Module 3: Building APIs and Web Services

HTTP Basics

1. Overview of HTTP Protocol

What is HTTP?

HTTP (HyperText Transfer Protocol) is an application-layer protocol that defines how messages are formatted and transmitted between web clients and servers. It serves as the foundation of data communication on the World Wide Web. Understanding its structure, methods, and status codes is essential for web development and API design. Each HTTP method serves a specific purpose, from retrieving data with GET to creating resources with POST. Status codes provide standardized communication about request outcomes, enabling robust error handling and user feedback. Modern applications leverage HTTP's flexibility while adhering to RESTful principles to create scalable and maintainable systems.

Key Characteristics

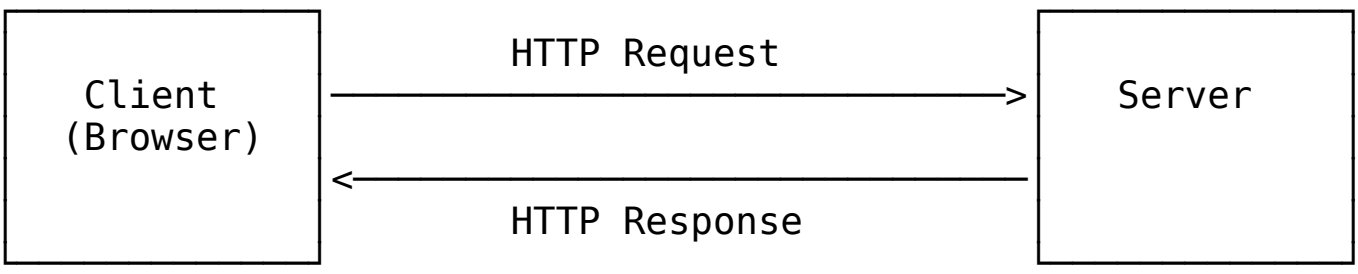
Stateless Protocol: Each request-response pair is independent. The server doesn't retain information about previous requests from the same client by default.

Client-Server Model: HTTP follows a request-response pattern where clients (browsers, mobile apps) initiate requests and servers provide responses.

Text-Based Protocol: HTTP messages are human-readable text, making debugging and analysis straightforward.

Port Usage: By default, HTTP uses port 80, while HTTPS (secure version) uses port 443. and place holder for Module2

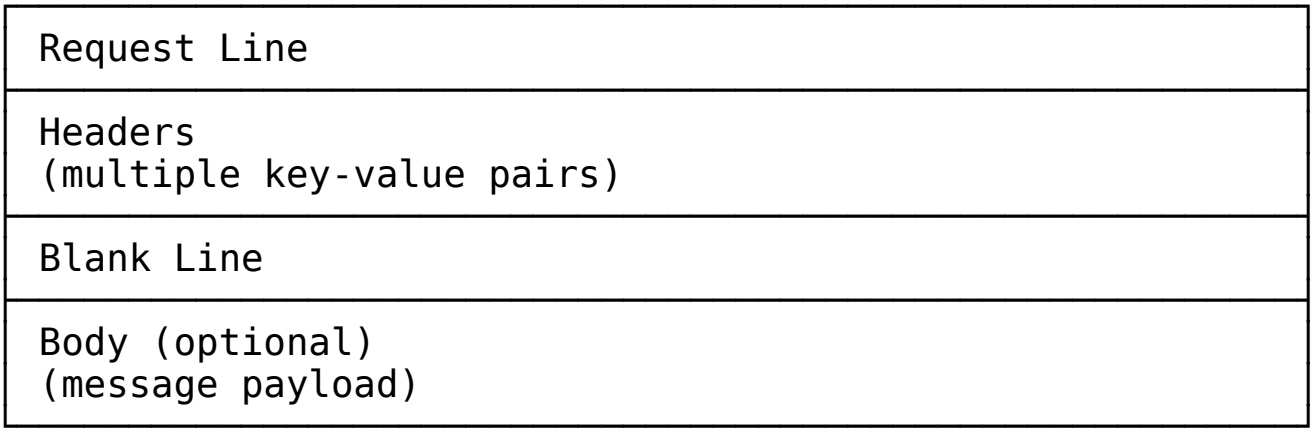
HTTP Architecture



2. Standard Request and Response Structure

HTTP Request Structure

An HTTP request consists of four main components:



Request Line Format

```
METHOD /path/to/resource HTTP/version
```

Example Request:

```
POST /api/users HTTP/1.1
Host: example.com
Content-Type: application/json
Content-Length: 45
User-Agent: Mozilla/5.0
Accept: application/json
Authorization: Bearer token123

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

Request Components Breakdown

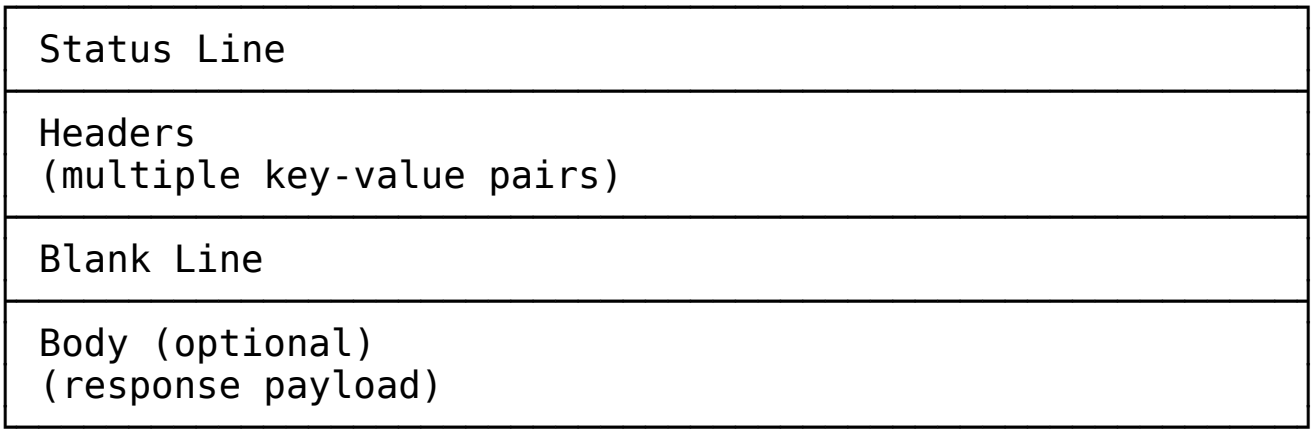
Request Line: Contains the HTTP method, resource path, and protocol version.

Headers: Metadata about the request providing additional context like content type, accepted formats, authentication credentials, and client information.

Blank Line: Separates headers from the body (mandatory even if there's no body).

Body: Contains data being sent to the server (used in POST, PUT, PATCH requests).

HTTP Response Structure



Status Line Format

```
HTTP/version StatusCode ReasonPhrase
```

Example Response:

```
HTTP/1.1 200 OK
Date: Thu, 22 Jan 2026 10:30:00 GMT
Content-Type: application/json
Content-Length: 89
Server: Apache/2.4.41
Cache-Control: no-cache

{"id":123,"username":"john","email":"john@example.com","created":"2026-01-22T10:30:00Z"}
```

3. HTTP Methods: Structure and Usage

HTTP methods (also called verbs) indicate the desired action to be performed on a resource.

GET Method

Purpose: Retrieve data from the server without modifying it.

Request Body: Not used (data sent via URL query parameters).

Structure:

```
GET /api/users?page=1&limit=10 HTTP/1.1
Host: api.example.com
Accept: application/json
```

Response Example:

```
HTTP/1.1 200 OK
Content-Type: application/json

[{"id":1,"name":"Alice"},{"id":2,"name":"Bob"}]
```

Use Cases: Fetching web pages, retrieving API data, searching, filtering results.

POST Method

Purpose: Submit data to create a new resource or trigger processing.

Request Body: Contains the data being sent.

Structure:

```
POST /api/users HTTP/1.1
Host: api.example.com
Content-Type: application/json
Content-Length: 58

{"name":"Charlie","email":"charlie@example.com","age":28}
```

Response Example:

```
HTTP/1.1 201 Created
Location: /api/users/3
Content-Type: application/json

{"id":3,"name":"Charlie","email":"charlie@example.com"}
```

Use Cases: Creating new resources, submitting forms, uploading files, triggering server-side operations.

PUT Method

Purpose: Update an existing resource or create it if it doesn’t exist (full replacement).

Structure:: Similar to POST but targets a specific resource.

Use Cases: Complete resource updates, replacing entire documents.

PATCH Method

Purpose: Partially update an existing resource.

Request Body: Contains only the fields to be updated.

Structure: Similar to PUT but with partial data.

Use Cases: Updating specific fields without sending the entire resource.

DELETE Method

Purpose: Remove a resource from the server.

Request Body: Usually empty.

Structure:

```
DELETE /api/users/3 HTTP/1.1
Host: api.example.com
```

Response Example:

```
HTTP/1.1 204 No Content
```

Use Cases: Removing resources, canceling subscriptions, clearing data.

HEAD Method

Purpose: Same as GET but retrieves only headers, not the body.

Request Body: Not used.

Structure:

```
HEAD /api/users/3 HTTP/1.1
Host: api.example.com
```

Response Example:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
Content-Length: 89
Last-Modified: Thu, 22 Jan 2026 10:30:00 GMT

(no body)
```

Use Cases: Checking if a resource exists, getting metadata, checking last modification time.

OPTIONS Method

Purpose: Describe communication options for the target resource.

Request Body: Not used.

Structure:

```
OPTIONS /api/users HTTP/1.1
Host: api.example.com
```

Response Example:

```
HTTP/1.1 200 OK
Allow: GET, POST, PUT, DELETE, OPTIONS
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Origin: *
```

Use Cases: CORS preflight requests, discovering allowed methods on a resource.

4. Important HTTP Status Codes

Status codes are three-digit numbers that indicate the result of an HTTP request. They are grouped into five categories.

Status Code Categories

```
Status Codes (3-digit)
│
├── 1xx: Informational (request received, processing)
│   └── Rarely used in modern applications
│
├── 2xx: Success (request successfully processed)
│   └── Action completed successfully
│
├── 3xx: Redirection (further action needed)
```

```
│    └── Client must take additional action
│
├── 4xx: Client Error (request contains errors)
│    └── Problem with the request itself
│
└── 5xx: Server Error (server failed to fulfill request)
     └── Server encountered an error
```

200 OK: Standard success response. Request succeeded.

201 Created: New resource created successfully (typically after POST).

301 Moved Permanently: Resource permanently moved to a new URL.

```
HTTP/1.1 301 Moved Permanently
Location: https://newsite.com/resource
```

400 Bad Request: Server cannot process the request due to client error (malformed syntax).

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{"error":"Invalid JSON format in request body"}
```

401 Unauthorized: Authentication required or failed.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="API"

{"error":"Authentication required"}
```

403 Forbidden: Server understood request but refuses to authorize it.

```
HTTP/1.1 403 Forbidden

{"error":"You don't have permission to access this resource"}
```

404 Not Found: Requested resource doesn’t exist.

```
HTTP/1.1 404 Not Found
```

```
{"error":"User not found"}
```

405 Method Not Allowed: HTTP method not supported for this resource.

```
HTTP/1.1 405 Method Not Allowed
Allow: GET, POST

{"error":"DELETE method not allowed"}
```

500 Internal Server Error: Generic server error.

```
HTTP/1.1 500 Internal Server Error

{"error":"An unexpected error occurred"}
```

Status Code Quick Reference (For understanding purposes)

Code	Name	Category	Meaning
200	OK	Success	Request succeeded
201	Created	Success	New resource created
204	No Content	Success	Success but no content to return
301	Moved Permanently	Redirection	Resource moved permanently
302	Found	Redirection	Resource temporarily moved
304	Not Modified	Redirection	Resource unchanged (cache valid)
400	Bad Request	Client Error	Malformed request
401	Unauthorized	Client Error	Authentication required
403	Forbidden	Client Error	Access denied
404	Not Found	Client Error	Resource doesn’t exist
409	Conflict	Client Error	Request conflicts with current state
422	Unprocessable Entity	Client Error	Validation failed
429	Too Many Requests	Client Error	Rate limit exceeded
500	Internal Server Error	Server Error	Generic server error
502	Bad Gateway	Server Error	Invalid upstream response
503	Service Unavailable	Server Error	Server temporarily unavailable