Node.js

Lesson 05—Working with HTTP Module

Lesson Overview

In this lesson, you will be able to understand HTTP Protocol, built, render and process HTTP Server and query string.

Learning Objectives

- ✓ Understanding HTTP Protocol
- ✓ Building an HTTP Server
- ✓ Rendering a response from an HTTP Server
- ✓ Processing Query Strings
- ✓ Processing Posted Data

Working with HTTP Module

Topic 1—What is HTTP Protocol?

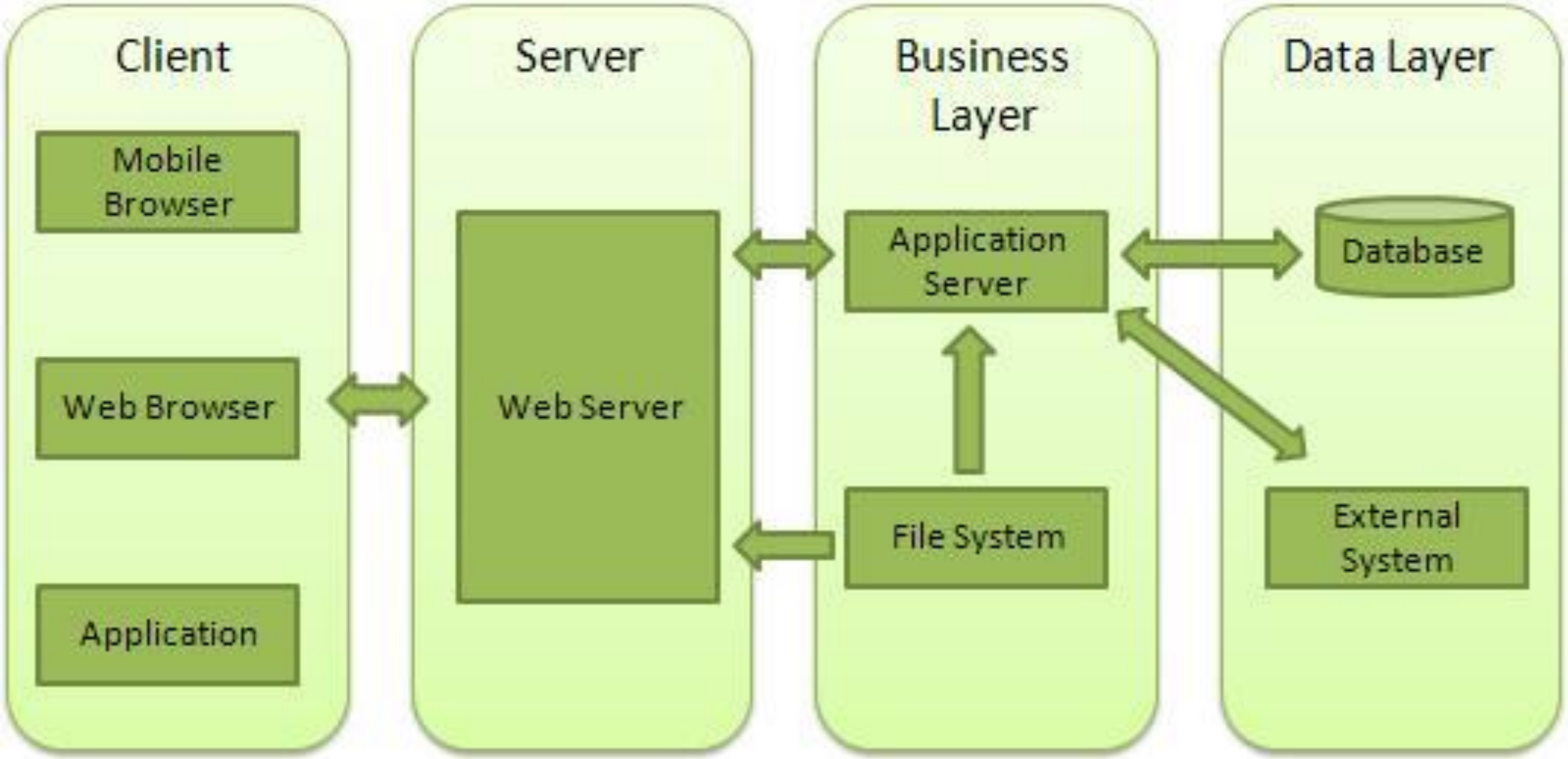
What Is HTTP Protocol?

INTRODUCTION

- Hypertext Transfer Protocol (HTTP) is a standard application protocol.
- HTTP is connectionless, media independent, and stateless in nature.
- It is mainly useful for distribution, collaboration, and hypermedia information-based systems.
- It is a core medium for data communication for WWW since 1990.
- It is a stateless protocol used to deliver data (i.e., HTML files, images, query results, and many more over the web).
- HTTP specification describes how to handle client requests and to respond to servers when data is sent to the servers.

What Is HTTP Protocol?

HTTP ARCHITECTURE



This layer consists of browsers that usually send HTTP requests to the server.

This layer maintains Web server which filters the incoming requests send by clients and responds to them.

This layer maintains business logic and connects through the database or some external system.

This standard layer maintains the databases or any external sources of data.

Working with HTTP Module

Topic 2—Building an HTTP Server

Building an HTTP Server

INTRODUCTION

- Node.js framework is primarily used to develop server-centric applications.
- This framework can also be used to create web servers which serve client requests.
- HTTP module is a standard built-in service which comes with node.js installation.
- Node.js excels at HTTP and helps developers to create standard servers and HTTP clients with minimum lines of code.

Example:

```
var http = require('http');
http.createServer(function (req, res) {
res.end('Hello world welcome to node \n');
}).listen(3000, "127.0.0.1");
console.log('Server is running at http://127.0.0.1:3000/');
```

Building an HTTP Server

HTTP HEADER

With every HTTP request and response, HTTP headers are sent which include type of content, date, details of the server that has sent the response, and HTTP status codes.

HTTP/1.1 200 OK

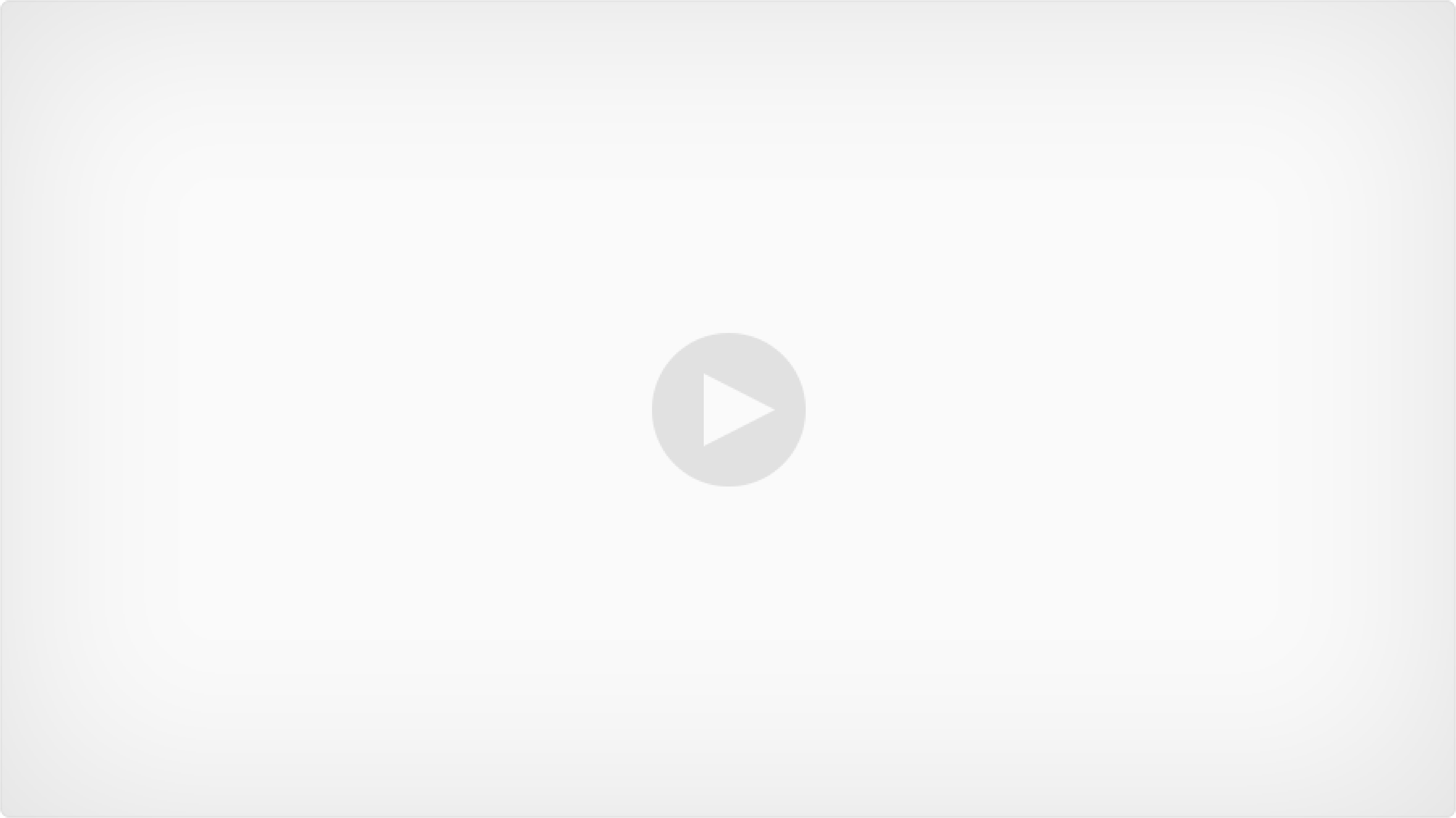
Connection: keep-alive. This displays the standard information for requesting a page from the server.

The version of HTTP is 1.1.

The response code is 200, which is a successful response.

The connection is persistent with the HTTP 1.1 protocol.

Demo for HTML Header



Working with HTTP Module

Topic 3—Rendering a response from HTTP Server

Rendering a Response

The ***res*** object of HTTP represents the HTTP response when it gets an HTTP request.

To understand this well, let’s create a simple application.

```
var http = require('http’); ←-------------Http
Module
var fs = require("fs"); ←----------File system
module

http.createServer(function(request, response) {
}).listen(3000);
```

Let’s create an HTML Page.

```
<html>
 <head>
<body>
 serving HTML files using Node.js!!!
 </body>
</html>
```

Rendering a Response

Serve this HTML page to the client on every new request to the server.

```
fs.readFile(“user.html",
function(err, data){
 response.writeHead(200,
{'Content-Type': 'text/html'});
 response.write(data);
 response.end();
});
```

response.writeHead: this HTTP method sets HTTP in response status, and the content type based on the outcome shall be treated by the HTTP client.

Response.write: this method will write the output in a browser console.

Response.end: this method will respond as client receives the outcome.

Demo for Rendering a Response

Working with HTTP Module

Topic 4—Processing Query String

Processing Query String

- The querystring module of node.js provides utilities for parsing and formatting incoming URL query strings
- It can be accessed using const querystring = require('querystring’);
- The querystring.escape() method performs URL encoding on the incoming string in way that is optimized for all the requirements of URL query strings

```
querystring.parse(str[, sep[, eq[, options]]])
```

`str <string>` represents standard incoming URL query to parse

`sep <string>` represents substring of the URL and is used to limit key and value pairs in Defaults to '&'

`eq <string>` represents substring of the URL and is used to limit keys and values in the query string in Defaults to '='

Working with HTTP Module

Topic 5—Processing Post Data

Processing Post Data

- Node.js does not provide the inbuilt logic to parse a form.

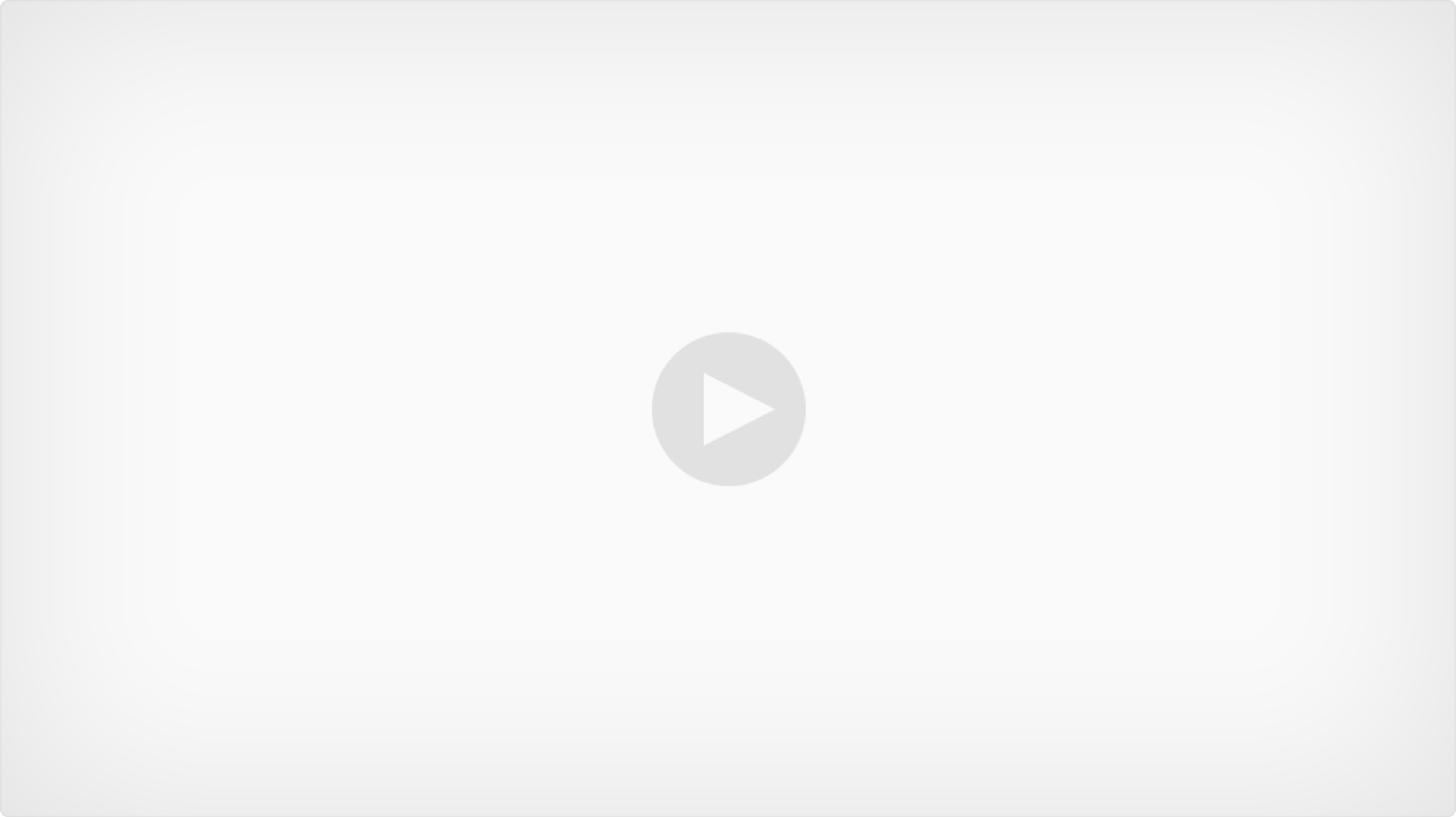
- This would require a lot of additional codes to be inserted in the application.

- Node.js's "formidable" module helps us to take care of parsing the incoming form data and returning the data of the submitted form.

- To use this module in our project, use the following command: `npm install formidable -save`

```
function processTheForm(req, res) {
 var form = new formidable.IncomingForm();
 form.parse(req, function (err, fields, files) {
res.writeHead(200, {
   'content-type': 'text/plain'
  });
  res.write('received form data:\n\n');
  res.end(util.inspect({
   fields: fields,
   files: files }));
```

Demo for Processing Query String



? Quiz

QUIZ

1

How are Node-based web servers different from traditional web servers?

a. Node-based servers process request much faster than traditional servers.

b. Node-based servers use a single-threaded model and can service much larger number of requests than traditional servers like Apache HTTP Server.

c. There is no much difference between the two.

d. None of the above.

QUIZ

1

How are Node-based web servers different from traditional web servers?

a. Node-based servers process request much faster than traditional servers.

b. Node-based servers use a single-threaded model and can service much larger number of requests than traditional servers like Apache HTTP Server.

c. There is no much difference between the two.

d. None of the above.

The correct answer is **b.**

Node-based servers use a single-threaded model and can service much larger number of requests than traditional servers like Apache HTTP Server.

QUIZ 2

Which of the following is not a valid HTTP method?

a. get

b. put

c. post

d. header

QUIZ

2

Which of the following is not a valid HTTP method?

a. get

b. put

c. post

d. header

The correct answer is **b. Header.**

Header is not valid in HTTP method.

QUIZ
3

_____ is the status code of http which says all ok.

a. 200

b. 300

c. 400

d. 5001

QUIZ 3

_____ is the status code of http which says all ok.

a. 200

b. 300

c. 400

d. 5001

The correct answer is **a. 200.**

200 is the status code of http which says all ok.

QUIZ 4

___ http method sets HTTP in response status and in content type.

a. response.writeHead

b. response.write

c. res.write

d. response.write

QUIZ

4

___ http method sets HTTP in response status and in content type.

a. response.writeHead

b. response.write

c. res.write

d. response.write

The correct answer is **a. response.writeHead.**

response.writeHead http method sets HTTP in response status and in content type

Key Takeaways

- ✓ Hypertext Transfer Protocol (HTTP) is a standard application protocol.
- ✓ Node.js excels at HTTP and helps developers to create standard servers and HTTP clients with minimum lines of code.
- ✓ The *res* object of HTTP represents the HTTP response when it gets an HTTP request.
- ✓ The querystring module of node.js provides utilities for parsing and formatting incoming URL query strings.
- ✓ Node.js's "formidable" module helps us to take care of parsing the incoming form data and returning the data of the submitted form.

Thank You