1. **Program for using registration form react.js class component**

```
import React, { Component } from 'react';
class RegistrationForm extends Component {
 constructor() {
  super();
  this.state = { name: '', email: '' };
 }
 handleChange = (e) => {
  this.setState({ [e.target.name]: e.target.value });
 };
 handleSubmit = (e) => {
  e.preventDefault();
  console.log(this.state);
 };
 render() {
  return (
   <form onSubmit={this.handleSubmit}>
    <input type="text" name="name" placeholder="Name"
onChange={this.handleChange} />
    <input type="email" name="email" placeholder="Email"
onChange={this.handleChange} />
    <button type="submit">Submit</button>
   </form>
  );
 }
}
export default RegistrationForm;
```

2. **Program for creating registration form using react.js function component**

```
import React, { useState } from 'react';
const RegistrationFormFunc = () => {
 const [formData, setFormData] = useState({ name: '', email: '' });
 const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
```

```
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log(formData);
  };
  return (
    <form onSubmit={handleSubmit}>
      <input type="text" name="name" placeholder="Name" onChange={handleChange}
/>
      <input type="email" name="email" placeholder="Email" onChange={handleChange}
/>
      <button type="submit">Submit</button>
    </form>
  );
};
export default RegistrationFormFunc;
```

3. Program for applying CSS style in react.js application

```
import React from 'react';
import './App.css';
const StyledComponent = () => {
  return <h1 className="styled-text">Hello, styled world!</h1>;
};
export default StyledComponent;

// App.css
/* .styled-text { color: blue; font-size: 24px; } */
```

4. Program for display any 5 MUI Components

```
import React from 'react';
import { Button, TextField, Checkbox, AppBar, Toolbar } from '@mui/material';
const MUIExample = () => {
  return (
    <div>
      <Button variant="contained">MUI Button</Button>
```

```
      <TextField label="Name" variant="outlined" />
      <Checkbox />
      <AppBar position="static">
        <Toolbar>MUI AppBar</Toolbar>
      </AppBar>
    </div>
  );
};
export default MUIExample;
```

5. **Program for printing hello on the browser using Node.js web module.**

```
const http = require('http');
http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello');
}).listen(3000, () => console.log('Server running on http://localhost:3000'));
```

6. **Program for demonstrating the concept of callback function in Node.js**

```
const callbackExample = (message, callback) => {
  console.log(message);
  callback();
};
callbackExample('Hello!', () => console.log('Callback executed'));
```

7. **Program to read the file contents using Node.js file system**

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

fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

8. **Program to write the contents to the file using Node.js file system**

```
fs.writeFile('example.txt', 'Hello, Node.js!', (err) => {
  if (err) throw err;
  console.log('File written successfully');
});
```

9. **Program to read the contents from the directory and display on console using Node.js**

```
fs.readdir('.', (err, files) => {
```

```
  if (err) throw err;
  console.log(files);
});
```

10. Program for demonstrating any 5 functions of file systems.

```
fs.rename('example.txt', 'newExample.txt', (err) => {
  if (err) throw err;
  console.log('File renamed');
});
fs.unlink('newExample.txt', (err) => {
  if (err) throw err;
  console.log('File deleted');
});
fs.mkdir('testDir', (err) => {
  if (err) throw err;
  console.log('Directory created');
});
fs.rmdir('testDir', (err) => {
  if (err) throw err;
  console.log('Directory removed');
});
fs.stat('example.txt', (err, stats) => {
  if (err) throw err;
  console.log(stats);
});
```

11. Program for demonstrating any 5 functions of console global object

```
console.log('Log message');
console.error('Error message');
console.warn('Warning message');
console.time('Timer');
console.timeEnd('Timer');
```

12. Program for demonstrating any 5 functions of process global object.

```
console.log(`Process ID: ${process.pid}`);
```

```
console.log(`Node Version: ${process.version}`);
console.log(`Platform: ${process.platform}`);
console.log(`Current Directory: ${process.cwd()}`);
process.on('exit', () => console.log('Process exiting'));
```

13. Program for demonstrating any 5 functions of OS utility module

```
const os = require('os');
console.log(`OS Type: ${os.type()}`);
console.log(`Total Memory: ${os.totalmem()}`);
console.log(`Free Memory: ${os.freemem()}`);
console.log(`Home Directory: ${os.homedir()}`);
console.log(`Uptime: ${os.uptime()}`);
```

14. Program for demonstrating any 5 functions of Path utility module

```
const path = require('path');
console.log(`Basename: ${path.basename('/test/example.txt')}`);
console.log(`Directory Name: ${path.dirname('/test/example.txt')}`);
console.log(`Extension: ${path.extname('/test/example.txt')}`);
console.log(`Join Path: ${path.join('/test', 'example.txt')}`);
console.log(`Absolute Path: ${path.resolve('example.txt')}`);
```

15. Program for demonstrating any 5 functions of Net utility module

```
const net = require('net');
const server = net.createServer((socket) => {
  console.log('Client connected');
  socket.write('Hello Client');
  socket.end();
});
server.listen(4000, () => console.log('Server running on port 4000'));
console.log(`Server Address: ${server.address().port}`);
console.log(`Server Listening: ${server.listening}`);
```

16. Program for demonstrating any 3 functions of DNS utility module

```
const dns = require('dns');
dns.lookup('example.com', (err, address) => {
  if (err) throw err;
```

```
  console.log(`IP Address: ${address}`);
});
dns.resolve('example.com', 'MX', (err, addresses) => {
  if (err) throw err;
  console.log('MX Records:', addresses);
});
dns.reverse('8.8.8.8', (err, hostnames) => {
  if (err) throw err;
  console.log('Hostnames:', hostnames);
});
```

17. Program for reading data from stream using Node.js

```
const readable = fs.createReadStream('example.txt');
readable.on('data', (chunk) => {
  console.log('Data:', chunk.toString());
});
```

18. Program for writing data to the stream using Node.js

```
const writable = fs.createWriteStream('example.txt');
writable.write('Stream data written!');
writable.end();
```

19. Program for creating a module for arithmetic operations and use it in another program using Node.js

```
const arithmetic = {
  add: (a, b) => a + b,
  subtract: (a, b) => a - b,
  multiply: (a, b) => a * b,
  divide: (a, b) => a / b,
};
module.exports = arithmetic;
// Usage in Another Program
const arithmeticOps = require('./arithmetic');
console.log(`Add: ${arithmeticOps.add(5, 3)}`);
```

20. Program for demonstrating any 5 functions of request object in Express.js

```
const express = require('express');
const app = express();
// Middleware to parse JSON data from the request body
app.use(express.json());
app.get('/', (req, res) => {
  // 1. req.url: Get the URL of the request
  console.log(`URL: ${req.url}`);

  // 2. req.method: Get the HTTP method of the request
  console.log(`Method: ${req.method}`);

  // 3. req.query: Access query parameters
  console.log(`Query Parameters: ${JSON.stringify(req.query)}`);

  // 4. req.headers: Access headers from the request
  console.log(`Headers: ${JSON.stringify(req.headers)}`);

  // 5. req.ip: Get the IP address of the client
  console.log(`IP Address: ${req.ip}`);
  res.send('Request object functions demonstrated in the console.');
});
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

21. Program for demonstrating any 5 functions of response object in Express.js

```
const express = require('express');

const app = express();

app.get('/', (req, res) => {

  // 1. res.send: Send a response to the client

  res.send('Hello, this is res.send');

  // 2. res.json: Send a JSON response

  // Uncomment the below line to test it

  // res.json({ message: 'Hello, this is res.json' });

  // 3. res.status: Set the HTTP status code

  // res.status(404).send('Page not found');

  // 4. res.set: Set a response header

  // res.set('Custom-Header', 'This is a custom header');

  // 5. res.redirect: Redirect to another URL

  // res.redirect('/new-path');

});

app.listen(3000, () => {
```

```
  console.log('Server running on http://localhost:3000');
});
```

22. Program for GET HTTP Method Using Express.js

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
  res.send('This is a GET request');
});
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

23. Program for POST HTTP Method Using Express.js

```
const express = require('express');
const app = express();
app.use(express.json()); // Middleware to parse JSON
app.post('/', (req, res) => {
  res.json({ message: 'This is a POST request', data: req.body });
});
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

24. Program for PUT HTTP Method Using Express.js

```
const express = require('express');
const app = express();
app.use(express.json()); // Middleware to parse JSON
app.put('/', (req, res) => {
  res.json({ message: 'This is a PUT request', data: req.body });
});
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

25. Program for Demonstrating the Concept of Express.js Router

```
const express = require('express');
const app = express();
const router = express.Router();
router.get('/', (req, res) => {
  res.send('Router GET request');
});
router.post('/', (req, res) => {
  res.send('Router POST request');
});
app.use('/api', router);
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000/api');
});
```

26. Program for Demonstrating the Use of app.use() in Express.js

```
const express = require('express');
const app = express();
// Middleware to log request details
app.use((req, res, next) => {
  console.log(`Request Method: ${req.method}, Request URL: ${req.url}`);
  next();
});
app.get('/', (req, res) => {
  res.send('Middleware demonstrated with app.use()');
});
app.listen(3000, () => {
  console.log('Server running on http://localhost:3000');
});
```

27. Create MongoDB Database and Insert Records Using Node.js

```
const { MongoClient } = require('mongodb');
const uri = 'mongodb://localhost:27017';
const client = new MongoClient(uri);
async function run() {
```

```
  try {
    await client.connect();
    const database = client.db('myDatabase');
    const collection = database.collection('myCollection');
    const records = [
      { name: 'Alice', age: 25 },
      { name: 'Bob', age: 30 },
      { name: 'Charlie', age: 35 }
    ];
    const result = await collection.insertMany(records);
    console.log(`${result.insertedCount} records inserted`);
  } finally {
    await client.close();
  }
}
run().catch(console.dir);
```