Commands

`node file.js`

`npm init`

1. To implement the console application to generate welcome & slice the arguments

```
console.log("Hi Hello World! Welcome");
console.log("Hello World");
console.log(process.argv.slice(2));

/*
process.argv is an array in Node.js that contains the command-line arguments.
slice(2) is used to remove the first two elements of the array,

- Here, arg1 and arg2 are the command-line arguments passed to the script.
- node script.js arg1 arg2

Hi Hello World! Welcome
Hello World
[ 'arg1', 'arg2' ]
*/
```

2. To implement Console Application To create event model

```
class EventEmitter {
  listeners = {};
  addListener(eventName, fn) {
    this.listeners[eventName] = this.listeners[eventName] || [];
    this.listeners[eventName].push(fn);
    return this;
  }
  on(eventName, fn) {
    return this.addListener(eventName, fn);
  }
  once(eventName, fn) {
    this.listeners[eventName] = this.listeners[eventName] || [];
    const onceWrapper = () => {
      fn();
      this.off(eventName, onceWrapper);
    };
```

```
    this.listeners[eventName].push(onceWrapper);
    return this;
  }
  off(eventName, fn) {
    return this.removeListener(eventName, fn);
  }
  removeListener(eventName, fn) {
    let lis = this.listeners[eventName];
    if (!lis) return this;
    for (let i = lis.length; i > 0; i--) {
      if (lis[i] === fn) {
        lis.splice(i, 1);
        break;
      }
    }
    return this;
  }
  emit(eventName,  ... args) {
    let fns = this.listeners[eventName];
    if (!fns) return false;
    fns.forEach((f) => {
      f( ... args);
    });
    return true;
  }
  listenerCount(eventName) {
    let fns = this.listeners[eventName] || [];
    return fns.length;
  }
  rawListeners(eventName) {
    return this.listeners[eventName];
  }
}
const myEmitter = new EventEmitter();
function c1() {
  console.log("an event occurred!");
}
function c2() {
  console.log("yet another event occurred!");
}
myEmitter.on("eventOne", c1); // Register for eventOne
myEmitter.on("eventOne", c2); // Register for eventOne
myEmitter.once("eventOnce", () => console.log("eventOnce once fired"));
myEmitter.once("init", () => console.log("init once fired"));
myEmitter.on("status", (code, msg) => console.log(`Got ${code} and ${msg}`));
myEmitter.emit("eventOne");
myEmitter.emit("eventOnce");
myEmitter.emit("eventOne");
myEmitter.emit("init");
myEmitter.emit("init");
```

```
myEmitter.emit("eventOne");
myEmitter.emit("status", 200, "ok");
console.log(myEmitter.listenerCount("event1"));
console.log(myEmitter.rawListeners("event1"));
class WithTime extends EventEmitter {
  execute(asyncFunc,  ... args) {
    this.emit("begin");
    console.time("execute");
    this.on("data", (data) => console.log("got data ", data));
    asyncFunc( ... args, (err, data) => {
      if (err) {
        return this.emit("error", err);
      }
      this.emit("data", data);
      console.timeEnd("execute");
      this.emit("end");
    });
  }
}
const withTime = new WithTime();
withTime.on("begin", () => console.log("About to execute"));
withTime.on("end", () => console.log("Done with execute"));
const readFile = (url, cb) => {
  fetch(url)
    .then((resp) => resp.json()) // Transform the data into json
    .then(function (data) {
      cb(null, data);
    });
};
withTime.execute(readFile, "https://jsonplaceholder.typicode.com/todos/1");
myEmitter.off("eventOne", c1);
myEmitter.off("eventOne", c2);
console.log(myEmitter.listenerCount("eventOne"));
console.log(withTime.rawListeners("begin"));
```

3. To implement the file system calls.

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

// Write function
fs.writeFile("test.txt", "Hello, World!", (err) => {
  if (err) throw err;
  console.log("The file has been saved.");
});

// Read function
fs.readFile("test.txt", "utf8", (err, data) => {
  if (err) throw err;
```

```
  console.log("Content:", data);
});

// Append function
fs.appendFile("test.txt", "\nAppended Text!", (err) => {
  if (err) throw err;
  console.log("The text has been appended.");
});

// Close function
fs.close(1, (err) => {
  if (err) throw err;
  console.log("File descriptor 1 was closed.");
});
```

4. To implement and demonstrate

a) Application to convert JSON to CSV.

```
npm install json2csv
```

```
const json2csv = require("json2csv").parse;

// Example JSON data
const jsonData = [
  { name: "John", age: 30, city: "New York" },
  { name: "Alice", age: 25, city: "San Francisco" },
  { name: "Bob", age: 35, city: "Los Angeles" },
];

try {
  // Convert JSON to CSV
  const csvData = json2csv(jsonData);
  console.log(csvData);
} catch (error) {
  // Handle the error
  console.error("Error converting JSON to CSV:", error.message);
}
```

b) Application to convert CSV to JSON.

```
npm install csvtojson
```

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

```
// Convert CSV to JSON
csvtojson()
  .fromFile("data.csv") // Replace 'data.csv' with the path to your CSV file
  .then((jsonData) => {
    console.log(JSON.stringify(jsonData, null, 2));
  })
  .catch((error) => {
    console.error("Error converting CSV to JSON:", error.message);
  });
```

```
name,age,city
John,30,New York
Alice,25,San Francisco
Bob,35,Los Angeles
```

5. To implement and demonstrate the opening of a file read and write a content to and from a file

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

// Read function
function readFile(sourceFile, callback) {
  fs.readFile(sourceFile, "utf8", (err, data) => {
    if (err) throw err;
    callback(data);
  });
}

// Write function
function writeFile(destinationFile, content) {
  fs.writeFile(destinationFile, content, (err) => {
    if (err) throw err;
    console.log("The file has been saved.");
  });
}

// Use the functions - to open the file
readFile("source.txt", (data) => {
  writeFile("destination.txt", data);
});
```

6. To design a webpage and implant various features using Nodejs

app.js

```
const express = require("express");
const app = express();
const port = 3000;

// Set up a basic route for the homepage
app.get("/", (req, res) => {
  res.send("<h1>Welcome to my Node.js Webpage!</h1>");
});

// Serve static files (e.g., CSS, images)
app.use(express.static("public"));

// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="/styles.css" />
    <title>Node.js Webpage</title>
  </head>
  <body>
    <h1>Welcome to my Node.js Webpage!</h1>
    <p>This is a simple webpage created using Node.js and Express.</p>
  </body>
</html>
```

public/index.css

```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
  padding: 20px;
  text-align: center;
}

h1 {
  color: #333;
}

p {
```

```
  color: #666;
}
```

update app.js

```
// ... (previous code)

// Serve static files (e.g., CSS, images)
app.use(express.static("public"));

// ... (remaining code)
```

run file

```
node app.js
```

7. To implement the application using routing.

```
const express = require("express");
const app = express();
const port = 3000;

// Middleware to log incoming requests
app.use((req, res, next) => {
  console.log(`Received ${req.method} request for ${req.url}`);
  next();
});

// Route for the home page
app.get("/", (req, res) => {
  res.send("Welcome to the home page!");
});

// Route for about page
app.get("/about", (req, res) => {
  res.send("This is the about page.");
});

// Route for contact page
app.get("/contact", (req, res) => {
  res.send("Contact us at contact@example.com");
});

// Route for handling 404 errors
app.use((req, res) => {
  res.status(404).send("404 - Page not found");
});
```

```
// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

8. write a code in nodejs to analyse the server side application using http methods

https methods implementation

1.get method
2.post method
3.put method
4.delete method

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>

  <body>
    <h1>FETCH api</h1>
    <h2>delete method</h2>
    <script>
      fetch("https://jsonplaceholder.typicode.com/todos/1", {
        method: "DELETE",
      })
        .then((response) => response.json())
        .then((data) => console.log(data));

      fetch("https://jsonplaceholder.typicode.com/todos/")
        .then((response) => response.json())
        .then((data) => console.log(data));

      fetch("https://jsonplaceholder.typicode.com/todos/5", {
        method: "PUT",
        headers: { "content-type": "application/json" },
        body: JSON.stringify({
          userId: 1,
          id: 5,
          title: "hello fetch api",
          completed: false,
        }),
```

```
      })
        .then((response) => response.json())
        .then((data) => console.log(data));

      fetch("https://jsonplaceholder.typicode.com/todos", {
        method: "POST",
        headers: { "content-type": "application/json" },
        body: JSON.stringify({
          userId: 6,
          id: 300,
          title: "hello fetch api",
          completed: false,
        }),
      })
        .then((response) => response.json())
        .then((data) => console.log(data));
    </script>
  </body>
</html>
```

10.write a code in nodejs to analyse the server side application using http methods

```
const express = require("express");
const app = express();
const bodyParser = require("body-parser");
const cors = require("cors");
const port = 3000;

app.use(bodyParser.json());

app.use(
  bodyParser.urlencoded({
    extended: true,
  })
);

app.use(cors());

app.get("/", (req, res) => {
  res.send("Welc to server");
});

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

```
app.post("/signup", (req, res) => {
  console.log(req.body.fullname);
  console.log(req.body.username);
  console.log(req.body.password);
});
```

9.create a code to query string method

```
// app.js
const readline = require("readline");
const querystring = require("querystring");

// Function to handle query string input
function handleQueryString() {
  const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
  });

  // Prompt the user to enter a query string
  rl.question("Enter a query string: ", (queryStringInput) => {
    // Parse the query string
    const parsedQuery = querystring.parse(queryStringInput);

    // Close the readline interface
    rl.close();

    // Display the parsed query
    console.log("\nParsed Query:");
    console.log(parsedQuery);
  });
}

// Run the application
handleQueryString();
```

```
name=John&age=30&city=NewYork
```

10. file system implementation

1.open
2.close
3.read
4.write

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

// Write function
fs.writeFile("test.txt", "Hello, World!", (err) => {
  if (err) throw err;
  console.log("The file has been saved.");
});

// Read function
fs.readFile("test.txt", "utf8", (err, data) => {
  if (err) throw err;
  console.log("Content:", data);
});

// Open function
fs.open("test.txt", "w", (err, fileDescriptor) => {
  if (err) {
    throw err;
  }
  console.log("File opened successfully with descriptor:", fileDescriptor);
});

// Append function
fs.appendFile("test.txt", "\nAppended Text!", (err) => {
  if (err) throw err;
  console.log("The text has been appended.");
});

// Close function
fs.close(1, (err) => {
  if (err) throw err;
  console.log("File descriptor 1 was closed.");
});
```

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

// Open function
fs.open("test.txt", "w", (err, fileDescriptor) => {
  if (err) {
    throw err;
  }

  console.log("File opened successfully with descriptor:", fileDescriptor);

  // Close the file descriptor when you're done with it
  fs.close(fileDescriptor, (closeErr) => {
    if (closeErr) {
      throw closeErr;
```

```
    }

    console.log("File descriptor closed successfully.");
  });
});
```