1) Dynamic dropdown

```
<!DOCTYPE html>
<html>
<head>
  <title>Dynamic Dropdown Population</title>
</head>
<body>
  <h1>Dynamic Dropdown Population</h1>

  <select id="categorySelect">
    <option value="fruits">Fruits</option>
    <option value="vegetables">Vegetables</option>
  </select>

  <select id="itemSelect">
    <option value="">Select an item</option>
  </select>

  <script>
    // Get references to the select elements
    const categorySelect = document.getElementById('categorySelect');
    const itemSelect = document.getElementById('itemSelect');

    // Define a data structure for items
    const items = {
      fruits: ['Apple', 'Banana', 'Orange'],
      vegetables: ['Carrot', 'Broccoli', 'Tomato'],
    };

    // Function to populate the item dropdown based on the selected category
    function populateItems() {
      const selectedCategory = categorySelect.value;
      const itemOptions = items[selectedCategory];

      // Clear the current options
      itemSelect.innerHTML = '';

      // Add a default option
      const defaultOption = document.createElement('option');
      defaultOption.value = '';
      defaultOption.text = 'Select an item';
      itemSelect.appendChild(defaultOption);

      // Populate the item options
      if (itemOptions) {
        itemOptions.forEach(item => {
          const option = document.createElement('option');
          option.value = item;
          option.text = item;
          itemSelect.appendChild(option);
        });
      }
    }
```

```
        // Event listener to call the populateItems function when the category changes
        categorySelect.addEventListener('change', populateItems);

        // Initial population of items
        populateItems();
    </script>
</body>
</html>
```

2) Movie ticketing website and calculate total amount.

3) React Hooks.

In App.js

```
import React, { useState, useEffect } from "react";

function App() {
  // Declare a state variable named 'count' with an initial value of 0
  const [count, setCount] = useState(0);

  // Declare a side effect using the useEffect hook
  useEffect(() => {
    document.title = `Count: ${count}`;
  }, [count]);

  return (
    <div>
      <h1>React Hooks Example</h1>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
    </div>
  );
}

export default App;
```

4) Make a form and validate using JS not HTML attributes.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Form Validation</title>
  </head>
  <body>
    <h1>Form Validation with JavaScript</h1>
    <form id="myForm" onsubmit="return validateForm()">
      <label for="name">Name:</label>
      <input type="text" id="name" placeholder="Enter your name" /><br /><br />

      <label for="email">Email:</label>
      <input
        type="text"
        id="email"
        placeholder="Enter your email"
```

```
    /><br /><br />

    <label for="password">Password:</label>
    <input
      type="password"
      id="password"
      placeholder="Enter your password"
    /><br /><br />

    <button type="submit">Submit</button>
  </form>

  <p id="errorText" style="color: red"></p>

  <script>
    function validateForm() {
      // Get form elements
      var name = document.getElementById("name").value;
      var email = document.getElementById("email").value;
      var password = document.getElementById("password").value;
      var errorText = document.getElementById("errorText");

      // Simple validation: check if fields are empty
      if (name === "" || email === "" || password === "") {
        errorText.textContent = "All fields are required";
        return false; // Prevent form submission
      }

      // Validate email using a simple regular expression
      var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailPattern.test(email)) {
        errorText.textContent = "Invalid email format";
        return false; // Prevent form submission
      }

      // Password length validation
      if (password.length < 6) {
        errorText.textContent = "Password must be at least 6 characters long";
        return false; // Prevent form submission
      }

      // If all validation checks pass, the form is submitted
      errorText.textContent = ""; // Clear any previous error message
      return true;
    }
  </script>
 </body>
</html>
```

5) Create a NodeJS module and use it to perform arithmetic operations.

In mathOperation.js

```
function add(a, b) {
  return a + b;
}
```

```
function subtract(a, b) {
  return a - b;
}

function multiply(a, b) {
  return a * b;
}

function divide(a, b) {
  if (b == 0) {
    throw new Error("Division by zero is not possible");
  }
  return a / b;
}

module.exports = {
  add,
  subtract,
  multiply,
  divide,
};
```

In App.js

```
const mathOperations = require("./mathOperation.js");

const a = 10;
const b = 5;

console.log(`Addition: ${a} + ${b} = ${mathOperations.add(a, b)}`);
console.log(`Subtraction: ${a} - ${b} = ${mathOperations.subtract(a, b)}`);
console.log(`Multiplication: ${a} * ${b} = ${mathOperations.multiply(a, b)}`);

try {
  console.log(`Division: ${a} / ${b} = ${mathOperations.divide(a, b)}`);
} catch (e) {
  console.log(e.message);
}
```

6) Inheritance in react

In App.js

```
import React from "react";

// Base class
class Animal extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: "Unknown",
    };
  }

  render() {
    return (
```

```
      <div>
        <h2>Animal</h2>
        <p>Name: {this.state.name}</p>
      </div>
    );
  }
}

// Child class that inherits from Animal
class Dog extends Animal {
  constructor(props) {
    super(props);
    this.state.name = "Dog";
  }

  render() {
    return (
      <div>
        <h2>Dog</h2>
        <p>Name: {this.state.name}</p>
      </div>
    );
  }
}

function App() {
  return (
    <div>
      <h1>Inheritance in React</h1>
      <Animal />
      <Dog />
    </div>
  );
}

export default App;
```

7) Routing using React,Express,Node

React

App.js

```
import "./App.css";
import Home from "./components/Home.jsx";
import About from "./components/About.jsx";
import Contact from "./components/Contact.jsx";
import { Routes, Route, BrowserRouter } from "react-router-dom";

function App() {
  return (
    <>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
```

```
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </BrowserRouter>
  </>
 );
}

export default App;
```

/components/Home.js

```
import React from 'react'

const Home = () => {
 return (
  <div>
   Home component
  </div>
 )
}

export default Home
```

/components/About.js

```
import React from 'react'

const About = () => {
 return (
  <div>
   About component
  </div>
 )
}

export default About
```

/components/Contact.js

```
import React from 'react'

const Contact = () => {
 return (
  <div>
   Contact component
  </div>
 )
}

export default Contact
```

Express

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

```
const port = 3000;

app.listen(port, () => console.log(`Server challa hai at port ${port}!`));

app.get('/', (req, res) => {
   res.send('Home');
});

app.get('/about', (req, res) => {
   res.send('About');
})

app.get('/contact', (req, res) => {
   res.send('Contact');
})
```

Node

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

// Create a server object
http.createServer(function (req, res) {

        // http header
        res.writeHead(200, {'Content-Type': 'text/html'});

        const url = req.url;

        if(url ==='/about') {
                res.write(' Welcome to about us page');
                res.end();
        }
        else if(url ==='/contact') {
                res.write(' Welcome to contact us page');
                res.end();
        }
        else {
                res.write('Hello World!');
                res.end();
        }
}).listen(3000, function() {

        // The server object listens on port 3000
        console.log("server start at port 3000");
});
```

8) Bootstrap tooltips and breadcrumbs

Tooltips: added to breadcrumbHome.html similarly sab me daal do

Breadcrumbs:
breadcrumbHome.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MP
K8M2HN" crossorigin="anonymous">
<body>
       <nav aria-label="breadcrumb">
       <ol class="breadcrumb">
         <li class="breadcrumb-item" >Home</li>
         <li class="breadcrumb-item" ><a href="breadcrumbAbout.html" data-toggle="tooltip"
title="Click to go to about page">About</a></li>
         <li class="breadcrumb-item active" aria-current="page"><a
href="breadcrumbContact.html" data-toggle="tooltip" title="Click to go to Contact
page">Contact</a></li>
       </ol>
   </nav>
   <h1>This is home page</h1>
</body>
</html>
```

breadcrumbContact.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MP
K8M2HN" crossorigin="anonymous">
<body>
   <nav aria-label="breadcrumb">
      <ol class="breadcrumb">
        <li class="breadcrumb-item"> <a href="breadcrumbHome.html"> Home </a> </li>
        <li class="breadcrumb-item"><a href="breadcrumbAbout.html">About</a></li>
```

```
        <li class="breadcrumb-item active" aria-current="page">Contact</li>
      </ol>
    </nav>
    <h1>This is contact page</h1>
</body>
</html>
```

breadcrumbAbout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MP
K8M2HN" crossorigin="anonymous">
<body>
     <nav aria-label="breadcrumb">
       <ol class="breadcrumb">
          <li class="breadcrumb-item"> <a href="breadcrumbHome.html"> Home </a> </li>
         <li class="breadcrumb-item active">About</li>
         <li class="breadcrumb-item" aria-current="page"><a
href="breadcrumbContact.html">Contact</a></li>
       </ol>
    </nav>
    <h1>This is about page</h1>
</body>
</html>
```

9) NodeJS FS Asynchronous and Synchronous

Make a file example.txt in the same dir

In AsynchronousFS.js

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

// Asynchronous file read
fs.readFile("example.txt", "utf8", (err, data) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log("Asynchronous File Content:");
  console.log(data);
});
console.log("Reading file asynchronously...");
```

Output:
Reading file asynchronously...
Asynchronous File Content:
Hello this is the example . txt file
required for the FS module of NODEJS

In SynchronousFS.js

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

// Synchronous file read
try {
  const data = fs.readFileSync("example.txt", "utf8");
  console.log("Synchronous File Content:");
  console.log(data);
} catch (err) {
  console.error(err);
}
console.log("Reading file synchronously...");
```

Output:
Synchronous File Content:
Hello this is the example . txt file
required for the FS module of NODEJS
Reading file synchronously...

10) Take i/p from a form calculate BMI and give results based on the data table

11) Props, State example

In App.js

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

function WelcomeMessage(props) {
  return (
    <div>
      <h1>Props Example</h1>
      <p>Hello, {props.name}!</p>
    </div>
  );
}

function Counter() {
  // Declare a state variable named 'count' with an initial value of 0
  const [count, setCount] = useState(0);

  return (
    <div>
      <h1>React Hooks Example</h1>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
    </div>
  );
}
```

```
function App() {
  return (
    <div>
      <WelcomeMessage name="Shashwat" />
      <Counter />
    </div>
  );
}

export default App;
```

12) Node Routing

13) Make a kid accessories website using bootstrap tooltips and Breadcrumb

14) Javascript: Take form input child name, DOB, height, weight

a) calculate age by giving DOB as input

b) tell whether the child is underweight or overweight by age and weight

Eg

if the newborn baby age 0 years 0 months

so if its weight is less than 3.3 kg then underweight more than 3.3 kg then overweight

similarly table will display results for babies 3 months 6 months 9 months 1 year

15) node program to perform synchronous and asynchronous operations on the file

16) Write a Javascript program for menu-driven categories that enable you to perform the following set of tasks

1) To calculate the Number of links in a webpage

2) Design a drop-down option consisting of colors, and upon choosing the color from the down it should change the background color of the webpage

```
<!DOCTYPE html>
<html>
<head>
    <title>Background Color Changer</title>
</head>
<body>
    <h1>Background Color Changer</h1>

    <label for="colorSelect">Select a Color:</label>
    <select id="colorSelect">
        <option value="white">White</option>
        <option value="red">Red</option>
        <option value="blue">Blue</option>
        <option value="green">Green</option>
        <option value="yellow">Yellow</option>
    </select>

    <script>
        const colorSelect = document.getElementById('colorSelect');

        // Function to change the background color
        function changeBackgroundColor() {
            const selectedColor = colorSelect.value;
            document.body.style.backgroundColor = selectedColor;
```

```
        }

        // Event listener to call the changeBackgroundColor function when a color is selected
        colorSelect.addEventListener('change', changeBackgroundColor);
    </script>
</body>
</html>
```

3)Create a Registration Form and use data validation using Regular Expression upon Registration, it should display Registered Successful or Details not entered or wrong mobile number/email ID/d.o.b depending on the validation and that message should be displayed as a pop-up with a timeout.

17) Create a sharemarket website using html css. Take a person's name,address, phone no,share purchase value, quantity, share current value. Calculate the profit/loss