{THE} CODING SKILLS

Understanding Server Response in Node.js

What is a Server Response?

When a **client** (like a browser) sends a request to a server (e.g., by visiting `google.com`), the **server processes the request** and sends back a **response**.
This response can be **HTML**, **JSON**, **plain text**, files, or anything else.

Example:

When you visit **google.com**, the server sends back HTML that builds the **Google homepage** on your screen.

Basic Example of Server Response

```
const http = require("http");
```

```
http
  .createServer((req, resp) => {
    resp.setHeader("Content-Type", "text/html"); // Tell browser the
response is HTML
    resp.write("<h1>Hello Anil Sidhu</h1>");
    resp.end(); // End the response
  })
  .listen("4800");
```

How to run:

Save as `app.js`

Run: `nodemon app.js` or `node app.js`

Open your browser and go to: http://localhost:4800

Passing Variables and Functions in Response

You can also include **variables**, **functions**, or **dynamic values** (like current time):

```
const http = require("http");

const age = 29;

http
  .createServer((req, resp) => {
    resp.setHeader("Content-Type", "text/html");

    resp.write(`
      <html>
        <head>
          <title>Code Step by Step</title>
        </head>
        <body>
          <h1>Hello Anil Sidhu</h1>
        <h2>Age: ${age}</h2>
        <h2>Current Date & Time: ${Date()}</h2>
      </body>
    </html>
  `);

  resp.end();
})
.listen(4800);
```

`resp.setHeader("Content-Type", "text/html")`: Tells the browser that we’re sending back HTML.

Template literals (` `): Used to inject JavaScript variables like age and `Date()` into the HTML.