🔷 What is an HTTP Status Code?

An **HTTP Status Code** is a **3-digit number** sent by the **server** in response to a client's **HTTP request**. It indicates whether the request was:

- Successful
- Redirected
- Errored
- Unauthorized
- Or something else

🔢 Categories of HTTP Status Codes

Each status code falls into one of these **5 categories** based on the **first digit**:

Code Range	Category	Meaning
1xx	Informational	Request received, continuing process
2xx	Success	The request was successfully received
3xx	Redirection	Client must take further action
4xx	Client Error	Request has bad syntax or cannot be fulfilled
5xx	Server Error	The server failed to fulfill a valid request

---

✅ Common HTTP Status Codes (Used in Node.js)

🔹 1xx – Informational

Code	Description
100	Continue
101	Switching Protocols

Rarely used in Node.js apps.

🔹 2xx – Success

Code	Description	Use Case
200	OK	Standard success response
201	Created	New resource created (POST)
204	No Content	Success, but no response body

✅ **Example**:

```
res.status(200).send("Success");
res.status(201).json({ message: "User created" });
res.sendStatus(204); // No content
```

🔹 3xx – Redirection

Code	Description
301	Moved Permanently
302	Found (Temporary Redirect)
304	Not Modified

Used rarely unless implementing redirections or caching.

🔹 4xx – Client Errors

Code	Description	Use Case
400	Bad Request	Validation errors, malformed data
401	Unauthorized	Missing/invalid auth credentials
403	Forbidden	Not allowed to access the resource
404	Not Found	Resource not found
409	Conflict	Duplicate entries (e.g., email)

✅ **Example**:

```
res.status(400).json({ error: "Invalid input" });
res.status(404).send("User not found");
```

🔹 5xx – Server Errors

Code	Description	Use Case
500	Internal Server Error	Unhandled server-side error
502	Bad Gateway	Invalid response from upstream server
503	Service Unavailable	Server is down or overloaded

✅ **Example**:

```
res.status(500).json({ error: "Something went wrong" });
```

📌 How to Use Status Codes in Node.js

✅ Using Express.js (Most common)

```
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.status(200).send("Welcome!");
```

```
});

app.post("/user", (req, res) => {
  // logic to create a user
  res.status(201).json({ message: "User created" });
});

app.get("/user/:id", (req, res) => {
  const user = null; // let's say user not found
  if (!user) {
    return res.status(404).json({ error: "User not found" });
  }
});
```

💡 Why Are Status Codes Important?

- 🔍 **Helps in debugging** and error tracking.
- 📲 **Front-end uses status codes** to make decisions (like showing error messages).
- ✅ **Follows RESTful standards** in APIs.
- ⚙️ Used in tools like **Postman, curl**, and **browser DevTools**.

🧪 Real-Time Use Case

In a login route:

```
app.post("/login", (req, res) => {
  const { username, password } = req.body;

  // Validate user
  if (!username || !password) {
    return res.status(400).json({ error: "Missing credentials" });
  }

  const user = authenticate(username, password);
```

```
  if (!user) {
    return res.status(401).json({ error: "Invalid login" });
  }

  res.status(200).json({ message: "Login successful" });
});
```

📝 Summary Table

Status Code	Meaning	Method Example
200	OK	`res.status(200).send()`
201	Created	`res.status(201).json()`
204	No Content	`res.sendStatus(204)`
400	Bad Request	`res.status(400).json()`
401	Unauthorized	`res.status(401).send()`
404	Not Found	`res.status(404).send()`
500	Server Error	`res.status(500).json()`