IP Practical Exam QB Soln

Q1. Design / Create static website include Hyperlink, Formatting, Images, Multimedia, lists.

Code: You can modify the code according to you

```
<!DOCTYPE html>
<html>
<head>
   <title>Static Website</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 20px;
      }
      h1 {
         color: #0077b6;
      }
      p {
         color: #333;
      }
      ul {
         list-style-type: disc;
      }
      ol {
         list-style-type: decimal;
      }
   </style>
</head>
<body>
   <h1>Welcome to TE-IT</h1>
   <p>This is a simple example of a static website of TE-IT.</p>

   <h2>Links</h2>
   <p>Visit these websites:</p>
   <ul>
      <li><a href="https://www.xavier.ac.in">Xavier Institute of
Engineering</a></li>
      <li><a href="https://www.google.com">Google</a></li>
   </ul>

   <h2>Images</h2>
   <p>Here are some images:</p>
   <img src="image1.jpg" alt="Image 1" width="300">
   <img src="image2.jpg" alt="Image 2" width="300">

   <h2>Multimedia</h2>
   <p>Watch this video:</p>
   <video width="400" controls>
```

```
    <source src="sample.mp4" type="video/mp4">
    Your browser does not support the video tag.
  </video>

  <h2>Lists</h2>
  <p>Here are some lists:</p>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
  <ol>
    <li>First item</li>
    <li>Second item</li>
  </ol>
</body>
</html>
```

Q2. Design / Create webpages including CSS3 Syntax, Background, Fonts, Tables, lists, CSS3 selectors.

Code: CSS is applied for above Html Code

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

body {
  background-color: #f5f5f5;
}

h1 {
  color: #0077b6;
  font-size: 24px;
}

p {
  color: #333;
  font-size: 16px;
}

table {
  border-collapse: collapse;
  width: 80%;
  margin: 20px auto;
}

table, th, td {
  border: 1px solid #333;
}
```

```
th, td {
    padding: 10px;
    text-align: left;
}

ul li:nth-child(odd) {
    background-color: #f0f0f0;
}

ul li:nth-child(even) {
    background-color: #e0e0e0;
}

h2 {
    font-size: 20px;
    color: #009688;
}
```

Q3. Write a program to display CSS border properties & Color properties.

Code:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .border-example {
            border: 2px solid #0077b6;
            padding: 10px;
            margin: 20px;
            border-radius: 10px;
        }

        .color-example {
            background-color: #f0f0f0; /* Background color */
            color: #333; /* Text color */
        }
    </style>
</head>
<body>
    <h1>CSS Border and Color Properties</h1>
```

```
    <div class="border-example">
        <h2>Border Example</h2>
        <p>This div has a border with various properties, including width, style, color, padding, margin, and rounded corners.</p>
    </div>

    <div class="color-example">
        <h2>Color Example</h2>
        <p>This div has background color and text color set using CSS properties.</p>
    </div>
</body>
</html>
```

Output:

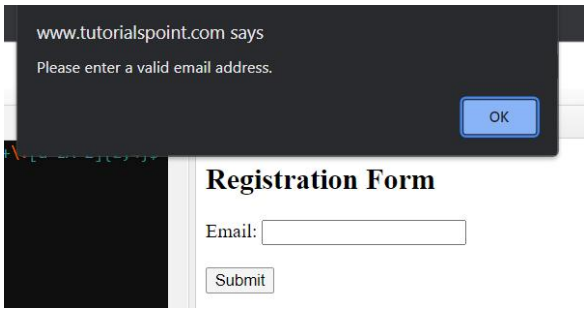
Q4. Write a program in JS to validate Registration Form including email validation.

Code:

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function validateForm() {
            var email = document.forms["registrationForm"]["email"].value;
            var emailPattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
            if (!email.match(emailPattern)) {
                alert("Please enter a valid email address.");
                return false; // Prevent form submission
```

```
        }
        return true;
      }
    </script>
</head>
<body>
    <h2>Registration Form</h2>
    <form name="registrationForm" onsubmit="return validateForm()" method="post">
        <label for="email">Email:</label>
        <input type="text" name="email" id="email">
        <br><br>
        <!-- Add more form fields here as needed -->
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Output:



Q5. Write a JS program to demonstrate Iterators & Generators.

Code:

```
// Define an iterable object using a generator function
function* iterableObject() {
    yield 'Hello';
    yield 'World';
    yield '!';
}

// Create an iterator from the iterable object
const iterator = iterableObject();

// Use a for...of loop to iterate over the values
for (const value of iterator) {
    console.log(value);
}
```

Q6. Write JS program to demonstrate Promise.

Code:

```
// Create a Promise that immediately resolves with a value
const myPromise = Promise.resolve("Promise resolved!");
// Using the Promise
myPromise
   .then((result) => {
      console.log(result);
   });
```

Q7. Write a program to demonstrate Inheritance.

Code:

```
// Define a base class
class Animal {
   constructor(name) {
      this.name = name;
   }
   sayHello() {
      console.log(`Hello, I'm ${this.name}.`);
   }
}
// Define a subclass that inherits from Animal
class Dog extends Animal {
   constructor(name, breed) {
      super(name);
      this.breed = breed;
   }
   bark() {
      console.log(`${this.name} (a ${this.breed} dog) barks: Woof! Woof!`);
   }
}
// Create instances of the base class and subclass
const animal = new Animal("Generic Animal");
const dog = new Dog("Buddy", "Golden Retriever");
```

```
// Demonstrate inheritance
animal.sayHello();
dog.sayHello();
dog.bark();
```

Q8. Write a program to demonstrate Class and Function Component.

Code:

```
import React from 'react';
// Class Component
class ClassComponent extends React.Component {
  render() {
    return (
      <div>
        <h2>This is a Class Component</h2>
      </div>
    );
  }
}
// Function Component
function FunctionComponent() {
  return (
    <div>
      <h2>This is a Function Component</h2>
    </div>
  );
}
// Render both components
function App() {
  return (
    <div>
      <ClassComponent />
      <FunctionComponent />
    </div>
  );
}
export default App;
```

Q9. Write a program to demonstrate State and Props.

Code:

```
import React, { useState } from 'react';
function SimpleReactApp() {
  const [displayText, setDisplayText] = useState('Click the button to change text');
  const handleButtonClick = () => {
    setDisplayText('Text changed!');
  };
  return (
    <div>
      <p>{displayText}</p>
      <button onClick={handleButtonClick}>Change Text</button>
    </div>
  );
}
export default SimpleReactApp;
```

Q10. Write a program to demonstrate Form Handling in Reactjs.

Code:

```
import React, { useState } from 'react';

function FormHandling() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
  });

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData({
      ...formData,
      [name]: value,
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log('Form submitted with data:', formData);
  };

  return (
```

```
    <div>
      <h2>Form Handling in React</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Name:</label>
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleInputChange}
          />
        </div>
        <div>
          <label>Email:</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleInputChange}
          />
        </div>
        <button type="submit">Submit</button>
      </form>
    </div>
  );
}
export default FormHandling;
```

Q11. Write a program to display fibonacii series in REPL 1 1 2 3 5 8 13…

Code:

```
> .editor
// Entering editor mode (Ctrl+D to finish, Ctrl+C to cancel)
function fib(n) {
if (n <= 1){
return n;
} else {
return fib(n-1) + fib(n-2);
}
}

console.log('Fibonacci Series: \n');

for(var i=0; i<12; i++){
console.log(fib(i));
}
Fibonacci Series:

0
1
1
2
3
5
8
13
21
34
55
89
undefined
```

Q12. REPL Environment demonstrate variables and multiline expression.

Code:

```
// Declare and initialize a variable
const x = 10;

// Display the value of the variable
x

// Perform calculations and display the results
const y = x * 2;
y

// Define a multiline expression
const multiLineExpression = `
This is a multiline expression.
You can use multiple lines to write code and explanations.
`;

// Display the multiline expression
multiLineExpression
```

Q13. Demonstrate Nodejs Read Stream &Write Stream.

Code:

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

// Create a Read Stream for an input file
const readStream = fs.createReadStream('input.txt', 'utf8');

// Create a Write Stream for an output file
const writeStream = fs.createWriteStream('output.txt');

// Pipe the Read Stream to the Write Stream to copy the data
readStream.pipe(writeStream);

// Display a message when the process is completed
writeStream.on('finish', () => {
  console.log('Data has been copied to output.txt');
});
```

Q14. Demonstrate Nodejs Web Module.

Code:

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

// Create an HTTP server
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello, Node.js Web Server!\n');
});

// Listen on port 3000
const port = 3000;
server.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}/`);
});
```

Q15. Demonstrate Express Router.

Code:

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

// Create an instance of the Express Router
const router = express.Router();

// Define a route on the router
router.get('/', (req, res) => {
  res.send('Hello from the router!');
});

// Mount the router at a specific path
app.use('/myrouter', router);

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

Q16. Write a program to display class timetable and apply styles on it. Write a program to display even or odd number.

Code:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    table {
      width: 50%;
      border-collapse: collapse;
      margin: 0 auto;
    }

    th, td {
      border: 1px solid #000;
      padding: 8px;
      text-align: center;
    }

    th {
      background-color: #f2f2f2;
    }

    tr:nth-child(even) {
      background-color: #f2f2f2;
    }

    tr:nth-child(odd) {
      background-color: #ffffff;
    }
  </style>
</head>
<body>
  <h2>Class Timetable</h2>
  <table>
    <tr>
      <th>Time</th>
      <th>Subject</th>
    </tr>
    <tr>
      <td>08:00 AM</td>
      <td>Math</td>
    </tr>
    <tr>
      <td>09:30 AM</td>
      <td>Science</td>
    </tr>
    <tr>
      <td>11:00 AM</td>
```

```
    <td>History</td>
   </tr>
  </table>
</body>
</html>
```

Script.js

```
function isEvenOrOdd(number) {
  if (number % 2 === 0) {
    return 'Even';
  } else {
    return 'Odd';
  }
}

const numberToCheck = 7;
console.log(numberToCheck + ' is ' + isEvenOrOdd(numberToCheck));
```

Q17. Write a program to display pattern in REPL

```
*
**
***
```

Code:

```
function printPattern(rows) {
  for (let i = 1; i <= rows; i++) {
    let pattern = '';
    for (let j = 1; j <= i; j++) {
      pattern += '*';
    }
    console.log(pattern);
  }
}

const numberOfRows = 3; // Replace with the number of rows you want
printPattern(numberOfRows);
```

Q18. Write a program to demonstrate Nodejs File system.

Code:

```
const fs = require('fs');
const filePath = 'example.txt';
fs.readFile(filePath, 'utf8', (err, data) => {
  if (err) {
    console.error('Error reading the file:', err);
  } else {
    console.log('File contents:');
    console.log(data);
  }
});
```

Q19. Write a program to display react Router.

Code:

```
import React from 'react';
import {
  BrowserRouter as Router,
  Route,
  Link,
  Switch
} from 'react-router-dom';

const Home = () => <h2>Home</h2>;
const About = () => <h2>About</h2>;
const Contact = () => <h2>Contact</h2>;

const App = () => (
  <Router>
    <div>
      <nav>
        <ul>
          <li>
            <Link to="/">Home</Link>
```

```
          </li>
          <li>
            <Link to="/about">About</Link>
          </li>
          <li>
            <Link to="/contact">Contact</Link>
          </li>
        </ul>
      </nav>

      <Switch>
        <Route path="/about" component={About} />
        <Route path="/contact" component={Contact} />
        <Route path="/" component={Home} />
      </Switch>
    </div>
  </Router>
);
export default App;
```

Q20. Demonstrate Nodejs Piping Streams.

Code:

```
const fs = require('fs');
const readStream = fs.createReadStream('input.txt');
const writeStream = fs.createWriteStream('output.txt');
readStream.pipe(writeStream);
console.log('Data is being piped from input.txt to output.txt.');
```

Q21. Design / Create static website include Tables, List, Forms, alert messages.

Code: Refer Q1, Q2 and Q4

Q22. Write a program to apply CSS using react (Inline , External , Internal)

Code:

```
import React from 'react';
function App() {
  const inlineStyles = {
    color: 'blue',
    fontSize: '20px',
  };
  return (
    <div>
      <p style={inlineStyles}>This text uses inline styles.</p>
    </div>
  );
}
export default App;
```

Q23. Write a program to display factorial of number in REPL.

Code:

```
 function factorial(n) {
  if (n === 0 || n === 1) {
    return 1;
  } else {
    return n * factorial(n - 1);
  }
}

console.log(factorial(5));
```