1. Resume Using HTML Tags

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Resume</title>
</head>
<body>
    <header>
        <h1>John Doe</h1>
        <p>Email: johndoe@example.com | Phone: 123-456-7890</p>
    </header>
    <hr>
    <section>
        <h2>Objective</h2>
        <p>To obtain a challenging position in a reputable
organization to expand my learning, knowledge, and skills.</p>
    </section>
    <section>
        <h2>Education</h2>
        <ul>
            <li><strong>Bachelor of Science in Computer
Science</strong> - XYZ University (2017-2021)</li>
        </ul>
    </section>
    <section>
        <h2>Experience</h2>
        <ul>
            <li><strong>Software Developer Intern</strong> - ABC Corp
(June 2020 - August 2020)</li>
        </ul>
    </section>
    <section>
        <h2>Skills</h2>
        <ul>
```

```
            <li>HTML, CSS, JavaScript</li>
            <li>React, Node.js</li>
        </ul>
    </section>
</body>
</html>
```

---

2. Registration Form with HTML5 Form Tags

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Registration Form</title>
</head>
<body>
    <h2>Registration Form</h2>
    <form action="/submit" method="post">
        <label for="name">Full Name:</label>
        <input type="text" id="name" name="name" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob" required><br><br>

        <label for="gender">Gender:</label>
        <input type="radio" id="male" name="gender" value="male">
        <label for="male">Male</label>
        <input type="radio" id="female" name="gender" value="female">
        <label for="female">Female</label><br><br>

        <label for="file">Upload Profile Picture:</label>
        <input type="file" id="file" name="file"><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
</html>
```

---

3. Web Page Using CSS Font and Text Properties

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>CSS Font and Text Properties</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        h1 {
            font-size: 36px;
            color: #333;
            text-align: center;
        }
        p {
            font-size: 16px;
            line-height: 1.6;
            text-align: justify;
        }
        .highlight {
            color: red;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is an example paragraph demonstrating <span
class="highlight">font and text properties</span> using CSS.</p>
```

```
</body>
</html>
```

---

4. Web Page with Background Properties

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Background Properties</title>
    <style>
        body {
            background-image: url('background.jpg');
            background-size: cover;
            background-repeat: no-repeat;
            background-position: center;
            color: white;
        }
        h1 {
            text-align: center;
            padding-top: 50px;
        }
    </style>
</head>
<body>
    <h1>Welcome to My Website</h1>
</body>
</html>
```

---

5. JavaScript Program to Calculate Volume of a Cylinder

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Volume of a Cylinder</title>
    <script>
        const PI = 3.14159;
        function calculateVolume() {
            const radius =
parseFloat(document.getElementById("radius").value);
            const height =
parseFloat(document.getElementById("height").value);
            const volume = PI * radius * radius * height;
            document.getElementById("result").innerText = "Volume: " +
volume.toFixed(2);
        }
    </script>
</head>
<body>
    <h2>Calculate Volume of a Cylinder</h2>
    <label for="radius">Radius:</label>
    <input type="number" id="radius"><br><br>
    <label for="height">Height:</label>
    <input type="number" id="height"><br><br>
    <button onclick="calculateVolume()">Calculate</button>
    <p id="result"></p>
</body>
</html>
```

---

6. JavaScript Program to Set Background Color and Highlight Text

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Highlight Paragraph</title>
    <script>
        function highlightText() {
            document.getElementById("para").style.backgroundColor =
"yellow";
            document.getElementById("para").style.fontWeight = "bold";
        }
    </script>
</head>
<body>
    <p id="para">This is a paragraph that will be highlighted with a
yellow background.</p>
    <button onclick="highlightText()">Highlight</button>
</body>
</html>
```

---

7. JavaScript Program to Calculate Simple Interest

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Simple Interest</title>
    <script>
        function calculateSimpleInterest(p, r, t) {
            return (p * r * t) / 100;
        }

        const calculateInterest = () => {
            const principal =
parseFloat(document.getElementById("principal").value);
            const rate =
parseFloat(document.getElementById("rate").value);
            const time =
parseFloat(document.getElementById("time").value);

            const interest = calculateSimpleInterest(principal, rate,
time);
```

```
            document.getElementById("result").innerText = "Simple
Interest: " + interest.toFixed(2);
        }
    </script>
</head>
<body>
    <h2>Calculate Simple Interest</h2>
    <label for="principal">Principal:</label>
    <input type="number" id="principal"><br><br>
    <label for="rate">Rate of Interest:</label>
    <input type="number" id="rate"><br><br>
    <label for="time">Time (Years):</label>
    <input type="number" id="time"><br><br>
    <button onclick="calculateInterest()">Calculate</button>
    <p id="result"></p>
</body>
</html>
```

---

8. JavaScript Program with Class and Inheritance

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Class and Inheritance</title>
    <script>
        class Person {
            constructor(name, age) {
                this.name = name;
                this.age = age;
            }

            displayInfo() {
                return `Name: ${this.name}, Age: ${this.age}`;
            }
        }
        class Student extends Person {
            constructor(name, age, course, marks) {
                super(name, age);
                this.course = course;
                this.marks = marks;
            }

            displayInfo() {
                return `${super.displayInfo()}, Course:
${this.course}, Marks: ${this.marks}`;
            }
        }

        function showStudents() {
            const student1 = new Student("Alice", 20, "Math", 95);
            const student2 = new Student("Bob", 22, "Physics", 88);

            document.getElementById("student1").innerText =
student1.displayInfo();
            document.getElementById("student2").innerText =
student2.displayInfo();
        }
    </script>
</head>
<body onload="showStudents()">
    <h2>Students Information</h2>
    <p id="student1"></p>
    <p id="student2"></p>
</body>
</html>
```

---

9. JavaScript Program Demonstrating Promises

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Promise Example</title>
    <script>
        const fetchData = new Promise((resolve, reject) => {
            const success = true; // Simulate success or failure
            if (success) {
                setTimeout(() => resolve("Data fetched
successfully!"), 1000);
            } else {
                setTimeout(() => reject("Error fetching data."),
1000);
            }
        });

        function handleData() {
            fetchData
                .then(response =>
document.getElementById("result").innerText = response)
                .catch(error =>
document.getElementById("result").innerText = error);
        }
    </script>
</head>
<body>
    <h2>Promise Example</h2>
    <button onclick="handleData()">Fetch Data</button>
    <p id="result"></p>
</body>
</html>
```

---

10. JavaScript Program Demonstrating Callbacks

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Callback Example</title>
    <script>
        function fetchData(callback) {
            setTimeout(() => {
                const data = "Data fetched successfully!";
                callback(data);
            }, 1000);
        }

        function displayData(data) {
            document.getElementById("result").innerText = data;
        }

        function handleFetch() {
            fetchData(displayData);
        }
    </script>
</head>
<body>
    <h2>Callback Example</h2>
    <button onclick="handleFetch()">Fetch Data</button>
    <p id="result"></p>
</body>
</html>
```

---

11. "Hello World" in React

```
import React from 'react';
import ReactDOM from 'react-dom';

function App() {
    return (
        <h1>Hello World</h1>
    );
}
```

```
ReactDOM.render(<App />, document.getElementById('root'));
```

---

12. State and Props in React

```
import React, { useState } from 'react';
import ReactDOM from 'react-dom';

function ChildComponent(props) {
    return <h2>Hello, {props.name}!</h2>;
}

function App() {
    const [name, setName] = useState('John');

    return (
        <div>
            <ChildComponent name={name} />
            <button onClick={() => setName('Jane')}>Change
Name</button>
        </div>
    );
}

ReactDOM.render(<App />, document.getElementById('root'));
```

---

13. React Registration Form with Validations

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

function RegistrationForm() {
    const [formData, setFormData] = useState({ name: '', email: '',
password: '' });
    const [errors, setErrors] = useState({});

    const validate = () => {
        let newErrors = {};
        if (!formData.name) newErrors.name = "Name is required";
        if (!formData.email) newErrors.email = "Email is required";
        if (!formData.password || formData.password.length < 6)
newErrors.password = "Password must be at least 6 characters long";
        setErrors(newErrors);
        return Object.keys(newErrors).length === 0;
    };

    const handleSubmit = (e) => {
        e.preventDefault();
        if (validate()) {
            alert("Form submitted successfully!");
        }
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>Name:</label>
                <input
                    type="text"
                    value={formData.name}
                    onChange={(e) => setFormData({ ...formData, name:
e.target.value })}
                />
                {errors.name && <span>{errors.name}</span>}
            </div>
            <div>
                <label>Email:</label>
                <input
                    type="email"
                    value={formData.email}
                    onChange={(e) => setFormData({ ...formData, email:
e.target.value })}
                />
                {errors.email && <span>{errors.email}</span>}
            </div>
            <div>
                <label>Password:</label>
                <input
                    type="password"
                    value={formData.password}
                    onChange={(e) => setFormData({ ...formData,
password: e.target.value })}
                />
                {errors.password && <span>{errors.password}</span>}
            </div>
            <button type="submit">Register</button>
        </form>
    );
}

export default RegistrationForm;
```

---

14. React Router and Single Page Application (SPA)

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

function Home() {
    return <h2>Home</h2>;
}

function About() {
    return <h2>About</h2>;
}

function App() {
    return (
        <Router>
            <nav>
                <Link to="/">Home</Link>
                <Link to="/about">About</Link>
            </nav>

            <Route exact path="/" component={Home} />
            <Route path="/about" component={About} />
        </Router>
    );
}

export default App;
```

---

15. React Program Demonstrating Hooks (useEffect) and Refs

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

function App() {
    const [count, setCount] = useState(0);
    const inputRef = useRef(null);

    useEffect(() => {
        document.title = `You clicked ${count} times`;
    }, [count]);

    return (
        <div>
            <h1>Count: {count}</h1>
            <button onClick={() => setCount(count +
1)}>Increment</button>
            <input ref={inputRef} placeholder="Focus on me" />
            <button onClick={() => inputRef.current.focus()}>Focus
Input</button>
        </div>
    );
}

export default App;
```

---

16. "Hello World" Using HTTP Module in Node.js

```
// hello-world.js
const http = require('http');
const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World\n');
});

server.listen(3000, () => {
    console.log('Server running at http://localhost:3000/');
});
```

Demonstrating REPL in Node.js:

1. Open a terminal and type `node` to start the REPL.
2. Type JavaScript commands like `console.log('Hello World')`.
3. The output will be displayed in the terminal.

---

17. File Operations in Node.js (Create a File and Write to It)

```
// file-operations.js
const fs = require('fs');

// i) Create a file and write a paragraph into it
fs.writeFile('example.txt', 'This is a sample paragraph written to the
file.', (err) => {
    if (err) throw err;
    console.log('File created and paragraph written successfully.');
});
```

---

18. File Operations in Node.js (Append and Rename)

```
// append-rename.js
const fs = require('fs');

// i) Append some text to an existing file
fs.appendFile('example.txt', '\nThis is the appended text.', (err) =>
{
    if (err) throw err;
    console.log('Text appended successfully.');

    // ii) Rename the file
    fs.rename('example.txt', 'renamed-example.txt', (err) => {
        if (err) throw err;
        console.log('File renamed successfully.');
    });
});
```

---

19. Program to Demonstrate Buffers in Node.js

```
// buffer-example.js
const buffer = Buffer.from('Hello, Buffer in Node.js!');

console.log('Buffer Content:', buffer);
console.log('Buffer to String:', buffer.toString());
console.log('Buffer Length:', buffer.length);

// Create an empty buffer of 10 bytes
const emptyBuffer = Buffer.alloc(10);
console.log('Empty Buffer:', emptyBuffer);

// Write to buffer
buffer.write('Node.js');
console.log('Modified Buffer:', buffer.toString());
```

---

20. Program Demonstrating Asynchronous Programming in Node.js

```
// async-programming.js
const fs = require('fs');

// Asynchronous file reading
fs.readFile('renamed-example.txt', 'utf8', (err, data) => {
    if (err) {
        console.log('Error reading file:', err);
        return;
    }
    console.log('File content read asynchronously:', data);
});

// Simulating asynchronous operation using setTimeout
console.log('Start');

setTimeout(() => {
    console.log('Asynchronous operation complete after 2 seconds');
}, 2000);

console.log('End');
```

---

21. "Hello World" Program Using Express.js

```
// hello-express.js
const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.send('Hello World from Express!');
});

app.listen(3000, () => {
    console.log('Server is running on http://localhost:3000');
});
```

Instructions to Run Express Program:

1. First, install Express by running:
   `npm install express`
2. Run the Express app using:
   `node hello-express.js`
3. Open your browser and navigate to `http://localhost:3000` to see the message "Hello World from Express!".