

Topics

Thursday, October 31, 2024

12:07 PM

- 1. Client-Server model**
- 2. HTTP Request and Response**
- 3. HTTP Methods**
- 4. Status Codes**
- 5. Web Technologies**
 - *HTML*
 - *CSS*
 - *JavaScript*

Client-Server model/architecture

Thursday, October 31, 2024 12:14 PM

- The Client-Server model is a fundamental design framework for networked applications.
- It organizes interactions between two major entities: **clients** (requesters) and **servers** (providers of resources).
- This architecture underpins most modern networks, including the internet, web applications, email systems, and various enterprise systems.

Client:

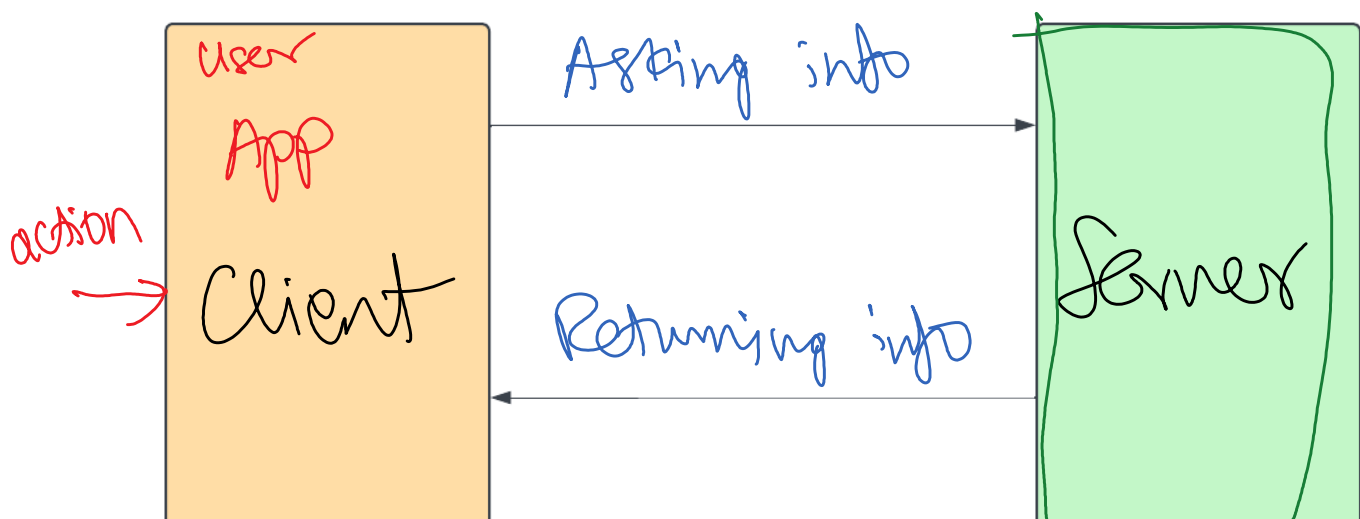
- Device or an application that initiates requests for services or resources.
- Clients are typically end-user devices (e.g., smartphones, laptops) or software applications (e.g., browsers, email clients) that communicate over a network.

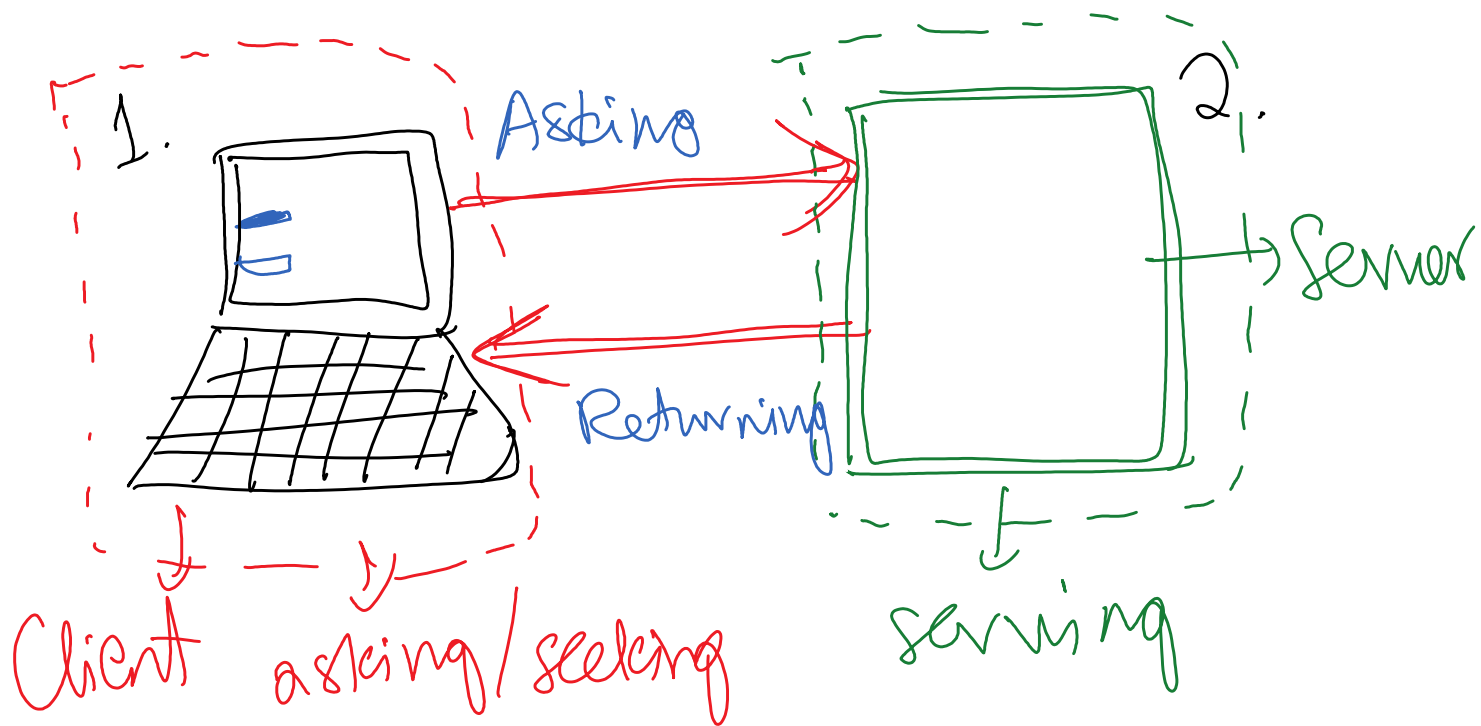
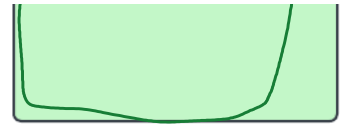
Server:

- A server is a dedicated system or application that listens for and fulfills requests from clients.
- Servers provide resources, data, or services to clients, typically through a network connection.

Working:

- *importance of server for websites/apps*
- *client-server communication*





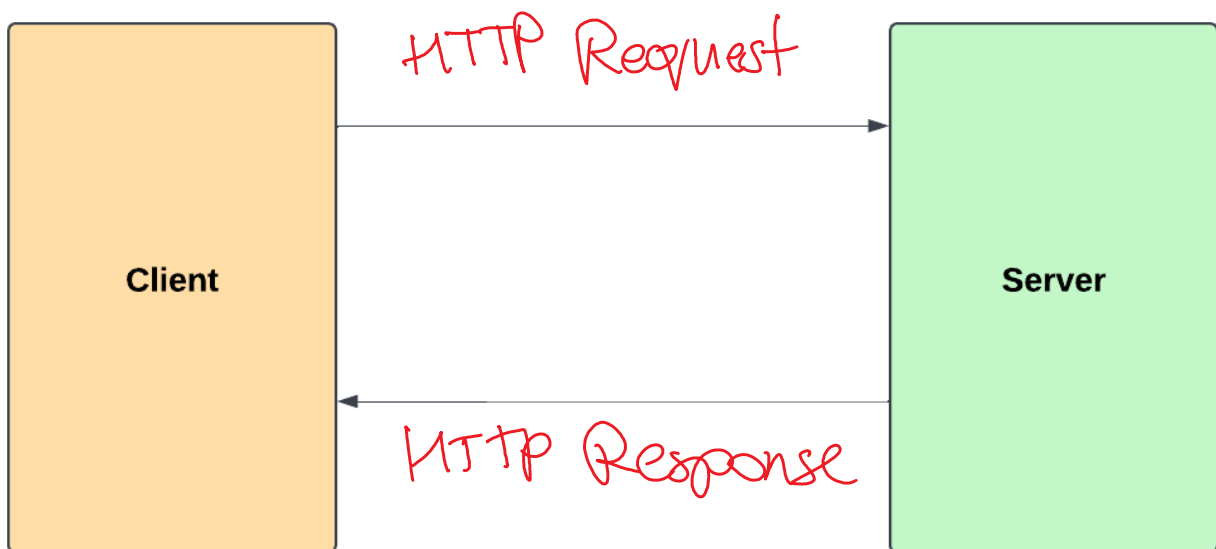
HTTP Request and Response

Thursday, October 31, 2024 12:14 PM

HTTP:

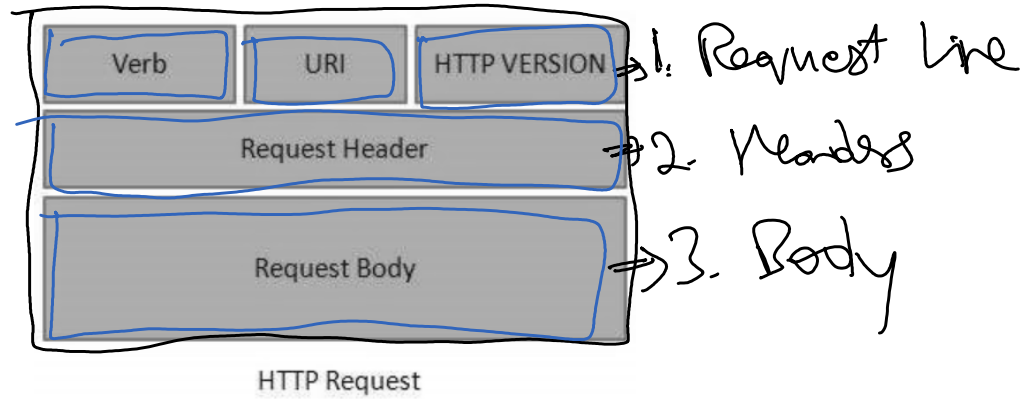
- The HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the web.
- It facilitates the exchange of information between clients and servers.
- The HTTP request-response cycle is central to how web applications function.

Working:



HTTP Request:

- An HTTP request is a message sent by the client to the server to initiate an action or request a resource.
- The request message consists of following components:
 - **Request Line:**
 - Method / Verb
 - Address (URI)
 - Version
 - **Headers:** used for conveying additional meta-data about the request
 - **Body:** contains the major contents of the request message



- Example Request message:

```

POST /api/login HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Accept: application/json
Content-Type: application/json
Content-Length: 55

{
  "username": "john_doe",
  "password": "securepassword123"
}

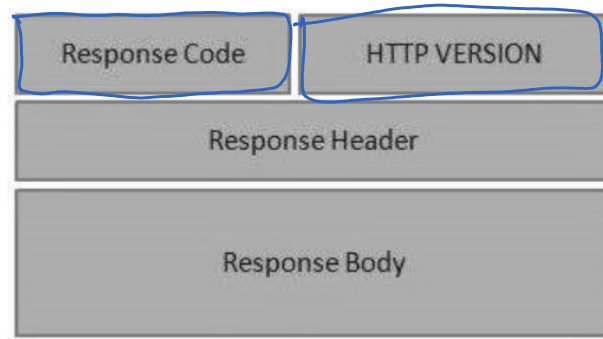
```

Handwritten annotations:

- Request Line:** Points to the first line: `POST /api/login HTTP/1.1`
- Headers:** Points to the lines between `Host:` and `Content-Length:`
- Body:** Points to the JSON object in the body.

HTTP Response:

- An HTTP response is the message sent by the server back to the client after processing the request.
- The response message consists of following components:
 - **Response Line:**
 - Status Code
 - Version
 - **Headers:** used for conveying additional meta-data about the response
 - **Body:** contains the requested contents (usually JSON format)



HTTP Response

⇒ 1. Response line

⇒ 2. Headers

⇒ 3. Body

- Example Response message:

1. Line

2. Headers

3. Body

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 55
Cache-Control: no-cache

{
  "success": true,
  "message": "Login successful."
}
```

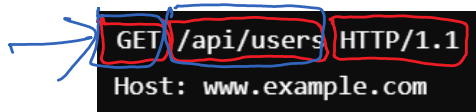
HTTP Methods

Thursday, October 31, 2024 12:14 PM

- HTTP methods, also known as HTTP verbs, are a fundamental part of the HTTP protocol.
- They define the action to be performed on a resource identified by a URI.
- Each method has specific semantics and is used for different purposes in client-server communication.

1. GET:

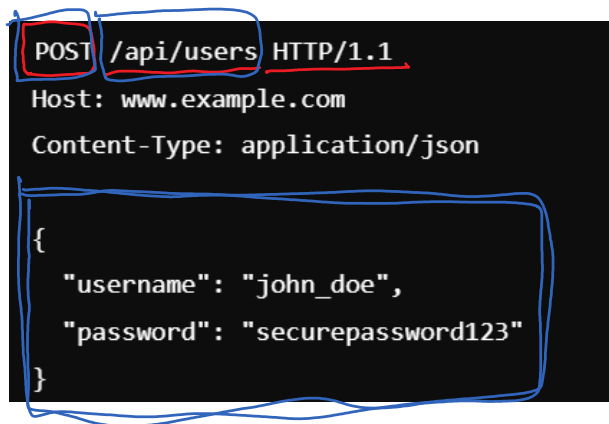
- Used to request data from a specified resource.
- It is the most commonly used HTTP method.
- **Application:** Retrieving web pages, images, or other resources from a server.



A diagram showing an HTTP GET request. The text "GET /api/users HTTP/1.1" is displayed on a black background. The word "GET" is enclosed in a red box, and the URI "/api/users" is enclosed in a blue box. A blue arrow points from the left towards the "GET" method. Below the request line, the text "Host: www.example.com" is shown.

2. POST:

- Used to submit data to be processed to a specified resource.
- This often results in the creation of a new resource.
- **Application:** Submitting forms or uploading files

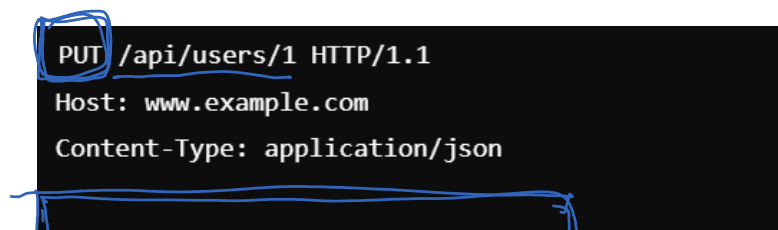


A diagram showing an HTTP POST request. The text "POST /api/users HTTP/1.1" is displayed on a black background. The word "POST" is enclosed in a red box, and the URI "/api/users" is enclosed in a blue box. Below the request line, the text "Host: www.example.com" and "Content-Type: application/json" are shown. A large blue box encloses the JSON body:

```
{
  "username": "john_doe",
  "password": "securepassword123"
}
```

3. PUT:

- Used to update an existing resource or create a new resource if it does not exist.
- It sends data to the server to replace the current representation of the resource.
- **Application:** Updating user details or replacing an entire user.



A diagram showing an HTTP PUT request. The text "PUT /api/users/1 HTTP/1.1" is displayed on a black background. The word "PUT" is enclosed in a red box, and the URI "/api/users/1" is enclosed in a blue box. Below the request line, the text "Host: www.example.com" and "Content-Type: application/json" are shown. A blue box encloses the start of the JSON body, which is partially cut off at the bottom.

```
Content-Type: application/json

{
  "username": "john_doe",
  "email": "john_doe@example.com"
}
```

4. PATCH:

- Used to apply partial modifications to a resource.
- It sends a set of instructions to update the resource rather than replacing it entirely.
- **Application:** Updating specific fields of a user, like changing a user's email without altering other attributes.

```
PATCH /api/users/1 HTTP/1.1
Host: www.example.com
Content-Type: application/json

{
  "email": "john_updated@example.com"
}
```

5. DELETE:

- Used to remove a specified resource from the server.
- **Application:** Deleting user accounts, posts, or other resources.

```
DELETE /api/users/1 HTTP/1.1
Host: www.example.com
```

Summary of HTTP Methods:

METHOD	PURPOSE	USE CASES
GET	Retrieve data	Fetching pages or resources
POST	Submit data to create/update	Submitting forms, creating data
PUT	Update/replace resource	Updating existing resources
PATCH	Partially update resource	Modifying specific fields
DELETE	Remove a resource	Deleting resources
HEAD	Retrieve headers only	Checking resource metadata
OPTIONS	Describe communication options	Discovering server capabilities
TRACE	Diagnostic, echo the received request	Debugging requests

HTTP Status Codes

Thursday, October 31, 2024 12:14 PM

- HTTP status codes are three-digit numbers sent by a server in response to a client's request made to the server.
- These codes are crucial for understanding the results of HTTP requests.
- They provide important feedback to clients about the success or failure of requests and help developers diagnose issues.
- Properly using and interpreting HTTP status codes is essential for effective communication between clients and servers in the web ecosystem.

<u>1xx</u>	Informational
<u>2xx</u>	Success
<u>3xx</u>	Redirection
<u>4xx</u>	Client Error
<u>5xx</u>	Server Error

1. 1xx - Informational:

- These codes indicate that the request has been received and the process is continuing.
- They are rarely used in web applications but are important for certain protocols.
- **Examples:**
 - *100 Continue*
 - *101 Switching Protocols*

2. 2xx - Success:

- These codes indicate that the client's request was successfully received, understood, and accepted.
- **Examples:**
 - 200 OK
 - *201 Created*
 - *202 Accepted*

3. 3xx - Redirection:

- These codes indicate that further action is needed to complete the request.

- This usually involves redirection to another URI.
- **Examples:**
 - *300 Multiple Choices*
 - *301 Moved Permanently*
 - *302 Found*

4. 4xx - Client Error:

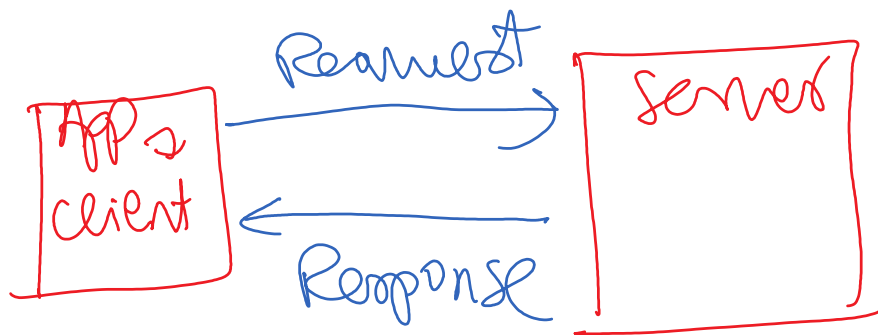
- These codes indicate that the client made an error, resulting in the request being unable to be fulfilled.
- **Examples:**
 - *400 Bad Request*
 - *401 Unauthorized*
 - *404 Not Found*

5. 5xx - Server Error:

- These codes indicate that the server failed to fulfill a valid request due to an error on the server side.
- **Examples:**
 - *502 Bad Gateway*
 - *503 Service Unavailable*

Summary of HTTP Status Codes:

Status Code	Description	Category
100	Continue	Informational
200	OK	Success
201	Created	Success
204	No Content	Success
300	Multiple Choices	Redirection
301	Moved Permanently	Redirection
302	Found	Redirection
400	Bad Request	Client Error
401	Unauthorized	Client Error
403	Forbidden	Client Error
404	Not Found	Client Error
500	Internal Server Error	Server Error
503	Service Unavailable	Server Error



- Web technologies encompass a wide array of tools, languages, and protocols used to create, maintain, and manage websites and web applications.
- HTML provides the foundational structure, CSS handles visual presentation, and JavaScript adds dynamic behavior.
- Together, they enable developers to build rich, interactive user experiences on the web.
- Understanding these technologies is crucial for web scraping, as it allows developers to extract data from web pages effectively, even when dealing with dynamic content.

1. HTML:

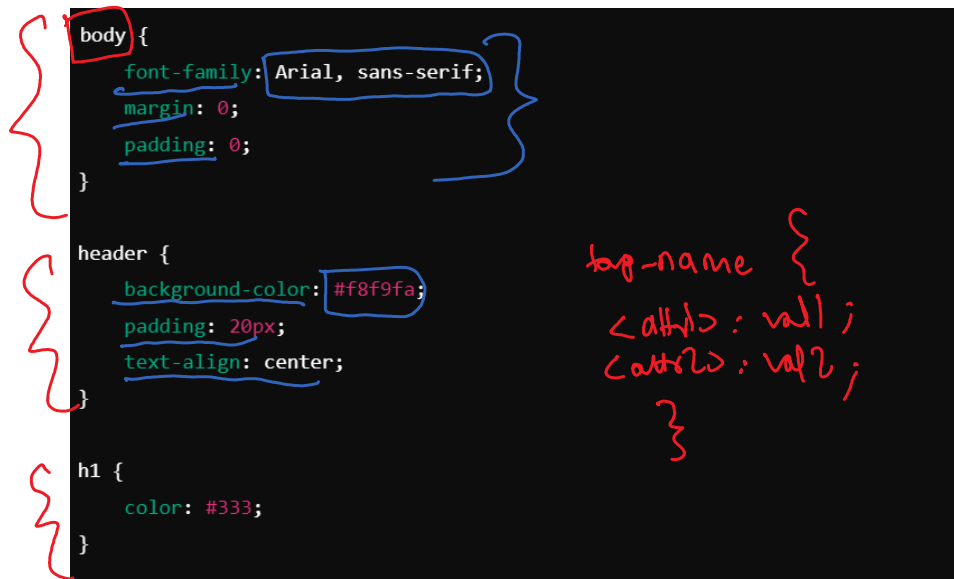
- HTML is the standard markup language used for creating web pages.
- It provides the structure and layout of a web document by defining elements like headings, paragraphs, links, images, and other types of content using appropriate tags.
- Tags can also be nested to create more complex structures.

```
<html>
<head>
  <title>My Web Page</title>
</head>
<body>
  <header>
    <h1>Welcome to My Website</h1>
  </header>
  <section>
    <p>This is a paragraph of text on my website.</p>
    <a href="https://example.com">Visit Example.com</a>
  </section>
  <footer>
    <p>&copy; 2024 My Website</p>
  </footer>
</body>
</html>
```

2. CSS:

- CSS is a stylesheet language used to describe the presentation of a document written in HTML.
- It controls the layout, colors, fonts, and overall visual appearance of web pages.

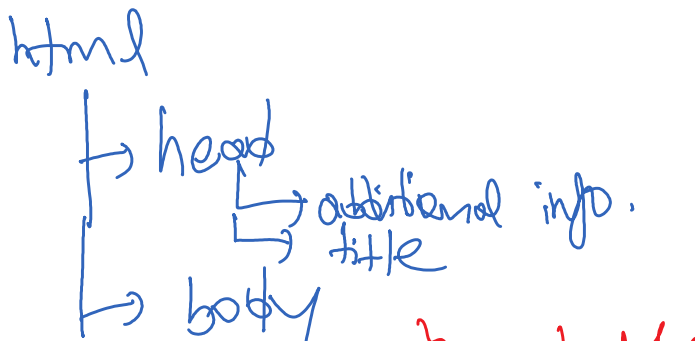
- CSS is a stylesheet language used to describe the presentation of a document written in HTML.
- It controls the layout, colors, fonts, and overall visual appearance of web pages.
- CSS uses **selectors** to target HTML elements and apply styles.



3. JavaScript:

- JavaScript is a high-level, dynamic programming language that enables **interactivity** and functionality on web pages.
- It allows developers to create rich user experiences through client-side scripting.

```
document.addEventListener("DOMContentLoaded", function() {
  const button = document.querySelector("button");
  button.addEventListener("click", function() {
    alert("Button clicked!");
  });
});
```



↳ body ^{title}
↳ paragraph
↳ link
↳ image
↳ form
table
video

html