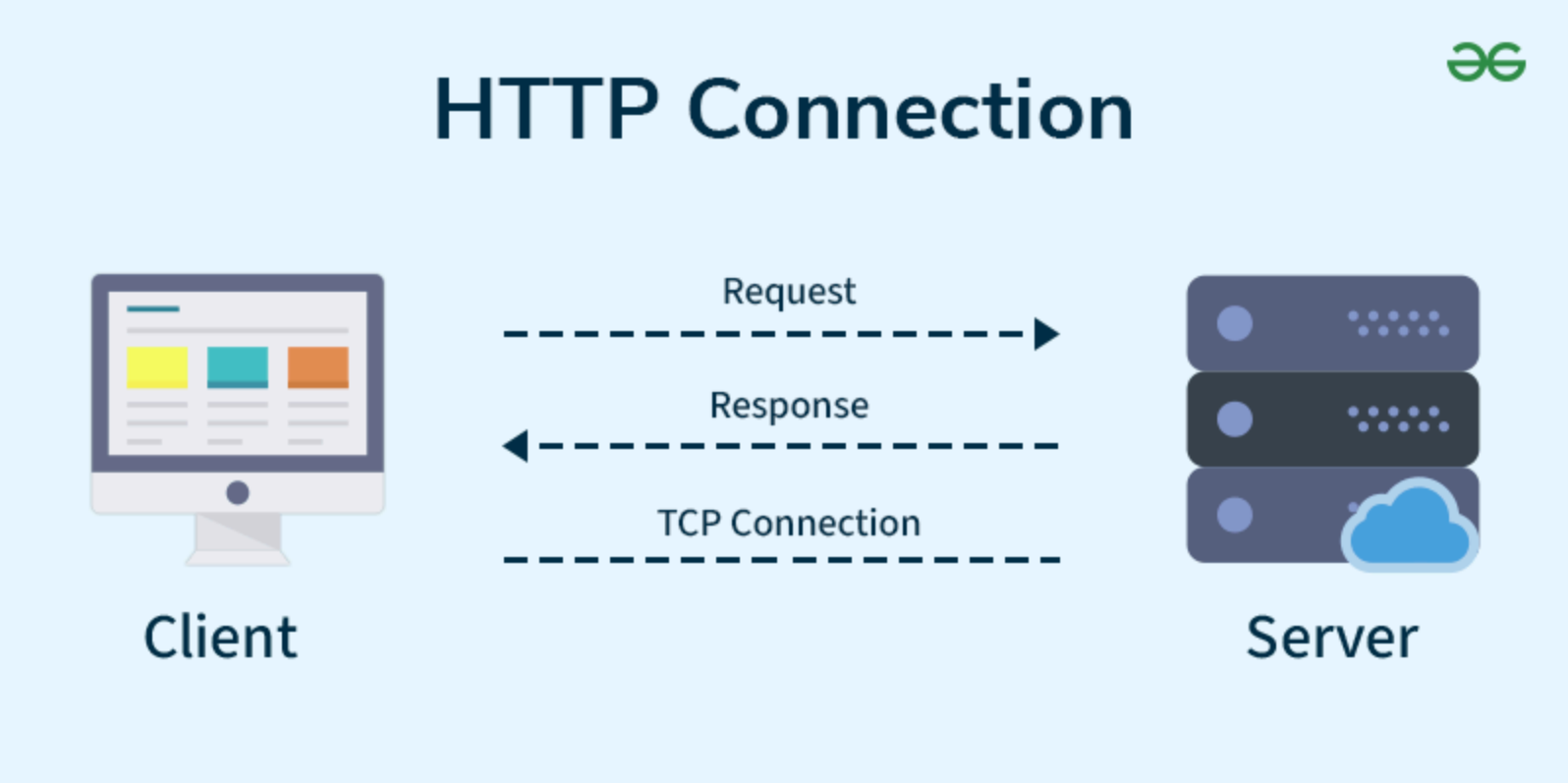
Agenda

1. **Introduction to HTTP**
    - What is HTTP?
    - Key concepts: Request-Response Model
2. **Overview of HTTP Methods**
    - GET, POST, PUT, DELETE, PATCH
3. **Making HTTP Requests in JavaScript**
    - Fetch API
    - Axios
4. **Some Useful Tools**
    - Inspect
    - Postman

What is HTTP?

- **HyperText Transfer Protocol (HTTP)**: Foundation of communication on the web.
- Works as a **request-response model** between a client and a server.
- Key components of an HTTP transaction:
  - **Request**: Method, URL, Headers, Body
  - **Response**: Status Code, Headers, Body

What is HTTP?



- What is HTTP?
- HTTP Full Form

HTTP Methods

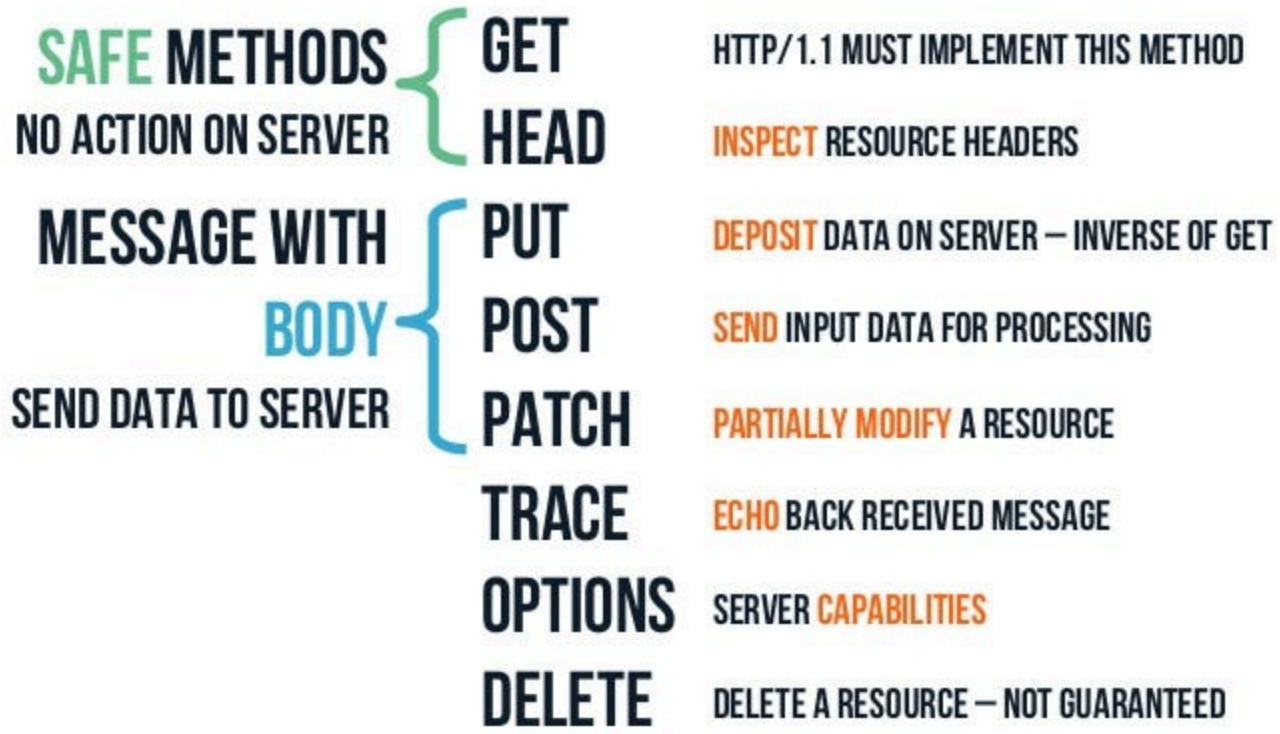
- **GET**: Retrieve data from a server.
  Example: Fetching user data.
- **POST**: Send data to the server to create a new resource.
  Example: Submitting a form.
- **PUT**: Update an existing resource on the server.
  Example: Editing user information.
- **PATCH**: Partially update a resource on the server.
  Example: Updating a single field of user data.
- **DELETE**: Remove a resource from the server.
  Example: Deleting an account.

HTTP Methods: Headers and Body

HTTP Method	Requires Headers?	Requires Body?	Use Case Example
GET	Optional	No	Retrieve data from a server.
POST	Yes	Yes	Create a new resource.
PUT	Yes	Yes	Update an entire resource.
PATCH	Yes	Yes	Partially update a resource.
DELETE	Optional	Optional	Remove a resource from server.

- **Headers**: Used for authentication, content type, etc.
- **Body**: Contains the data to send (for methods like POST, PUT, PATCH).

HTTP Methods: Safe Methods



JavaScript and HTTP Requests

Some Methods for Calling APIs in JavaScript:

1. **Fetch API**:
   - A modern, built-in browser API for making HTTP requests.
   - Example: `fetch(url).then(response => response.json())`
2. **Axios**:
   - A third-party library with simpler syntax and robust features.
   - Example: `axios.get(url).then(response => console.log(response))`
3. **jQuery AJAX**:
   - A method provided by the jQuery library for making HTTP requests.

Fetch API - GET Request

```
const url = 'https://jsonplaceholder.typicode.com/posts';

async function fetchData() {
  try {
    const response = await fetch(url);
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error('Error:', error);
  }
}
```

Fetch API - GET Request

```
const url = 'https://jsonplaceholder.typicode.com/posts';

fetch(url)
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Fetch API - POST Request

```
const url = 'https://jsonplaceholder.typicode.com/posts';
const data = {
  title: 'New Post',
  body: 'This is a new post.',
  userId: 1,
};

async function createPost() {
  try {
    const response = await fetch(url, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(data),
    });

    const result = await response.json();
    console.log('Created:', result);
  } catch (error) {
    console.error('Error:', error);
  }
}
```

Fetch API - POST Request

```
const url = 'https://jsonplaceholder.typicode.com/posts';
const bodyData = {
  title: 'New Post',
  body: 'This is a new post.',
  userId: 1,
};

fetch(url, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(bodyData),
})
    .then(response => response.json()).then(data => console.log('Created:', data))
    .catch(error => console.error('Error:', error));
```

Axios: Install

1. Using CDN (Content Delivery Network)

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

2. Using npm/yarn (Node.js Environment)

```
npm install axios
# or
yarn add axios
```

- Import Axios in your JavaScript file:

```
import axios from 'axios';
```

- **Using Require (CommonJS)**

```
const axios = require('axios');
```

Axios: GET Request

```
const url = 'https://jsonplaceholder.typicode.com/posts';

axios.get(url)
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

Axios: POST Request

```
const url = 'https://jsonplaceholder.typicode.com/posts';
const headers = {
    'Content-Type':'Application/json'
}
const data = {
  title: 'New Post',
  body: 'This is a new post.',
  userId: 1,
};

axios.post(url, data, headers)
  .then(response => console.log('Created:', response.data))
  .catch(error => console.error('Error:', error));
```

Fetch API vs. Axios

Feature	Fetch API	Axios
Built-in	Yes, native to modern browsers	No, requires installation
Promise-based	Yes	Yes
Error Handling	Manual (`response.ok`)	Automatic for non-2xx responses
JSON Handling	Requires `response.json()`	Automatic for JSON
Timeouts	Requires custom implementation	Built-in support
File Uploads	Requires manual setup	Simplified
Browser Compatibility	Modern browsers only	Older browsers (with polyfills)

Inspect Tool Tabs Overview

Tab	Description
Elements	View and edit HTML and CSS in real-time.
Console	Debug JavaScript, view errors, and log messages.
Sources	Access and debug JavaScript and source files, set breakpoints.
Network	Monitor network requests, response times, and performance metrics.
Application	Manage storage (cookies, localStorage, sessionStorage) and inspect resources.
...	...

Introduction to Postman

What is Postman?

- A powerful API development and testing tool.
- Simplifies interaction with RESTful, SOAP, and GraphQL APIs.
- Provides a graphical interface for crafting, testing, and debugging API requests.

- Postman documentation overview

Introduction to Postman

Key Features

- **Request Building**: Easily create and send HTTP requests (GET, POST, PUT, DELETE, etc.).
- **Testing & Automation**: Write tests using JavaScript and automate API workflows.
- **Collaboration**: Share collections and environments with your team.
- **Documentation**: Generate and publish API documentation directly from your collections.
- **Mock Servers**: Simulate APIs for development and testing.
- **Monitoring**: Track API performance over time.

Introduction to Postman

Why Use Postman?

- **Ease of Use**: Intuitive interface with minimal setup.
- **Efficiency**: Saves time with reusable collections and environments.
- **Integration**: Works seamlessly with CI/CD pipelines and version control.