Resumo Consolidado: Aulas 5 e 6

Os exemplos contidos neste resumo estão disponíveis no seguinte repositório do Github:
https://github.com/hdblouro/DWI_Aulas05e06

Aula 5: Media Queries (Consultas de Mídia)

O principal objetivo das **Media Queries** (Consultas de Mídia) é permitir o Desenvolvimento Web Responsivo, criando layouts flexíveis que se ajustam dinamicamente às características do dispositivo ou à largura da **viewport** (área visível da página). Isso garante uma experiência de usuário consistente e agradável, seja em desktops ou dispositivos móveis.

Elementos Chave

Tipos de Mídia (Media Types): Definem onde o estilo será aplicado.

Os tipos incluem:

- **screen**: Para exibição em telas.
- **print**: Aplica-se à visualização e impressão de documentos (permitindo remover elementos de navegação, por exemplo).
- **speech**: Aplica-se a leitores de tela.

Características de Mídia (Media Features): Especificam condições de dispositivo ou viewport.

As mais utilizadas são:

- **min-width** e **max-width**: Definem faixas de largura. Por exemplo, estilos podem ser aplicados se a largura for de até 700px (max-width: 700px).
- **orientation**: Determina a orientação da tela (**landscape** para paisagem ou **portrait** para retrato).

Sintaxe e Lógica: As regras CSS dentro de uma **media query** somente são aplicadas se a condição for verdadeira.

É possível combinar características usando operadores lógicos:

- **and**: Combina condições, exigindo que ambas sejam verdadeiras (ex: **min-width**: 600px **and** **max-width**: 700px).
- **not**: Nega uma condição.

Hierarquia de Estilos: Quando uma **media query** se torna verdadeira, as regras CSS definidas dentro dela substituem as regras de estilo definidas fora, seguindo a hierarquia normal de estilos.

Meta Viewport: É crucial incluir a **metatag viewport** no **<head>** do documento para que o navegador reconheça e utilize corretamente as **media queries** no dimensionamento da página, especialmente em dispositivos móveis. Um exemplo comum é **<meta name="viewport" content="width=device-width, initial-scale=1" />.**

Media Queries (Exemplos Práticos)

Exemplo 1: Esconder Elementos e Ajustar Fonte para Impressão

Este exemplo utiliza a **media query** print para otimizar a página para impressão, escondendo elementos não essenciais (como um menu lateral e um rodapé) e ajustando o tamanho da fonte.

Abaixo está um exemplo de código HTML que incorpora esses elementos, utilizando as tags semânticas e estruturais, como <!DOCTYPE html>, <html>, <head>, <body>, , <h1>, <p>, e <footer>.

Este código deve ser salvo, por exemplo, como Exemplo1.html.

```
html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Meta tag de viewport essencial --
>
  <title>Página Otimizada para Impressão</title>

  <!-- Link para o arquivo CSS externo (onde você aplicaria o Exemplo 1) -->
  <!-- Assumindo que o CSS esteja em 'estilos.css' na mesma pasta -->
  <link rel="stylesheet" href="estilosexemplo1.css">

  <!-- Estilos internos (Apenas para demonstração e melhor legibilidade no HTML) -->
  <style>
    / Estilos básicos para tela (serão sobrescritos pela media query 'print' no CSS externo) /
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    .menu-navegacao {
      background-color: #f8f8f8;
      padding: 15px;
      margin-bottom: 20px;
      border-bottom: 1px solid #ddd;
    }
    .menu-navegacao ul {
      list-style: none;
      padding: 0;
      margin: 0;
    }
    .menu-navegacao li {
      display: inline; / Elemento inline /
      margin-right: 15px;
    }
    footer {
      margin-top: 50px;
      padding: 10px;
      text-align: center;
      border-top: 1px solid #ddd;
```

```
        font-size: 0.9em;
        background-color: #f8f8f8;
      }
      .artigo-principal {
        border: 1px dashed #ccc; / Adiciona uma borda para visualização do artigo /
        padding: 20px;
      }
    </style>
</head>
<body>

    <!-- 1. Menu de Navegação (Elemento a ser escondido na impressão) -->
    <div class="menu-navegacao">
      <ul>
        <li><a href="#">Início</a></li>
        <li><a href="#">Sobre</a></li>
        <li><a href="#">Contato</a></li>
        <li><a href="#">Compartilhar</a></li>
      </ul>
    </div>

    <!-- 2. Conteúdo Principal do Artigo (Elemento principal que terá o estilo de fonte ajustado) -->
    <div class="artigo-principal">
      <h1>Título do Artigo: A Importância do Desenvolvimento Web</h1>

      <h2>Introdução (Subtítulo h2)</h2>
      <p>
        O desenvolvimento web (HTML, CSS e JavaScript) é fundamental para a criação de sistemas corporativos e aplicações. O HTML, como linguagem de marcação, é responsável pela estrutura semântica e o conteúdo.
      </p>

      <h2>Estrutura e Estilo</h2>
      <p>
        Elementos de bloco como o <code>&lt;p&gt;</code> e <code>&lt;div&gt;</code> ocupam toda a largura disponível. Já o CSS, definido em um arquivo externo (boa prática), lida com a apresentação e permite ajustes como o espaçamento e cores.
      </p>
    </div>

    <!-- 3. Rodapé (Elemento a ser escondido na impressão) -->
    <footer>
      <p>
        &copy; 2025 Desenvolvimento Web I. Todos os direitos reservados.
        <a href="#">Política de Privacidade</a>. <!-- Exemplo de link -->
      </p>
      <p>
        Contato: (12) 99999-9999
      </p>
    </footer>
```

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

Para que o HTML acima funcione com o HTML de exemplo você precisará ter o seguinte CSS no seu arquivo estilosexemplo1.css:

```
/ Estilos Padrão /
body {
  font-size: 16px;
}
.menu-navegacao, footer, .artigo-principal {
  / (Se houverem estilos de tela, eles estariam aqui) /
}

/ Regras aplicadas SOMENTE na impressão (Media Query print) /
@media print {
  / Esconde o menu de navegação e o rodapé /
  .menu-navegacao, footer {
    display: none;
  }

  / Ajusta o tamanho da fonte do corpo para melhor legibilidade no papel /
  body {
    font-size: 12pt; / Usando pt (pontos) para impressão é comum /
    color: black;
  }

  / Remove a borda do artigo (como no exemplo anterior) /
  .artigo-principal {
    border: none;
    padding: 0;
  }
}
```

Após criado os dois arquivos anteriores, execute o HTML em um navegador de sua preferência. Vá no menu lateral e escolha imprimir. Observe que o que será impresso difere um pouco da página HTML, pois teve alguns elementos suprimidos para melhorar a impressão.

Exemplo 2: Animação Condicional por Orientação de Tela

Este exemplo usa a característica de **mídia orientation** para aplicar uma rotação sutil a uma imagem apenas quando a tela do dispositivo estiver em modo **landscape** (paisagem).

Para que essa regra CSS possa ser aplicada, precisamos de um documento HTML que contenha a estrutura básica necessária, a **metatag viewport** (crucial para o design responsivo e media queries), e um elemento que será o alvo da estilização, neste caso, uma imagem (<img>) com a classe CSS que definimos, .imagem-perfil.

Este código deve ser salvo, por exemplo, como Exemplo2.html.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="utf-8" />
  <!--
    A metatag viewport é essencial para garantir que o navegador
    interprete corretamente as Media Queries em dispositivos móveis.
  -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Animação Condicional por Orientação</title>

  <!--
    Link para o arquivo CSS externo, onde as regras do Exemplo 2 serão aplicadas.
    Assumimos que o CSS esteja em 'estilosexemplo2.css' ou 'media-queries.css'.
  -->
  <link rel="stylesheet" href="estilosexemplo2.css" type="text/css">

  <!--
    Incluímos os estilos do Exemplo 2 aqui (como um estilo interno)
    apenas para facilitar a demonstração neste documento único.
  -->
  <style>
    / Estilo padrão da imagem, para telas em qualquer orientação /
    .imagem-perfil {
      / Definimos um tamanho para o Box Model /
      width: 250px;
      height: 250px;

      / Suaviza a transição de qualquer propriedade que mude (como o transform) /
      transition: transform 0.5s ease;

      / Torna a imagem circular, usando CSS3 /
      border-radius: 50%;

      / Garante que a imagem seja um elemento de bloco para respeitar as dimensões /
      display: block;
      margin: 50px auto; / Centraliza horizontalmente /
      object-fit: cover; / Garante que a imagem preencha o círculo /
    }
  </style>
</head>
<body>

  <!--
    O elemento <img> é o alvo. Ele usa os atributos 'src' (URL da imagem)
```

```
    e 'alt' (texto alternativo).
  -->
  <img
    src="caminho/para/sua/foto-perfil.jpg"
    alt="Foto de Perfil do Usuário"
    class="imagem-perfil"
  />

  <div style="text-align: center; margin-top: 20px;">
    <p>Redimensione a janela do navegador ou gire seu dispositivo para ver o efeito.</p>
    <p>A imagem só terá rotação e sombra quando a largura for maior que a altura (modo paisagem).</p>
  </div>

</body>
</html>
```

Crie o arquivo estilosexemplo2.css

```
/
  Regra CSS do Exemplo 2: Aplica rotação e sombra SOMENTE quando
  a orientação da tela for "landscape" (paisagem)
/
@media screen and (orientation: landscape) {
  .imagem-perfil {
    / Transformação CSS3: Gira 3 graus /
    transform: rotate(3deg);
    / Adiciona sombra (box-shadow CSS3) /
    box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.5);
  }
}

/
  Regra complementar: Reseta o Box Shadow no modo portrait
  (embora a transição de 0.5s cuide da volta, é bom ser explícito)
/
@media screen and (orientation: portrait) {
  .imagem-perfil {
    transform: rotate(0deg);
    box-shadow: none;
  }
}
```

O código HTML utiliza o elemento <img>, que requer o atributo src (caminho da imagem) e o atributo alt (texto alternativo). A classe .imagem-perfil é a chave para a estilização condicional via CSS.

No bloco <style>, utilizamos a @media screen and (orientation: landscape) para aplicar a propriedade transform: rotate(3deg) e box-shadow somente quando o dispositivo estiver na orientação paisagem, garantindo a Animação Condicional por Orientação de Tela.

Exemplo 3: Barra Lateral Dinâmica com Breakpoints Múltiplos

Este exemplo demonstra o uso de **min-width** com operadores **and** para aplicar um layout de múltiplas colunas em telas médias e grandes, mas voltando a um layout de coluna única em telas pequenas.

Como o HTML por si só é responsável pela estrutura e não pelos estilos responsivos, criaremos uma estrutura HTML básica contendo:

1. A estrutura fundamental (<!DOCTYPE html>, <head>, <body>).
2. A metatag viewport, essencial para o design responsivo e para que as media queries funcionem corretamente.
3. Dois elementos de bloco, como , que representarão o Conteúdo Principal e a Barra Lateral, e que terão suas larguras e posicionamentos manipulados pelo CSS em diferentes tamanhos de tela.

Código HTML: layout_responsivo.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <!-- Meta tag viewport crucial para o design responsivo -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Layout com Múltiplas Colunas Responsivas</title>

    <!-- Link para o arquivo CSS externo onde o Exemplo 3 estará definido -->
    <link rel="stylesheet" href="estilosexemplo3.css">

    <!--
    Incluímos a media query do Exemplo 3 aqui (como estilo interno)
    para que o código seja autocontido e demonstre o funcionamento.
  -->
    <style>
    /* Estilo Básico (Mobile-First: Coluna única) */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 20px;
    }
    * {
      box-sizing: border-box; /* Garante que padding e border NÃO aumentam o tamanho total do elemento */
    }
    .conteudo-principal {
      /* Padrão para telas pequenas: Ocupa a largura total (Block Box) */
      background-color: #f0f0f0;
      padding: 15px;
      margin-bottom: 10px;
      border: 1px solid #ccc;
    }
    .barra-lateral {
      /* Padrão para telas pequenas: Abaixo do conteúdo principal */
```

```
      background-color: #cceeff;
      padding: 15px;
      border: 1px solid #aaddff;
    }
    /* Técnica para limpar floats, essencial para layouts de colunas */
    .clearfix::after {
      content: "";
      display: table;
      clear: both;
    }
  </style>
  </head>
  <body>
    <!-- Contêiner principal para o layout, usando clearfix para evitar problemas de float -->
    <div class="clearfix" style="max-width: 1400px; margin: 20px auto;">

      <!-- 1. Conteúdo Principal -->
      <div class="conteudo-principal">
        <h1>Conteúdo Principal do Artigo</h1>
        <p>
          Este é o bloco de maior importância na página. Em telas
          pequenas, ele ocupa toda a largura.
          Em telas médias, ele ocupa 65% da largura, permitindo que a
          barra lateral fique ao lado.
        </p>
        <p>
          A estilização de elementos como parágrafos (&lt;p&gt;) e
          divisões
          ("&lt;div&gt;") é feita com CSS.
          O '&lt;div&gt;' é um contêiner genérico de bloco.
        </p>
      </div>

      <!-- 2. Barra Lateral -->
      <div class="barra-lateral">
        <h2>Navegação Lateral</h2>
        <!-- Exemplo de lista não ordenada -->
        <ul>
          <li>Links Rápidos</li>
          <li>Anúncios ou Destaques</li>
          <li>Informações Adicionais</li>
        </ul>
      </div>
    </div>
    <!-- Rodapé simples para fechar a página -->
    <footer style="clear: both; text-align: center; margin-top: 30px; border-top: 1px solid #ccc;">
      <p>&copy; Layout Responsivo com Media Queries</p>
    </footer>
  </body>
```

```
</html>
```

estilosexemplo3.css

```
/* Estilo para Telas Médias (Tablets e Desktops menores: 768px a 1199px) */
@media (min-width: 768px) and (max-width: 1199px) {
  .conteudo-principal {
    float: left;
    /* Flutua à esquerda [10] */
    width: 65%;
  }

  .barra-lateral {
    float: right;
    /* Flutua à direita [10] */
    width: 30%;
  }
}

/* Estilo para Telas Grandes (Desktops: 1200px ou mais) */
@media (min-width: 1200px) {
  .conteudo-principal {
    float: left;
    width: 75%;
  }

  .barra-lateral {
    float: right;
    width: 20%;
  }
}
```

Para ver o Exemplo 3 em ação, basta salvar o código acima e abri-lo no navegador. Use F12 e depois a visão em telas menores para redimensionar a largura da janela do navegador, você observará três comportamentos distintos:

1. Largura pequena (abaixo de 768px): O conteúdo principal e a barra lateral se empilham, um sobre o outro (coluna única - comportamento mobile-first).
2. Largura média (entre 768px e 1199px): O conteúdo principal e a barra lateral flutuam lado a lado, com o conteúdo principal ocupando 65% da largura.
3. Largura grande (1200px ou mais): O conteúdo principal ocupa 75%, e a barra lateral, 20%, ajustando o layout para telas maiores.

Aula 6: Bootstrap

O Bootstrap é um framework **front-end** de código aberto, originalmente desenvolvido pela equipe do Twitter, que utiliza HTML, CSS e JavaScript para o desenvolvimento de componentes de interface. Ele acelera o desenvolvimento e incentiva a padronização de interfaces web.

Elementos Chaves

Integração (CDN): Para usar o **Bootstrap**, deve-se incluir o arquivo bootstrap.css (estilos) no <head> via <link> e o arquivo bootstrap.js (interatividade e responsividade) via <script>. É comum usar links CDN para esses arquivos, geralmente em suas versões **minificadas** (.min).
Aplicação de Estilos: Os estilos do framework são aplicados aos elementos HTML através de classes CSS. Elementos estilizados com essas classes são chamados de componentes (ex: **Alert** ou **Button**).
Sistema de Grades (Grid System): O layout responsivo é baseado em um sistema de grades de 12 colunas.
Pontos de Interrupção (Breakpoints): O Bootstrap define classes de coluna que se aplicam em larguras específicas (**breakpoints**), permitindo layouts que se ajustam a diferentes tamanhos de tela:

- Small (col-sm): ≥576px.
- Medium (col-md): ≥768px.
- Large (col-lg): ≥992px.

Se nenhuma classe de breakpoint for especificada, o elemento tenta dividir o espaço igualmente entre os irmãos.
Contêineres (Containers): Elementos base para o layout. A classe container define uma largura máxima que respeita os breakpoints, e container-fluid faz com que ocupe 100% da largura.
Utilidades (Utilities): O **Bootstrap** oferece classes utilitárias para tarefas comuns:

- **Espaçamento (Spacing)**: Classes de margem (m) e padding (p) combinadas com direção (t, b, s, e, x, y, ou vazio para todos os lados) e tamanho (0 a 5). Ex: mx-auto para centralizar horizontalmente.
- **Alinhamento**: Classes baseadas em **Flexbox**, aplicadas ao row (eixo principal) ou col (eixo transversal), como **justify-content-center** e **align-items-center**.
- **Dimensionamento (Sizing)**: Classes w- e h- para definir largura e altura, geralmente em porcentagens (e.g., w-100).

Bootstrap (Exemplos Práticos)

Para os exemplos a seguir, assumimos que os links CDN do Bootstrap já foram incluídos no documento HTML.

Exemplo 4: Configuração Responsiva de Botões (Grid)

Este exemplo usa o sistema de grid do Bootstrap para controlar a largura dos botões. Em telas extra pequenas (padrão), eles ocupam a largura total (col-12), mas em telas médias e grandes (col-md-4), três botões são exibidos lado a lado em uma única linha (row).

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo Bootstrap 4</title>
```

```
  <!-- Link para o CSS do Bootstrap 5 via CDN -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
</head>
<body>
  <div class="container mt-4">
    <div class="row">
      <!-- Ocupa 100% (12 colunas) em telas pequenas e 33% (4 colunas) em telas médias/grandes -->
      <div class="col-12 col-md-4 mb-2">
        <button class="btn btn-info w-100">Ver Perfil</button>
      </div>
      <div class="col-12 col-md-4 mb-2">
        <button class="btn btn-success w-100">Enviar Mensagem</button>
      </div>
      <div class="col-12 col-md-4 mb-2">
        <button class="btn btn-danger w-100">Bloquear</button>
      </div>
    </div>
  </div>

  <!-- Scripts do Bootstrap 5 (Popper.js e Bootstrap JS) via CDN -->
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js" integrity="sha384-
BBtl+eGJRgqQAUMxJ7pMwbEyER4l1g+O15P+16Ep7Q9Q+zqX6gSbd85u4mG4QzX+"
crossorigin="anonymous"></script>
</body>
</html>
```

Exemplo 5: Card de Status Centralizado e Estilizado (Spacing & Sizing)

Este exemplo usa classes utilitárias para: definir a largura do elemento (w-50), aplicar margem vertical (my-5), aplicar padding de 3 unidades (p-3), centralizá-lo horizontalmente (mx-auto) e adicionar cor de fundo e bordas.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo Bootstrap 5 - Status do Servidor</title>

  <!-- Link para o CSS do Bootstrap 5 via CDN -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
</head>
```

```
<body>
  <!-- w-50: Largura de 50%; mx-auto: Centraliza horizontalmente; my-5: Margem vertical alta -->
  <div class="w-50 mx-auto my-5 p-3 bg-light border border-primary rounded shadow">
    <h4 class="text-primary text-center">Status do Servidor</h4>
    <p class="text-center">O sistema está operando normalmente. Utilize a classe text-center para alinhar textos.</p>
    <div class="text-end">
      <!-- Alinha o botão à direita -->
      <button class="btn btn-outline-primary btn-sm">Detalhes</button>
    </div>
  </div>

  <!-- Scripts do Bootstrap 5 (Popper.js e Bootstrap JS) via CDN -->
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js" integrity="sha384-BBtl+eGJRgqQAUMxJ7pMwbEyER4l1g+O15P+16Ep7Q9Q+zqX6gSbd85u4mG4QzX+"
crossorigin="anonymous"></script>
</body>
</html>
```

Exemplo 6: Formulário Horizontal com Deslocamento (Offset)

Este exemplo demonstra como usar as classes de grid não apenas para dimensionar campos em telas maiores, mas também para usar offset para criar um deslocamento (espaço em branco) no layout, o que é útil para alinhamento horizontal.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo Bootstrap 6 - Formulário</title>

  <!-- Link para o CSS do Bootstrap 5 via CDN -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
</head>
<body>
  <div class="container mt-4">
    <form>
      <div class="row mb-3">
        <!-- Label ocupa 3 colunas em telas grandes (col-lg-3) -->
        <label for="campo1" class="col-lg-3 col-form-label">Nome de Usuário</label>

        <!-- Campo de entrada ocupa as 9 colunas restantes -->
        <div class="col-lg-9">
```

```
            <input type="text" class="form-control" id="campo1">
          </div>
        </div>

        <div class="row mb-3">
           <!-- Deslocamento de 3 colunas (offset-lg-3), forçando o campo a começar na 4ª coluna -->
          <div class="col-lg-9 offset-lg-3">
            <input type="checkbox" id="lembrar" class="form-check-input">
            <label for="lembrar" class="form-check-label">Lembrar-me</label>
          </div>
        </div>

        <div class="row">
          <!-- Botão de Envio (col-lg-2) também deslocado em 3 colunas -->
          <div class="col-lg-2 offset-lg-3">
            <button type="submit" class="btn btn-warning">Acessar</button>
          </div>
        </div>
      </form>
    </div>

    <!-- Scripts do Bootstrap 5 (Popper.js e Bootstrap JS) via CDN -->
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.min.js" integrity="sha384-BBtl+eGJRgqQAUMxJ7pMwbEyER4l1g+O15P+16Ep7Q9Q+zqX6gSbd85u4mG4QzX+"
crossorigin="anonymous"></script>
</body>
</html>
```