Ganpat University

॥ विद्यया समाजोत्कर्षः ॥

#StudentsFirst

#CharacterMust

(Software Packages)

By

Prof. Hiteshri Modi

Node.js Web Server

- we will learn how to create a simple Node.js web server and handle HTTP requests.
- To access web pages of any web application, you need a web server.
- The web server will handle all the http requests for the web application
- e.g IIS is a web server for ASP.NET web applications and Apache is a web server for PHP or Java web applications.

Node.js Web Server

- Node.js provides capabilities to create your own web server which will handle HTTP requests asynchronously.

- You can use IIS or Apache to run Node.js web application but it is recommended to use Node.js web server.

Create Node.js Web Server

server.js

```
var http = require('http'); // 1 - Import Node.js core module

var server = http.createServer(function (req, res) {   // 2 - creating server

    //handle incomming requests here..

});

server.listen(5000); //3 - listen for any incoming requests

console.log('Node.js web server at port 5000 is running..')
```

createServer() and listen()

- The http module is a core module of Node.js, so no need to install it using NPM.
- The next step is to call createServer() method of http and specify callback function with request and response parameter.
- Finally, call listen() method of server object which was returned from createServer() method with port number, to start listening to incoming requests on port 5000.
- You can specify any unused port here.

Handle HTTP Request

- The http.createServer() method includes request and response parameters which is supplied by Node.js.
- The request object can be used to get information about the current HTTP request e.g., url, request header, and data.
- The response object can be used to send a response for a current HTTP request.

write() and writeHead()

- In Node.js, write and writeHead are two methods provided by the http.ServerResponse class, which is used to send a response back to the client during an HTTP request.

- **writeHead** sets the response status code and headers,
- while **write** is used to send the response body.

writeHead()

- The writeHead method is used to send the response header to the client.
- It takes in three parameters:
  - statusCode,
  - statusMessage,
  - and an optional headers object.

writeHead()

- statusCode (required): It represents the HTTP status code of the response, indicating the outcome of the request (e.g., 200 for success, 404 for not found, etc.).
- statusMessage (optional): It allows you to provide a custom status message that corresponds to the status code. If not provided, a default message will be sent based on the status code.
- headers (optional): It is an object containing additional headers to be sent with the response.

writeHead()

- This method must be called before any call to write or end.
- It sets the response status code and headers. For example, to send a 200 OK response with a custom header:

```
response.writeHead(200, { 'Custom-Header': 'Value' });
```

write()

- The write method is used to send the response body (i.e., the content) to the client.
- It takes a single parameter, which is the content to be sent.
- You can call write multiple times to send data in chunks.
- For example, sending "Hello, World!" as the response body:

```
response.write('Hello, World!');
```

Note that the write method does not end the response. You should follow it up with either a call to end or additional write calls to complete the response.

end()

- The end method is used to finalize the response and send it back to the client.

- After calling end, no further data can be written to the response.

response.end();

Example

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Hello World!');
  res.end();
}).listen(5000);
```

To run server

- Open browser and start localhost

  http://localhost:5000/