# WC Practical

- To design and Simulate the environment for Dynamic routing using Cisco packet tracer/GNS3

~~~
Here’s a **step-by-step guide** for designing and simulating **Dynamic Routing** using **Cisco Packet Tracer or GNS3**. I’ll keep it **very beginner-friendly**.

---

## **Step 1: Plan Your Topology**

Dynamic routing requires **multiple routers** so they can automatically exchange routes.

Example topology:

```
    [Router1]---[Router2]---[Router3]
       |           |           |
      PC1         PC2         PC3
```

* Each router has at least two networks.
* PCs are endpoints for testing connectivity.

---

## **Step 2: Add Devices**

1. Open **Cisco Packet Tracer / GNS3**.
2. Add **3 routers** (e.g., 1941).
3. Add **3 switches** (optional, for connecting PCs).
4. Add **3 PCs** as endpoints.
~~~

```
---

## **Step 3: Connect Devices**

1. Use **copper straight-through cables**:

   * Router → Router (interface to interface)
   * Router → Switch → PC

2. Example connections in Packet Tracer:

   * Router1 Fa0/0 → Router2 Fa0/0
   * Router2 Fa0/1 → Router3 Fa0/0
   * Router1 Fa0/1 → Switch → PC1
   * Router2 Fa0/2 → Switch → PC2
   * Router3 Fa0/1 → Switch → PC3

---

## **Step 4: Assign IP Addresses**

* Each router interface connected to a network needs an IP.

| Device  | Interface | IP Address  | Subnet Mask   |
| ------- | --------- | ----------- | ------------- |
| Router1 | Fa0/0     | 10.0.12.1   | 255.255.255.0 |
| Router1 | Fa0/1     | 192.168.1.1 | 255.255.255.0 |
| Router2 | Fa0/0     | 10.0.12.2   | 255.255.255.0 |
| Router2 | Fa0/1     | 10.0.23.1   | 255.255.255.0 |
| Router2 | Fa0/2     | 192.168.2.1 | 255.255.255.0 |
| Router3 | Fa0/0     | 10.0.23.2   | 255.255.255.0 |
| Router3 | Fa0/1     | 192.168.3.1 | 255.255.255.0 |

* PCs get IPs in their respective LAN:

  * PC1 → 192.168.1.2 / GW: 192.168.1.1
  * PC2 → 192.168.2.2 / GW: 192.168.2.1
  * PC3 → 192.168.3.2 / GW: 192.168.3.1
```

~~~
---

## **Step 5: Enable Dynamic Routing**

* Use **RIP** (simplest dynamic routing protocol for beginners).

**Router1 CLI:**

```plaintext
Router> enable
Router# configure terminal
Router(config)# router rip
Router(config-router)# version 2
Router(config-router)# network 10.0.0.0
Router(config-router)# network 192.168.1.0
Router(config-router)# exit
```

**Router2 CLI:**

```plaintext
Router> enable
Router# configure terminal
Router(config)# router rip
Router(config-router)# version 2
Router(config-router)# network 10.0.0.0
Router(config-router)# network 192.168.2.0
Router(config-router)# exit
```

**Router3 CLI:**

```plaintext
Router> enable
Router# configure terminal
Router(config)# router rip
Router(config-router)# version 2
~~~

```
Router(config-router)# network 10.0.0.0
Router(config-router)# network 192.168.3.0
Router(config-router)# exit
```

---

## **Step 6: Test Connectivity**

1. Ping from PC1 → PC2 → PC3.
2. If dynamic routing is working, all PCs should reach each other **without manually adding static routes**.

---

## **Step 7: Verification Commands**

* On Router CLI:

```plaintext
show ip route        # Shows routes learned dynamically
show running-config  # Verify RIP configuration
ping <other PC IP>   # Test connectivity
```

---

### **Tips for Beginners**

* Start with **3 routers + 3 PCs** before adding more complexity.
* Always **check IP addresses** and subnet masks.
* Make sure the router interfaces are **up (`no shutdown`)**.
* RIP automatically shares routes every 30 seconds in Packet Tracer.

---

I can also make a **super simple text diagram labeled with IPs and interfaces** for this dynamic routing setup so it’s very easy to follow.

~~~
Do you want me to do that?
Here's a **super simple text diagram** for a **basic dynamic routing setup
** with 3 routers, 3 PCs, and RIP:

```
      [Router1]----------------[Router2]----------------[Router3]
     Fa0/0 10.0.12.1        Fa0/0 10.0.12.2       Fa0/0 10.0.23.2
     Fa0/1 192.168.1.1      Fa0/1 10.0.23.1       Fa0/1 192.168.3.1
                      Fa0/2 192.168.2.1

        |                    |                    |
       PC1                  PC2                  PC3
   192.168.1.2/24       192.168.2.2/24       192.168.3.2/24
   GW: 192.168.1.1      GW: 192.168.2.1      GW: 192.168.3.1
```

### **Step-by-step Mapping:**

1. **Router1**

  * Fa0/0 → connects to Router2 (10.0.12.0/24)
  * Fa0/1 → LAN for PC1 (192.168.1.0/24)

2. **Router2**

  * Fa0/0 → connects to Router1 (10.0.12.0/24)
  * Fa0/1 → connects to Router3 (10.0.23.0/24)
  * Fa0/2 → LAN for PC2 (192.168.2.0/24)

3. **Router3**

  * Fa0/0 → connects to Router2 (10.0.23.0/24)
  * Fa0/1 → LAN for PC3 (192.168.3.0/24)

4. **PCs**

  * PC1 → 192.168.1.2 / GW: 192.168.1.1
~~~

```
   * PC2 → 192.168.2.2 / GW: 192.168.2.1
   * PC3 → 192.168.3.2 / GW: 192.168.3.1

---

### **Dynamic Routing**

* Configure **RIP v2** on all routers.

* Use networks:

  * Router1: `10.0.12.0`, `192.168.1.0`
  * Router2: `10.0.12.0`, `10.0.23.0`, `192.168.2.0`
  * Router3: `10.0.23.0`, `192.168.3.0`

* After RIP is running, all PCs can **ping each other automatically** without static routes.

---

If you want, I can also make a **step-by-step “what to click in Packet Tracer” beginner guide** for this setup, so you can **build it visually without CLI confusion**.

Do you want me to do that?
```

---

- To design and Simulate VLANs on the switch/router using Cisco packet tracer/GNS3

```
Got it! I’ll give you a **very basic, step-by-step guide** to set up VLANs in **Cisco Packet Tracer**, assuming you are a beginner and haven’t used it before.

---
```

~~~
## **Step 1: Open Cisco Packet Tracer**

1. Download and install Cisco Packet Tracer (if not installed).
2. Open Packet Tracer.

---

## **Step 2: Add Devices**

1. Go to the **“End Devices”** panel and drag **3 PCs** to the workspace.
2. Go to the **“Switches”** panel and drag **1 switch** (e.g., `2960`).
3. Go to the **“Routers”** panel and drag **1 router** (e.g., `1941`).

---

## **Step 3: Connect Devices**

1. Select **“Connections” → Copper Straight-Through** cable.
2. Connect PCs to switch:

   * PC1 → Fa0/1
   * PC2 → Fa0/2
   * PC3 → Fa0/3
3. Connect switch to router:

   * Switch Fa0/24 → Router Fa0/0

---

## **Step 4: Configure VLANs on Switch**

1. Click the **Switch → CLI** tab.
2. Enter commands:

```plaintext
Switch> enable
Switch# configure terminal
~~~

~~~
# Create VLANs
Switch(config)# vlan 10
Switch(config-vlan)# name Sales
Switch(config-vlan)# exit

Switch(config)# vlan 20
Switch(config-vlan)# name HR
Switch(config-vlan)# exit

Switch(config)# vlan 30
Switch(config-vlan)# name IT
Switch(config-vlan)# exit

# Assign ports to VLANs
Switch(config)# interface fastEthernet 0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 10
Switch(config-if)# exit

Switch(config)# interface fastEthernet 0/2
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 20
Switch(config-if)# exit

Switch(config)# interface fastEthernet 0/3
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 30
Switch(config-if)# exit
```

---

## **Step 5: Configure Router for Inter-VLAN Routing**

1. Click the **Router → CLI** tab.
2. Enter commands:

```plaintext
~~~

~~~
Router> enable
Router# configure terminal

# Enable router interface
Router(config)# interface fastEthernet 0/0
Router(config-if)# no shutdown
Router(config-if)# exit

# Create sub-interfaces for each VLAN
Router(config)# interface fastEthernet 0/0.10
Router(config-subif)# encapsulation dot1Q 10
Router(config-subif)# ip address 192.168.10.1 255.255.255.0
Router(config-subif)# exit

Router(config)# interface fastEthernet 0/0.20
Router(config-subif)# encapsulation dot1Q 20
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
Router(config-subif)# exit

Router(config)# interface fastEthernet 0/0.30
Router(config-subif)# encapsulation dot1Q 30
Router(config-subif)# ip address 192.168.30.1 255.255.255.0
Router(config-subif)# exit
```

---

## **Step 6: Assign IP Addresses to PCs**

1. Click each PC → **Desktop → IP Configuration**.
2. Enter the following:

| PC  | IP Address   | Subnet Mask   | Default Gateway |
| --- | ------------ | ------------- | --------------- |
| PC1 | 192.168.10.2 | 255.255.255.0 | 192.168.10.1    |
| PC2 | 192.168.20.2 | 255.255.255.0 | 192.168.20.1    |
| PC3 | 192.168.30.2 | 255.255.255.0 | 192.168.30.1    |
~~~

~~~
---

## **Step 7: Test Connectivity**

1. Click a PC → **Desktop → Command Prompt**.
2. Test ping:

```plaintext
PC1> ping 192.168.10.2   # Same VLAN → should work
PC1> ping 192.168.20.2   # Different VLAN → works if router configured
PC1> ping 192.168.30.2   # Different VLAN → works if router configured
```

---

## **Step 8: Verify Configuration**

* On Switch CLI:

```plaintext
show vlan brief
show interfaces status
```

* On Router CLI:

```plaintext
show ip interface brief
ping <PC IP>
```

---

### **Summary**

* **Switch** → VLANs created and ports assigned.
* **Router** → Sub-interfaces for inter-VLAN routing.
* **PCs** → IP addresses in VLAN subnet.
~~~

~~~
* **Test** → Ping to check connectivity.

---



Perfect! Here’s a **super simple text diagram** for **one switch, 3 VLANs,
3 PCs**, which you can directly replicate in Cisco Packet Tracer:

```
              [ Router ]
                Fa0/0
                  |
              Trunk Link
                  |
               [ Switch ]
       -------------------------------
       |           |             |
     Fa0/1       Fa0/2        Fa0/3
     VLAN10       VLAN20        VLAN30
     PC1         PC2           PC3
  192.168.10.2   192.168.20.2   192.168.30.2
  Gateway:       Gateway:       Gateway:
 192.168.10.1   192.168.20.1   192.168.30.1
```

### **How to Set Up in Packet Tracer**

1. **Router → Switch**

  * Use **copper straight-through**.
  * Configure router interface as **trunk** (for VLAN tagging).

2. **Switch → PCs**

  * Connect Fa0/1 → PC1 (VLAN10)
  * Connect Fa0/2 → PC2 (VLAN20)
  * Connect Fa0/3 → PC3 (VLAN30)
~~~

```
3. **Switch VLAN Config**

  * Create VLAN10, VLAN20, VLAN30.
  * Assign ports Fa0/1, Fa0/2, Fa0/3 to respective VLANs.

4. **Router Sub-Interfaces for Inter-VLAN Routing**

  * Fa0/0.10 → 192.168.10.1 (VLAN10)
  * Fa0/0.20 → 192.168.20.1 (VLAN20)
  * Fa0/0.30 → 192.168.30.1 (VLAN30)

5. **PC IP Settings**

  * PC1 → 192.168.10.2 / 255.255.255.0 / GW: 192.168.10.1
  * PC2 → 192.168.20.2 / 255.255.255.0 / GW: 192.168.20.1
  * PC3 → 192.168.30.2 / 255.255.255.0 / GW: 192.168.30.1

6. **Test**

  * Ping within same VLAN → works.
  * Ping between VLANs → works if router sub-interfaces configured.

---

If you want, I can also make a **step-by-step “what to click in Packet Tracer” guide** for each connection and IP configuration so it’s **really beginner-friendly**.

Do you want me to do that?
```

- Write a program to execute HTML links, Images and tables using JavaScript

```html
<!DOCTYPE html>
<html>
<head>
  <title>HTML Links, Images & Tables with JS</title>
  <style>
    table, th, td {
      border: 1px solid black;
      border-collapse: collapse;
      padding: 8px;
    }
    table {
      margin-top: 20px;
    }
    img {
      margin-top: 20px;
      width: 150px;
      height: auto;
    }
    a {
      display: block;
      margin-top: 20px;
      font-size: 18px;
    }
  </style>
</head>
<body>

  <h2>HTML Links, Images & Tables using JavaScript</h2>

  <div id="content"></div>

  <script>
    const contentDiv = document.getElementById("content");

    // 1. Add a Link
    const link = document.createElement("a");
    link.href = "https://www.example.com";
    link.target = "_blank"; // Open in new tab
    link.innerText = "Visit Example.com";
    contentDiv.appendChild(link);

    // 2. Add an Image
    const img = document.createElement("img");
    img.src = "https://via.placeholder.com/150";
    img.alt = "Placeholder Image";
    contentDiv.appendChild(img);

    // 3. Add a Table
    const table = document.createElement("table");

    // Table header
    const header = table.insertRow();
    const th1 = document.createElement("th");
    th1.innerText = "Name";
    const th2 = document.createElement("th");
    th2.innerText = "Age";
    header.appendChild(th1);
    header.appendChild(th2);

    // Table data
    const students = [
      {name: "Shravani", age: 20},
      {name: "John", age: 22},
      {name: "Alice", age: 19}
    ];

    students.forEach(student => {
      const row = table.insertRow();
      const cell1 = row.insertCell();
      cell1.innerText = student.name;
      const cell2 = row.insertCell();
      cell2.innerText = student.age;
    });

    contentDiv.appendChild(table);
  </script>

</body>
</html>
```

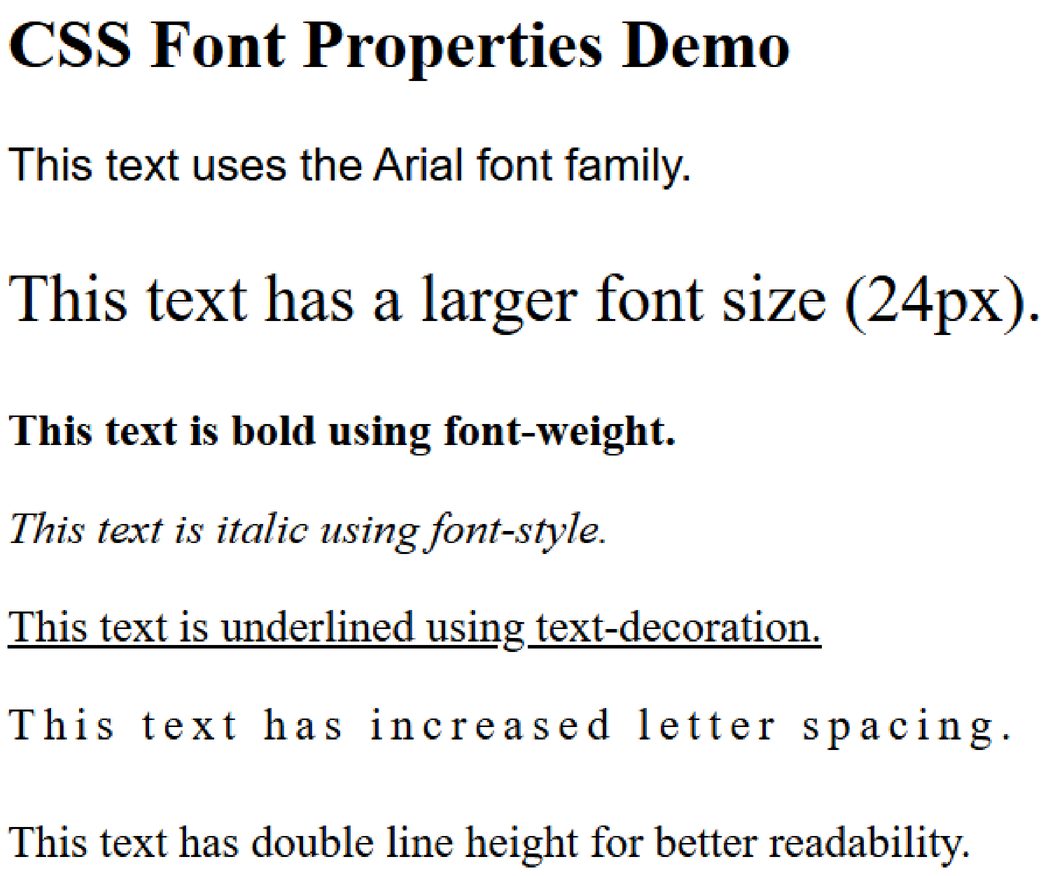
- Write a program to execute Student Login Form in JavaScript using HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Login Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f2f2f2;
    }
    .login-form {
      background-color: white;
      padding: 20px 30px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0,0,0,0.2);
    }
    input {
      width: 100%;
      padding: 10px;
      margin: 8px 0;
      box-sizing: border-box;
    }
    button {
      width: 100%;
      padding: 10px;
      background-color: #4CAF50;
      color: white;
      border: none;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
    }
  </style>
</head>
<body>

  <div class="login-form">
    <h2>Student Login</h2>
    <form id="studentLogin">
      <input type="text" id="studentID" placeholder="Enter Student ID" required><br>
      <input type="password" id="password" placeholder="Enter Password" required><br>
      <button type="submit">Login</button>
    </form>
    <p id="message"></p>
  </div>

  <script>
    document.getElementById('studentLogin').addEventListener('submit', function(e) {
      e.preventDefault(); // Prevent form submission

      const studentID = document.getElementById('studentID').value;
      const password = document.getElementById('password').value;

      // Simple validation (you can enhance it)
      if(studentID === "12345" && password === "password") {
        document.getElementById('message').style.color = "green";
        document.getElementById('message').innerText = "Login Successful!";
      } else {
        document.getElementById('message').style.color = "red";
        document.getElementById('message').innerText = "Invalid Student ID or Password.";
      }
    });
  </script>

</body>
</html>
```

---

- Write a program to execute different CSS font properties

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS Font Properties Example</title>
  <style>
    /* 1. Font family */
    .font-family {
      font-family: Arial, sans-serif;
      font-size: 16px;
    }

    /* 2. Font size */
    .font-size {
      font-size: 24px;
    }

    /* 3. Font weight */
    .font-weight {
      font-weight: bold;
    }

    /* 4. Font style */
    .font-style {
      font-style: italic;
    }

    /* 5. Text decoration */
    .text-decoration {
      text-decoration: underline;
    }

    /* 6. Letter spacing */
    .letter-spacing {
      letter-spacing: 3px;
    }

    /* 7. Line height */
    .line-height {
      line-height: 2;
    }
  </style>
</head>
<body>
  <h2>CSS Font Properties Demo</h2>

  <p class="font-family">This text uses the Arial font family.</p>
  <p class="font-size">This text has a larger font size (24px).</p>
  <p class="font-weight">This text is bold using font-weight.</p>
  <p class="font-style">This text is italic using font-style.</p>
  <p class="text-decoration">This text is underlined using text-decoration.</p>
  <p class="letter-spacing">This text has increased letter spacing.</p>
  <p class="line-height">This text has double line height for better readability.</p>

</body>
</html>
```



# CSS Font Properties Demo

This text uses the Arial font family.

This text has a larger font size (24px).

**This text is bold using font-weight.**

*This text is italic using font-style.*

This text is underlined using text-decoration.

This text has increased letter spacing.

This text has double line height for better readability.

---

- Write a program to execute A navigation bar at the top with links to various sections of the dashboard (e.g., Home, Products, Orders, Customers, Reports).

```
<!DOCTYPE html>
<html>
<head>
  <title>Dashboard Navigation Bar</title>
  <style>
    /* Navbar container */
    .navbar {
      background-color: #333;
      overflow: hidden;
      position: fixed; /* Keep it at the top */
      top: 0;
      width: 100%;
      z-index: 1000;
    }

    /* Navbar links */
    .navbar a {
      float: left;
      display: block;
      color: white;
      text-align: center;
      padding: 14px 20px;
      text-decoration: none;
      font-size: 17px;
    }

    /* Hover effect */
    .navbar a:hover {
      background-color: #575757;
      color: white;
    }

    /* Page content */
    .content {
      padding: 70px 20px; /* Add top padding to avoid navbar overlap */
    }
  </style>
</head>
<body>

  <!-- Navigation Bar -->
  <div class="navbar">
    <a href="#home">Home</a>
    <a href="#products">Products</a>
    <a href="#orders">Orders</a>
    <a href="#customers">Customers</a>
    <a href="#reports">Reports</a>
  </div>

  <!-- Main Content →
  <div class="content">
    <h1>Dashboard</h1>
    <section id="home"><h2>Home Section</h2><p>Welcome to the dashboard home.</p></section>
    <section id="products"><h2>Products Section</h2><p>Manage your products here.</p></section>
    <section id="orders"><h2>Orders Section</h2><p>Check and manage orders.</p></section>
    <section id="customers"><h2>Customers Section</h2><p>View customer details.</p></section>
    <section id="reports"><h2>Reports Section</h2><p>Analyze your reports.</p></section>
  </div>

</body>
</html>
```

---

- Write a program to execute different CSS3 selectors(minimum 4)

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS3 Selectors Example</title>
  <style>
    /* 1. Element selector */
    p {
      color: blue;
      font-size: 16px;
    }

    /* 2. Class selector */
    .highlight {
      background-color: yellow;
      font-weight: bold;
    }

    /* 3. ID selector */
    #special {
      color: red;
      font-size: 20px;
    }

    /* 4. Attribute selector */
    a[target="_blank"] {
      color: green;
      text-decoration: underline;
    }

    /* 5. Pseudo-class selector */
    li:hover {
      color: orange;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <h2>CSS3 Selectors Demo</h2>

  <!-- Element selector -->
  <p>This paragraph uses an element selector (p).</p>

  <!-- Class selector -->
  <p class="highlight">This paragraph uses a class selector (.highlight).</p>

  <!-- ID selector -->
  <p id="special">This paragraph uses an ID selector (#special).</p>

  <!-- Attribute selector -->
  <a href="https://www.example.com" target="_blank">Link opens in a new
```

```
tab</a>

  <!-- Pseudo-class selector -->
  <ul>
    <li>Hover over me!</li>
    <li>Hover over me too!</li>
  </ul>
</body>
</html>
```

---

- Write a program to execute Student Login Form using Bootstrap

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Login Form</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstra
p.min.css" rel="stylesheet">
</head>
<body>

  <div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-4">
        <div class="card">
          <div class="card-header text-center bg-primary text-white">
            <h3>Student Login</h3>
          </div>
          <div class="card-body">
            <form id="loginForm">
              <div class="mb-3">
                <label for="studentID" class="form-label">Student ID</label>
                <input type="text" class="form-control" id="studentID" placehol
der="Enter Student ID" required>
              </div>
              <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" placeholder="Enter Password" required>
              </div>
              <button type="submit" class="btn btn-primary w-100">Login</button>
            </form>
          </div>
          <div class="card-footer text-center">
            <small>&copy; 2025 Student Portal</small>
          </div>
        </div>
      </div>
    </div>
  </div>

  <!-- Bootstrap JS Bundle -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

  <!-- Optional JS to handle form submission -->
  <script>
    document.getElementById('loginForm').addEventListener('submit', function(e) {
      e.preventDefault(); // Prevent form from submitting
      const studentID = document.getElementById('studentID').value;
      const password = document.getElementById('password').value;
      alert(`Student ID: ${studentID} registred successfully`);
    });
  </script>

</body>
</html>
```

- Write a program to execute Simple Navigation Bar using Bootstrap

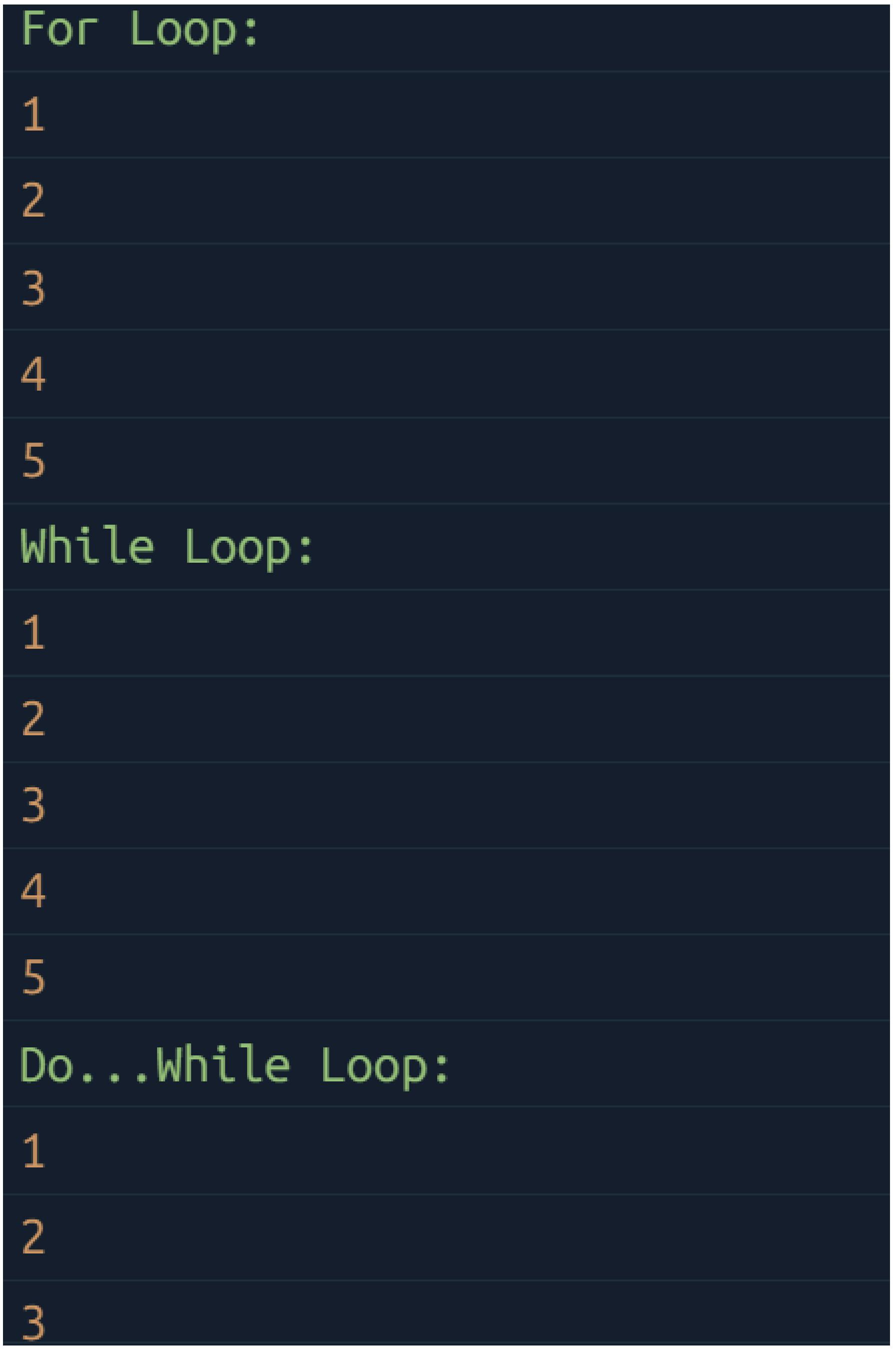
```
<!DOCTYPE html>
<html>
<head>
  <title>Bootstrap Navbar Example</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

  <!-- Navigation Bar -->
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">MyWebsite</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="#">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">About</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Services</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Contact</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

  <div class="container mt-5">
    <h1>Welcome to My Website</h1>
    <p>This is a simple Bootstrap navigation bar example.</p>
  </div>

  <!-- Bootstrap JS Bundle -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

---

- Write a program to execute different loops using JavaScript

```
// 1. For loop
console.log("For Loop:");
for (let i = 1; i <= 5; i++) {
  console.log(i);
}

// 2. While loop
console.log("While Loop:");
let j = 1;
while (j <= 5) {
  console.log(j);
  j++;
}

// 3. Do...While loop
console.log("Do...While Loop:");
let k = 1;
```

```
do {
  console.log(k);
  k++;
} while (k <= 5);

// 4. For...of loop (array)
const fruits = ["Apple", "Banana", "Cherry"];
console.log("For...of Loop:");
for (let fruit of fruits) {
  console.log(fruit);
}

// 5. For...in loop (object)
const person = { name: "Shravani", age: 20 };
console.log("For...in Loop:");
for (let key in person) {
  console.log(key + " = " + person[key]);
}
```

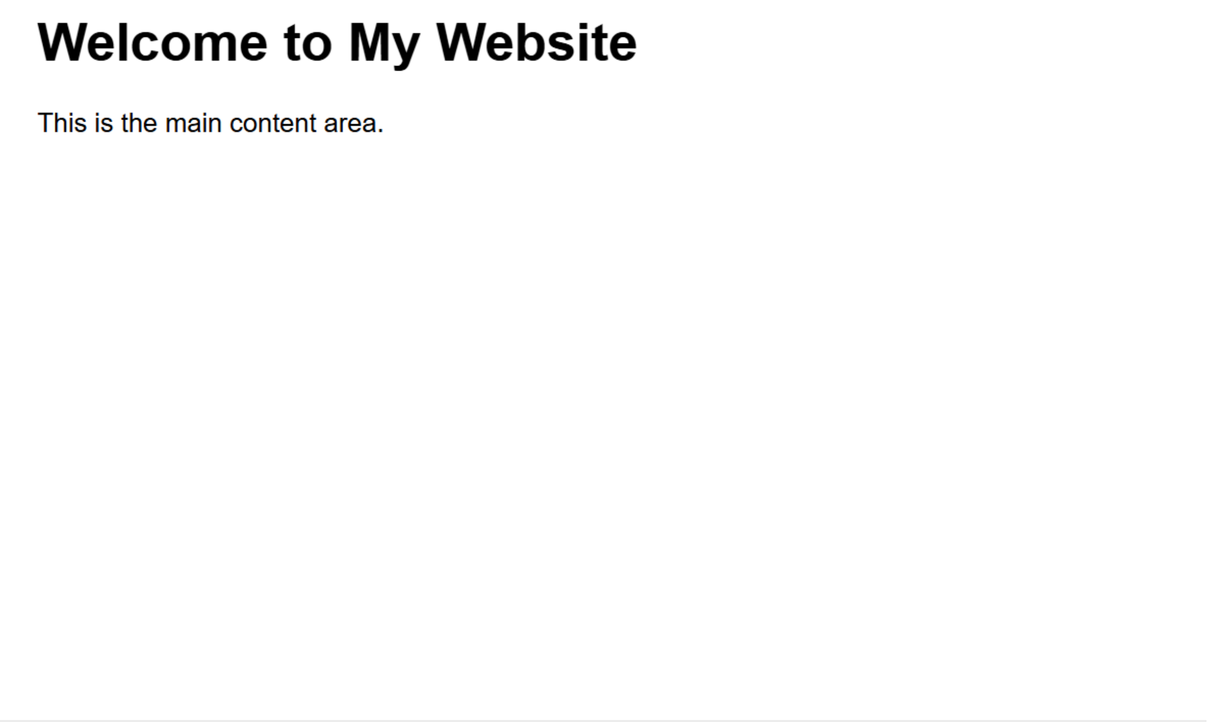


```
For Loop:
1
2
3
4
5
While Loop:
1
2
3
4
5
Do...While Loop:
1
2
3
```

- Write a program to execute a footer at the bottom of the page with copyright information.

```
<!DOCTYPE html>
<html>
<head>
  <title>Footer Example</title>
  <style>
```

```
    /* Basic page styling */
    body {
      margin: 0;
      font-family: Arial, sans-serif;
      min-height: 100vh;
      display: flex;
      flex-direction: column;
    }

    /* Main content */
    .content {
      flex: 1; /* Take remaining space */
      padding: 20px;
    }

    /* Footer styling */
    footer {
      background-color: #333;
      color: white;
      text-align: center;
      padding: 10px 0;
    }
  </style>
</head>
<body>

  <div class="content">
    <h1>Welcome to My Website</h1>
    <p>This is the main content area.</p>
  </div>

  <footer>
    &copy; 2025 MyWebsite. All rights reserved.
  </footer>

</body>
</html>
```

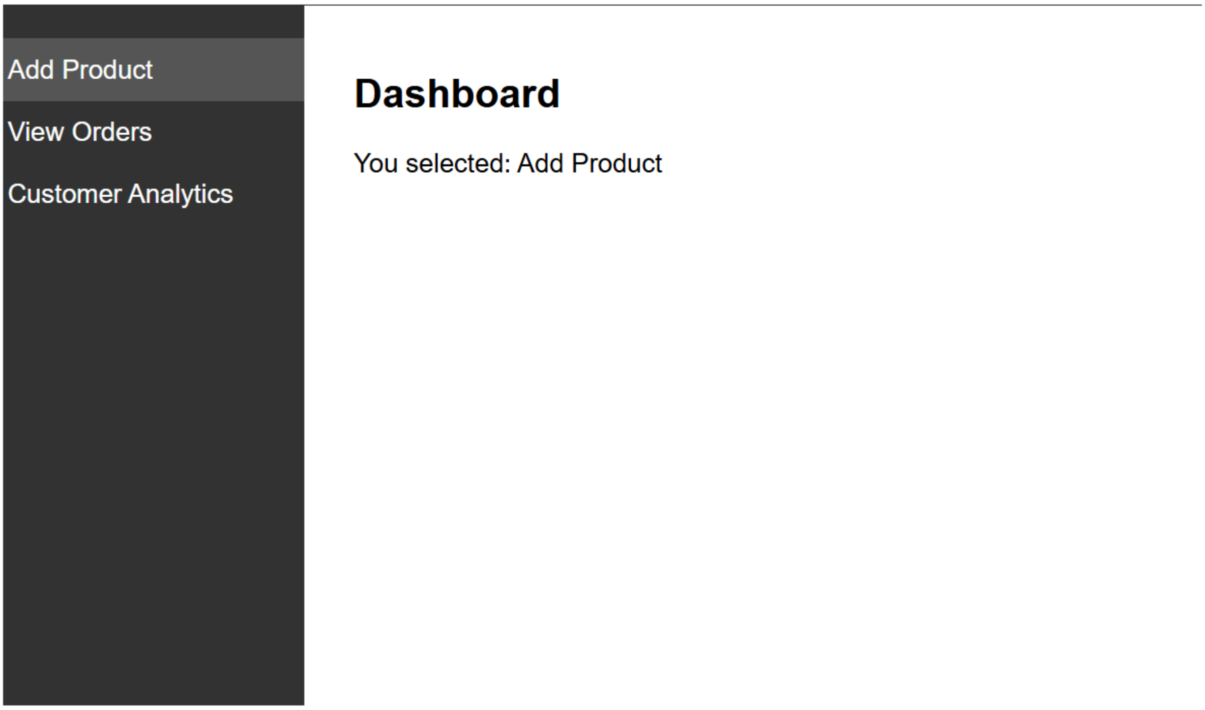


Welcome to My Website

This is the main content area.

- Write a program to execute a side menu on the left that displays quick links to key features (e.g., Add Product, View Orders, Customer Analytics).

```
<!DOCTYPE html>
<html>
<head>
  <title>Side Menu Example</title>
  <style>
    /* Basic page layout */
    body {
      margin: 0;
      font-family: Arial, sans-serif;
    }

    /* Side menu styling */
    .side-menu {
      height: 100%;
      width: 200px;
      position: fixed;
      top: 0;
      left: 0;
      background-color: #333;
      padding-top: 20px;
    }

    .side-menu a {
      display: block;
      color: white;
      padding: 10px 20px;
      text-decoration: none;
    }

    .side-menu a:hover {
      background-color: #575757;
    }

    /* Main content */
    .main {
      margin-left: 210px;
      padding: 20px;
    }
  </style>
</head>
<body>
  <!-- Side Menu -->
  <div class="side-menu">
    <a href="#" onclick="showFeature('Add Product')">Add Product</a>
    <a href="#" onclick="showFeature('View Orders')">View Orders</a>
    <a href="#" onclick="showFeature('Customer Analytics')">Customer Analytics</a>
  </div>

  <!-- Main content area -->
  <div class="main">
    <h2>Dashboard</h2>
    <p id="content">Select a feature from the left menu.</p>
  </div>
  <script>
    function showFeature(feature) {
      document.getElementById("content").innerHTML = `You selected: ${feature}`;
    }
  </script>
</body>
</html>
```



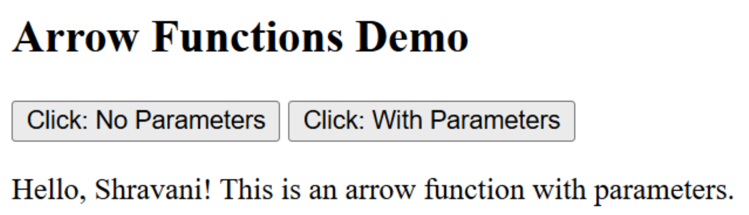
Dashboard

You selected: Add Product

- Write a program to execute Arrow function with Parameters and without Parameters using JavaScript

```
<!DOCTYPE html>
<html>
<head>
  <title>Arrow Functions Example</title>
</head>
<body>
```

```
  <h2>Arrow Functions Demo</h2>

  <button onclick="arrowNoParams()">Click: No Parameters</button>
  <button onclick="arrowWithParams()">Click: With Parameters</button>

  <p id="output"></p>

  <script>
    // Arrow function without parameters
    const greet = () => {
      document.getElementById("output").innerHTML = "Hello! This is an arrow function without parameters.";
    };

    // Arrow function with parameters
    const greetPerson = (name) => {
      document.getElementById("output").innerHTML = `Hello, ${name}! This is an arrow function with parameters.`;
    };

    // Functions triggered on button click
    function arrowNoParams() {
      greet();
    }

    function arrowWithParams() {
      greetPerson("Shravani");
    }
  </script>
</body>
</html>
```



Arrow Functions Demo

Hello, Shravani! This is an arrow function with parameters.

---

- Install and Configure React and Write a program to create a component that accepts and displays props

## Step 1: Install Node.js

Make sure Node.js is installed. Verify by running:

```
node -v
npm -v
```

---

## Step 2: Create a React App

Open your terminal and run:

```
npx create-react-app my-app
```

This will create a new React project called `my-app`.

Go into the project folder:

```
cd my-app
```

Start the development server:

```
npm start
```

Your default React app will open at `http://localhost:3000`.

## Step 3: Create a Component that Accepts Props

1. Inside the `src` folder, create a new file called `Greeting.js`.

```
// Greeting.js
import React from 'react';

function Greeting(props) {
  return (
    <div>
      <h2>Hello, {props.name}!</h2>
      <p>Welcome to React props example.</p>
    </div>
  );
}

export default Greeting;
```

Update `App.js` to use the `Greeting` component and pass props:

```
// App.js
import React from 'react';
import Greeting from './Greeting';

function App() {
  return (
    <div className="App">
      <h1>React Props Demo</h1>
      <Greeting name="Shravani" />
      <Greeting name="John" />
      <Greeting name="Alice" />
    </div>
  );
}

export default App;
```

## Step 4: Run the App

If you haven't already started the server, run:

```
npm start
```

Open `http://localhost:3000` in your browser.

You'll see:

```
Hello, Shravani!
Hello, John!
Hello, Alice!
```

---

- Install and Configure Node.Js and Write a program to create a Basic HTTP Server.

## Step 1: Install Node.js

1. Go to the Node.js official website.
2. Download the **LTS (Long Term Support)** version for your operating system.
3. Run the installer and follow the prompts.
4. Verify installation by opening a terminal (Command Prompt / PowerShell) and typing:

```
node -v
npm -v
```

You should see the installed versions of Node.js and npm.

## Step 2: Create a Project Folder

1. Create a folder for your project, e.g., `BasicServer`.
2. Open the folder in your terminal.
3. Initialize a Node.js project (optional but recommended):

```
npm init -y
```

## Step 3: Write a Basic HTTP Server

Create a file named `server.js` inside your project folder:

```
// Load the http module
const http = require('http');

// Define server port
const PORT = 3000;

// Create HTTP server
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello! This is a basic HTTP server.\n');
});

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

## Step 4: Run the Server

In the terminal, run:

```
node server.js
```

You should see:

```
Server is running at http://localhost:3000
```

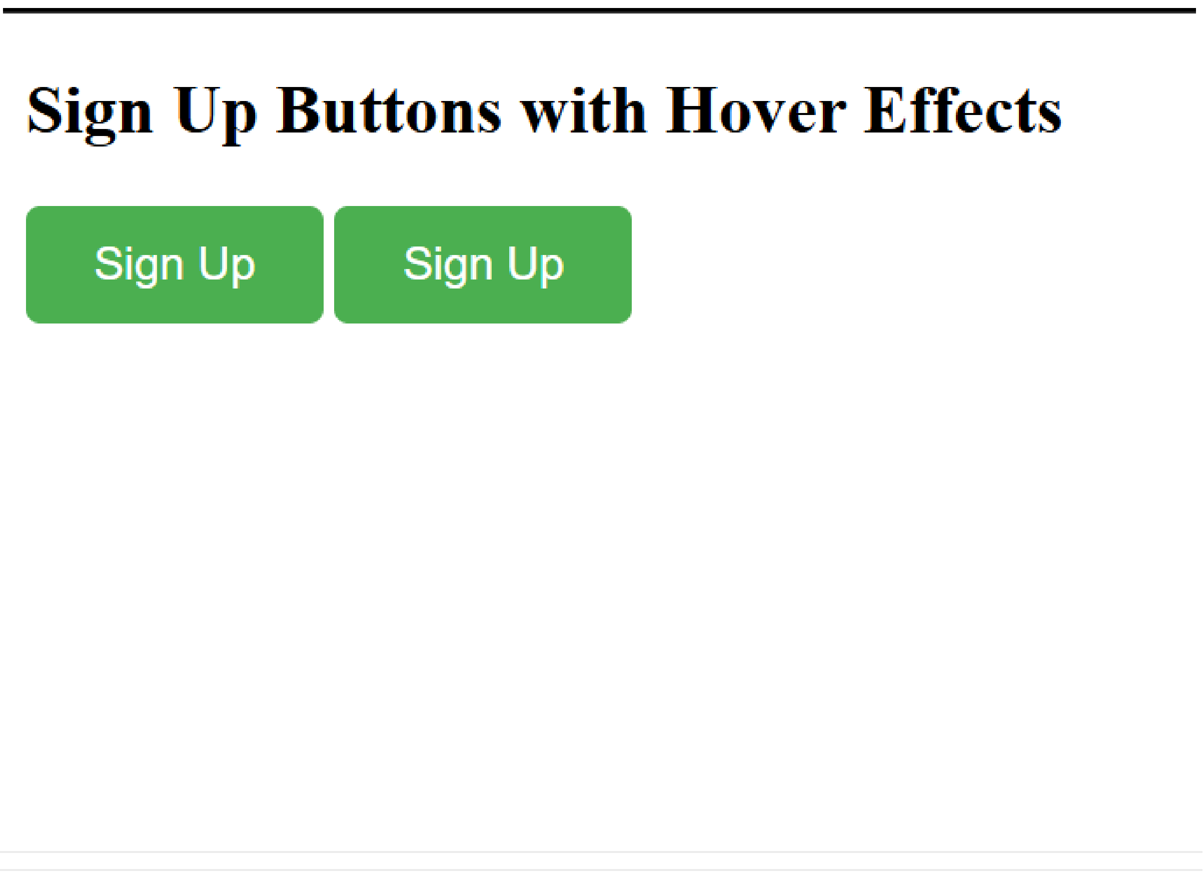
Open a browser and go to `http://localhost:3000` — you'll see the message:

```
Hello! This is a basic HTTP server
```

---

- Use CSS transitions or animations to add hover effects to the "Sign Up" buttons.

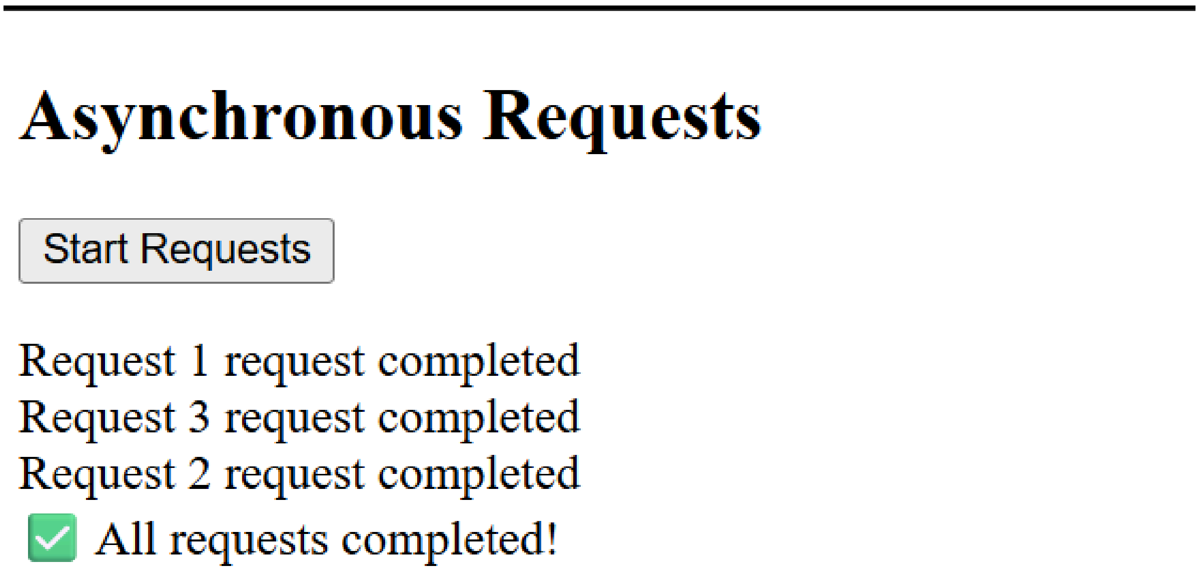
```
<!DOCTYPE html>
<html>
<head>
  <title>Sign Up Button Hover</title>
  <style>
    /* Basic button style */
    .signup-btn {
      background-color: #4CAF50; /* Green */
      color: white;
      padding: 12px 24px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      font-size: 16px;
      transition: background-color 0.3s ease, transform 0.3s ease;
    }

    /* Hover effect using transition */
    .signup-btn:hover {
      background-color: #45a049; /* Slightly darker green */
      transform: scale(1.1); /* Slight zoom */
    }
    /* Optional: Add a simple keyframe animation on hover */
    @keyframes pulse {
      0% { transform: scale(1); }
      50% { transform: scale(1.1); }
      100% { transform: scale(1); }
    }

    .signup-btn.animate:hover {
      animation: pulse 0.6s ease;
    }
  </style>
</head>
<body>
  <h2>Sign Up Buttons with Hover Effects</h2>

  <!-- Button with simple transition -->
  <button class="signup-btn">Sign Up</button>

  <!-- Button with transition + animation -->
  <button class="signup-btn animate">Sign Up</button>
</body>
</html>
```



Sign Up Buttons with Hover Effects

- Write a program that makes three asynchronous requests and executes a callback when all requests are complete.

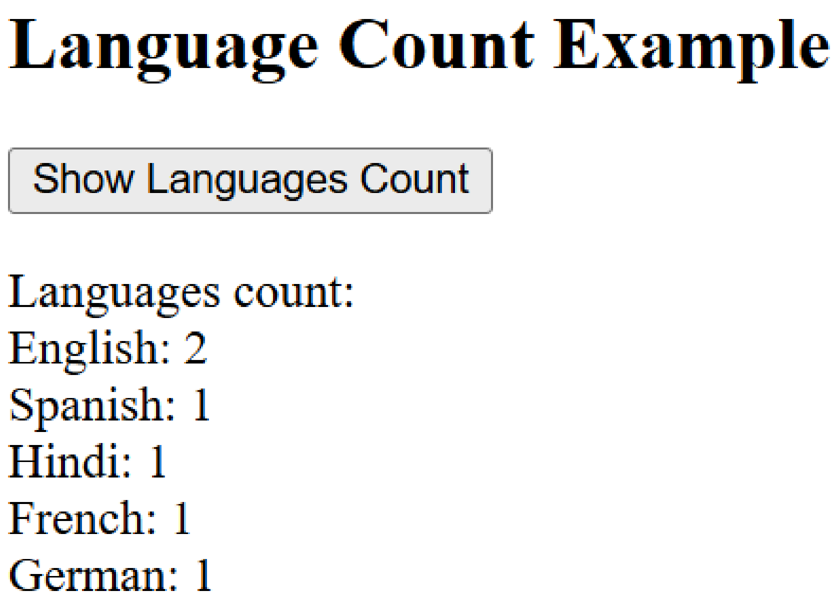
```
<!DOCTYPE html>
<html>
<head>
  <title>Async Requests Example</title>
</head>
<body>
  <h2>Asynchronous Requests</h2>
  <button onclick="makeRequests()">Start Requests</button>
  <p id="output"></p>

  <script>
    function asyncRequest(name, delay, callback) {
      setTimeout(() => {
        document.getElementById("output").innerHTML += name + " request completed<br>";
        callback();
      }, delay);
    }

    function makeRequests() {
      let completed = 0;

      function allDone() {
        document.getElementById("output").innerHTML += "✅ All requests completed!";
      }

      function checkCompletion() {
        completed++;
        if (completed === 3) {
          allDone();
        }
      }

      // Simulate three async requests
      asyncRequest("Request 1", 1000, checkCompletion);
      asyncRequest("Request 2", 2000, checkCompletion);
      asyncRequest("Request 3", 1500, checkCompletion);
    }
  </script>
</body>
</html>
```



```
Asynchronous Requests

[Start Requests]

Request 1 request completed
Request 3 request completed
Request 2 request completed
✅ All requests completed!
```

- Write a program to display number of languages and their count on click event of button

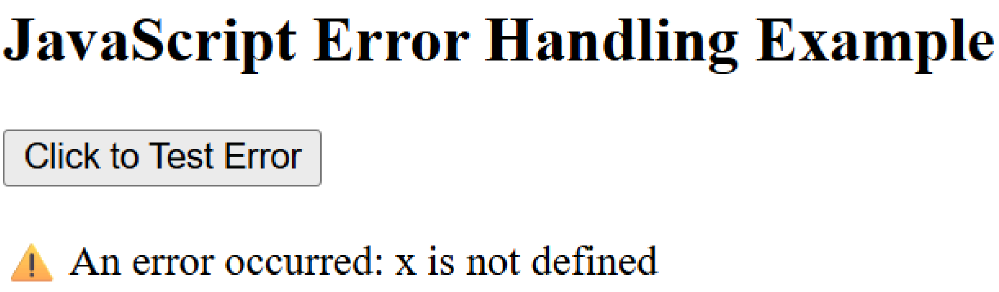
```
<!DOCTYPE html>
<html>
<head>
  <title>Language Counter</title>
</head>
<body>
  <h2>Language Count Example</h2>

  <button onclick="countLanguages()">Show Languages Count</button>

  <p id="output"></p>

  <script>
    // Sample array of languages
    const languages = ["English", "Spanish", "Hindi", "French", "German", "E
nglish"];

  function countLanguages() {
   // Create an object to store counts
   const counts = {};

   languages.forEach(lang => {
    counts[lang] = (counts[lang] || 0) + 1;
   });

   // Display results
   let message = "Languages count:<br>";
   for (let lang in counts) {
    message += `${lang}: ${counts[lang]}<br>`;
   }

   document.getElementById("output").innerHTML = message;
  }
 </script>
</body>
</html>
```



```
Languages count:
English: 2
Spanish: 1
Hindi: 1
French: 1
German: 1
```

- Write a JavaScript program that demonstrates error handling using try...catch blocks.

```
<!DOCTYPE html>
<html>
<head>
  <title>Try...Catch Example</title>
</head>
<body>
  <h2>JavaScript Error Handling Example</h2>

  <button onclick="handleError()">Click to Test Error</button>

  <p id="output"></p>

  <script>
    function handleError() {
      try {
        // Intentionally cause an error
        let result = 10 / x; // 'x' is not defined
        document.getElementById("output").innerHTML = "Result is " + result;
      }
      catch (error) {
        // Handle the error here
        document.getElementById("output").innerHTML =
          "⚠️ An error occurred: " + error.message;
      }
      finally {
        // Code here always runs, whether error occurs or not
        console.log("Execution completed.");
      }
    }
  </script>
</body>
</html>
```

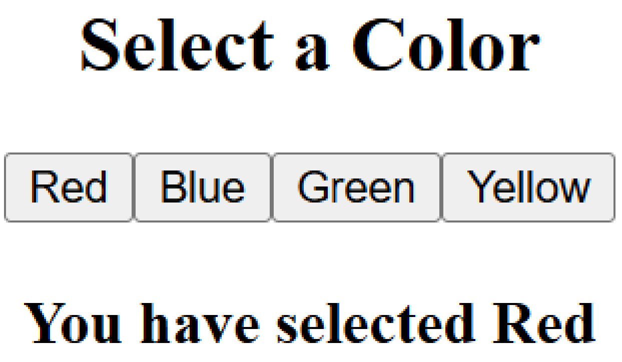


JavaScript Error Handling Example

⚠️ An error occurred: x is not defined

- Write a code making use of React Hooks that displays four buttons namely, “Red”, “Blue”, “Green”,“Yellow”. On clicking any of these buttons, the code displays the message that you have selected for that particular color.

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

function ColorSelector() {
  const [color, setColor] = useState("");

  const handleClick = (selectedColor) ⇒ {
    setColor(selectedColor);
  };

  return (
    <div style={{ textAlign: "center", marginTop: "50px" }}>
      <h2>Select a Color</h2>

      <button onClick={() ⇒ handleClick("Red")}>Red</button>
      <button onClick={() ⇒ handleClick("Blue")}>Blue</button>
      <button onClick={() ⇒ handleClick("Green")}>Green</button>
      <button onClick={() ⇒ handleClick("Yellow")}>Yellow</button>

      <h3 style={{ marginTop: "20px" }}>
        {color && `You have selected ${color}`}
      </h3>
    </div>
  );
}

export default ColorSelector;
```



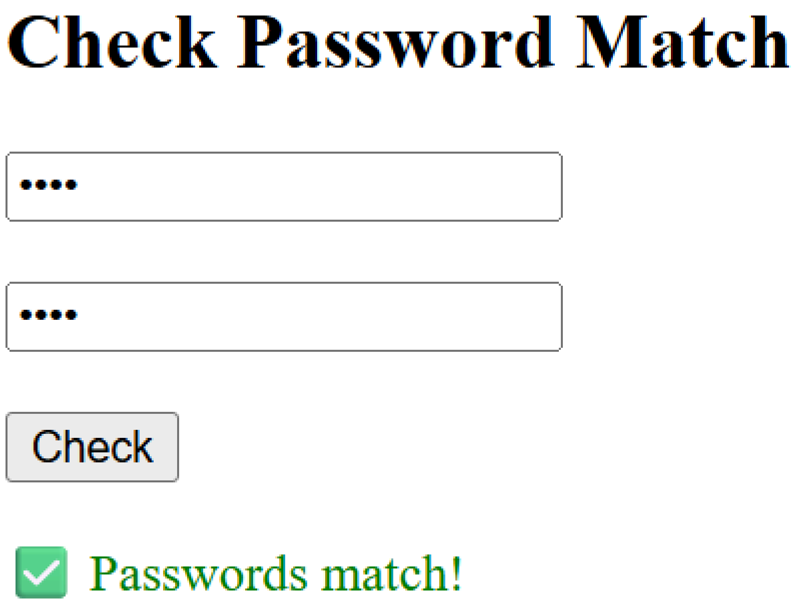
Select a Color

You have selected Red

---

- Write a JavaScript code to check password and confirm passwords are the same or not.

```
<!DOCTYPE html>
<html>
<head>
  <title>Password Match Check</title>
</head>
<body>
  <h2>Check Password Match</h2>

  <input type="password" id="password" placeholder="Enter Password"><br><br>
  <input type="password" id="confirmPassword" placeholder="Confirm Password"><br><br>
  <button onclick="checkPassword()">Check</button>

  <p id="message"></p>

  <script>
    function checkPassword() {
      let password = document.getElementById("password").value;
      let confirmPassword = document.getElementById("confirmPassword").value;

      if (password === confirmPassword) {
        document.getElementById("message").innerHTML = "✅ Passwords match!";
        document.getElementById("message").style.color = "green";
      } else {
        document.getElementById("message").innerHTML = "❌ Passwords do not match!";
        document.getElementById("message").style.color = "red";
      }
    }
  </script>
</body>
</html>
```



Check Password Match

✅ Passwords match!