ipvc estg

CG1

22as JORNADAS da COMPUTAÇÃO GRÁFICA e MULTIMÉDIA

Workshop

Bem-vindo ao nosso workshop de HTML 5 + JavaScript!

Durante esta sessão, vamos explorar passo a passo a construção de um formulário de inscrição, utilizando a linguagem de marcação HTML, uma das bases essenciais para o desenvolvimento web. Para melhorar o aspecto do website, vamos utilizar a linguagem de estilos CSS.

Este documento funciona como um tutorial passo a passo que aborda desde a estrutura básica do nosso documento HTML até a implementação de funcionalidades interativas utilizando JavaScript.

Como parte prática deste workshop, utilizamos um exemplo de formulário de inscrição como modelo. Este formulário inclui campos para o nome, idade e escola do participante, além de uma funcionalidade de envio dos dados para uma base de dados hospedada na Supabase:

No final deste workshop, esperamos que te sintas confortável não apenas em construir formulários simples, mas também em personalizá-los e adicionar interatividade conforme as necessidades do teu projeto.

Sem mais demoras, vamos mergulhar de cabeça na criação do teu formulário de inscrição em HTML!

1. HTML

1.1 Começamos por escrever no nosso ficheiro "**index.html**" a seguinte parte de código.

```
<!DOCTYPE html>

<html lang="pt">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width,
initial-scale=1.0">

  <title>Document</title>

</head>

<body>



</body>

</html>
```

Dica: Se não quiseres ter trabalho a escrever tudo, basta escreveres "**!**" e clicares ENTER.

1.2. Mudamos o título do documento para "**Formulário de Inscrição**".

```
 <title>Formulário de Inscrição</title>
```

1.3. Dentro do "**<body>**", começamos a criar o formulário.

```
<form id="formulario" method="post">

    <label for="nome">Nome:</label><br>

    <input type="text" id="nome" name="nome" required><br><br>


    <label for="idade">Idade:</label><br>

    <input type="number" id="idade" name="idade" required><br><br>


    <label for="escola">Escola:</label><br>

    <input type="text" id="escola" name="escola" required><br><br>


    <input type="submit" id="botao" value="Enviar Inscrição">

  </form>
```

Formulário (**<form>**):

- Define um formulário que será submetido ao servidor quando o botão de envio for clicado.
- Possui um identificador único (**id="formulario"**) para referência em CSS ou JavaScript.
- Utiliza o método POST (**method="post"**) para enviar os dados do formulário.

Rótulos (**<label>**):

- São etiquetas descritivas que identificam os campos de entrada.
- Utilizam o atributo "**for**" para associar o rótulo ao campo de entrada correspondente.

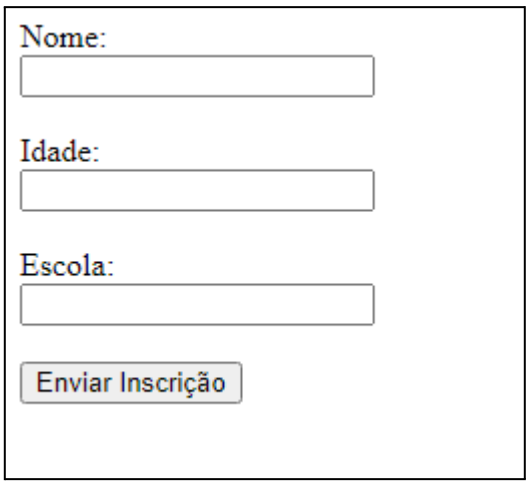
Campos de Entrada (**<input>**):

- São os elementos onde os usuários inserem dados.
- Cada campo possui um tipo específico, como texto (**type="text"**), número (**type="number"**), etc.
- Cada campo tem um identificador único (**id**) para referência e um nome (**name**) que é enviado ao servidor quando o formulário é submetido.
- O atributo "**required**" torna o preenchimento do campo obrigatório.

Botão de Envio (**<input type="submit">**):

- Cria um botão que, quando clicado, submete o formulário ao servidor.
- Pode ter um identificador único (**id="botao"**) para referência em CSS ou JavaScript.
- O texto dentro do botão (**value="Enviar Inscrição"**) é exibido como rótulo no botão.

Muitos parabéns, tens o formulário feito! Deves ter algo deste gênero:



Mas como é óbvio, não vamos deixar assim.

1.4. Primeiramente, temos de adicionar 2 novas linhas ao nosso ficheiro: uma para referir o ficheiro CSS e outra para referir o ficheiro JavaScript.

Podemos introduzi-las logo a seguir ao título do documento, mesmo antes de fecharmos o **<head>**.

```
<link rel="stylesheet" type="text/css" href="style.css">

<script src="script.js"></script>
```

Vamos também adicionar uma ** ,** que será usada mais tarde para melhorar o nosso site. (Uma ** ** é um elemento HTML usado para agrupar e organizar outros elementos num documento web).

```
 <div id="custom-cursor"></div>
```

Concluindo o HTML, o nosso código fica assim:

```
<!DOCTYPE html>

<html lang="pt">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<title>Formulário de Inscrição</title>

<link rel="stylesheet" type="text/css" href="style.css">

<script src="script.js"></script>

</head>



<body>

  <div id="custom-cursor"></div>

  <form id="formulario" method="post">

    <label for="nome">Nome:</label><br>

    <input type="text" id="nome" name="nome" required><br><br>

       <label for="idade">Idade:</label><br>

    <input type="number" id="idade" name="idade" required><br><br>

       <label for="escola">Escola:</label><br>

    <input type="text" id="escola" name="escola" required><br><br>

    <input type="submit" id="botao" value="Enviar Inscrição">

  </form>

</body>

</html>
```

2. CSS

2.1. Agora vamos utilizar estilos para melhorar a aparência do nosso website.

Começamos então pelo **body** e o **form**:

```
body {
    background-image: url('imgs/fundo.png'); /* Definir imagem de fundo */
    font-family: Arial, sans-serif; /* Fonte de texto */
    margin: 0; /* Para remover margens padrão */
    padding: 0; /* Para remover preenchimento padrão */
}
form {
    background-color: rgba(255, 255, 255, 0.8); /* Cor de fundo do formulário */
    padding: 20px; /* Espaçamento interno */
    border-radius: 10px; /* Borda arredondada */
    width: 400px; /* Largura do formulário */
    margin: 100px auto; /* Centralizar na página */
    margin-top: 350px;
}
```

2.2. Em seguida, passamos para os elementos do formulário através dos **ids** que atribuímos anteriormente na secção do **<input>** de cada elemento do **<form>**.

```
#idade, #nome, #escola {

    width: 100%; /* Expandir os campos de texto */

    padding: 10px; /* Espaçamento interno */

    margin-bottom: 10px; /* Espaçamento inferior */

    border-radius: 5px; /* Borda arredondada */

    border: 1px solid #ccc; /* Borda cinza */

    box-sizing: border-box; /* Incluir padding e border na largura */

}

#botao {

    width: 100%; /* Expandir o botão */

    padding: 10px; /* Espaçamento interno */

    border: none; /* Sem borda */

    border-radius: 5px; /* Borda arredondada */

    background-color: #007bff; /* Cor de fundo azul */

    color: #fff; /* Cor do texto */

}
```

2.3. Após isso, vamos substituir o cursor por uma imagem.

Começamos por acrescentar este bloco de código:

```
* {

    cursor: none;

}
```

O asterisco (*) é um seletor universal que se aplica a todos os elementos HTML.

Neste caso, o cursor passa a não existir em qualquer elemento da página.

O objetivo disto é simplesmente substituirmos o cursor padrão por uma imagem à nossa escolha, e vamos fazer isso utilizando outro **id** que referimos no nosso ficheiro HTML.

```
#custom-cursor {

    position: fixed;

    width: 35px; /* Largura da imagem do cursor */

    height: 30px; /* Altura da imagem do cursor */

    pointer-events: none; /* Faz com que o cursor não interfira com os elementos abaixo dele */

    background-image: url('imgs/logo.png'); /* Imagem escolhida */

    background-size: cover;

    z-index: 9999; /* Garante que o cursor fique na parte superior */

}
```

O cursor ainda não está funcional apenas com estas regras CSS, pois embora a aparência visual do cursor personalizado tenha sido definida, ainda não temos a lógica para atualizar dinamicamente a posição da imagem de acordo com a posição do cursor na tela.

2.4. Extra:

Para ficar mais bonito e responsivo, era engraçado que quando o cursor passasse por cima do botão de envio, este reagisse, ficando com uma cor mais escura.

```
#botao:hover {

    background-color: #0056b3; /* Cor de fundo azul mais escura no
hover */

}
```

Especificamente, ":hover" é um pseudo-seletor CSS que permite aplicar estilos quando o cursor está sobre um elemento.

Para já, deves ter isto:

Em resumo, o nosso ficheiro CSS fica com este aspeto:

```
body {
    background-image: url('imgs/fundo.png');
    font-family: Arial, sans-serif; /* Fonte de texto */
    margin: 0; /* Para remover margens padrão */
    padding: 0; /* Para remover preenchimento padrão */
}
form {
    background-color: rgba(255, 255, 255, 0.8); /* Cor de fundo do
formulário */
    padding: 20px; /* Espaçamento interno */
    border-radius: 10px; /* Borda arredondada */
    width: 400px; /* Largura do formulário */
    margin: 100px auto; /* Centralizar na página */
    margin-top: 350px;
}
#idade, #nome, #escola {
    width: 100%; /* Expandir os campos de texto */
    padding: 10px; /* Espaçamento interno */
    margin-bottom: 10px; /* Espaçamento inferior */
    border-radius: 5px; /* Borda arredondada */
    border: 1px solid #ccc; /* Borda cinza */
    box-sizing: border-box; /* Incluir padding e border na largura */
}
```

```css
#botao {
    width: 100%; /* Expandir o botão */
    padding: 10px; /* Espaçamento interno */
    border: none; /* Sem borda */
    border-radius: 5px; /* Borda arredondada */
    background-color: #007bff; /* Cor de fundo azul */
    color: #fff; /* Cor do texto */
}

* {
    cursor: none;
}
#custom-cursor {
    position: fixed;
    width: 35px; /* Largura da imagem do cursor */
    height: 30px; /* Altura da imagem do cursor */
    pointer-events: none; /* Faz com que o cursor não interfira com os elementos abaixo dele */
    background-image: url('imgs/logo.png'); /* Imagem escolhida */
    background-size: cover;
    z-index: 9999; /* Garante que o cursor fique na parte superior */
}
```

Vamos passar agora para a parte mais complicada…

3. JavaScript

3.1. Primeiramente, vamos tratar do assunto do cursor.

```
// Espera que o DOM (Document Object Model) esteja completamente
carregado antes de executar o código

document.addEventListener('DOMContentLoaded', function () {

    // Seleciona o elemento do cursor personalizado

    const cursor = document.getElementById('custom-cursor');

    // Adiciona um listener para o evento de movimento do mouse

    document.addEventListener('mousemove', function (e) {

      // Obtém as coordenadas X e Y do cursor do mouse

      const x = e.clientX;

      const y = e.clientY;

      // Atualiza a posição do cursor personalizado de acordo com as
coordenadas do mouse

      cursor.style.left = x + 'px';

      cursor.style.top = y + 'px';

    })

      // Deixa um espaço aqui, vai ser preciso mais à frente


  });
```

3.2. Abaixo desse código, vamos estabelecer uma ligação a uma base de dados.

Uma base de dados é um sistema organizado de armazenamento de informações. Essas informações podem incluir dados sobre pessoas, produtos, transações ou qualquer outra coisa que queiramos armazenar e consultar. As bases de dados são usadas em muitas aplicações, desde sites e aplicativos móveis até sistemas empresariais, para armazenar e recuperar dados de maneira eficiente.

Para começarmos esta ligação, primeiro precisamos de 2 variáveis fundamentais: o URL da base de dados e a API Key da mesma. Mas o que representam estas variáveis?

- Um URL de uma base de dados é como um endereço que aponta para onde os dados estão armazenados na internet. É como uma morada de uma casa, que indica onde os dados podem ser encontrados.
- Já uma API Key é como uma chave de acesso. É um código secreto que permite que um programa aceda e interaja com os dados na base de dados de forma segura. É como uma password que concede permissão para utilizar os recursos da base de dados de acordo com as regras estabelecidas.

```
// URL e chave de API do Supabase

const supabaseURL = "https://ycyquwkqvjrwxkkfphdy.supabase.co";

const supabaseAPIKey =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6
InljeXF1d2txdmpyd3hra2ZwaGR5Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3MTE1NzY3OD
UsImV4cCI6MjAyNzE1Mjc4NX0.URB1EljCsRybUkZdvoebl9q1lTAE7hfJiUwpGP7PtLA"
;
```

Tenham especial cuidado ao copiar estes dados, pois se algum deles estiver errado, a ligação com a base de dados não vai ser possível.

Agora, vamos criar uma função assíncrona que vai ser responsável por enviar os dados.

```
// Função para enviar os dados do formulário para o Supabase

async function enviarDados(event) {

    event.preventDefault(); // Evita que o formulário seja enviado
normalmente



    // Obtém os dados do formulário

    const formData = new
FormData(document.getElementById('formulario'));

    const nome = formData.get('nome');

    const idade = formData.get('idade');

    const escola = formData.get('escola');

      // O resto do código vai ser colado aqui

}
```

Aqui está o que cada parte do código faz:

- ***“ async function enviarDados(event) { “*** : Esta linha define uma função chamada **enviarDados** que aceita um argumento **event**. A palavra-chave **async** indica que esta função é assíncrona, o que significa que ela pode trabalhar com operações que podem demorar, como solicitações de rede.
- ***“ event.preventDefault(); “*** : Esta linha impede que o formulário seja enviado normalmente, o que é o comportamento padrão de um formulário HTML quando o botão de envio é clicado. Em vez disso, o código irá lidar com o envio dos dados.
- ***“ const formData = new FormData(document.getElementById('formulario')); “*** : Aqui, estamos a criar um objeto **FormData** que contém os dados do formulário com o **id** "**formulario**". Isso permite ter acesso aos valores dos campos do formulário facilmente.
- ***“ const nome = formData.get('nome'); const idade = formData.get('idade'); const escola = formData.get('escola'); “*** : Estas linhas extraem os valores dos campos do formulário (nome, idade e escola) usando o método **get()** do objeto **FormData**. Esses valores são armazenados em variáveis para serem usados posteriormente.

Vamos completar o envio dos dados, estabelecendo a ligação à base de dados.

```
try {
        // Envia os dados para o Supabase usando a API REST
        const response = await
fetch(`${supabaseURL}/rest/v1/Utilizador?apikey=${supabaseAPIKey}`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ nome, idade, escola })
        });
        // Verifica se a resposta foi bem-sucedida
        if (response.ok) {
            alert("Inscrição concluída!!!");
            // Limpa o formulário após o envio bem-sucedido
            document.getElementById('formulario').reset();
        } else {
            throw new Error('Erro ao enviar os dados para o
Supabase');
        }
    } catch (error) {
        console.error("Erro ao enviar dados para a Supabase:", error);
        alert("Ocorreu um erro ao enviar os dados. Por favor, tente
novamente mais tarde.");
    }
```

Esta parte é um pouco confusa, mas vamos tentar explicar:

- Neste código, estamos a tentar enviar os dados do formulário para a nossa base de dados. Primeiro, usamos a função "**fetch()**" para fazer uma requisição **POST** para o URL da API do Supabase, incluindo a chave de API para autenticação.
- Os dados que queremos enviar (nome, idade, escola) são formatados como um objeto JSON e enviados no corpo da requisição.
- Em seguida, verificamos se a resposta da requisição foi bem-sucedida. Se sim, exibimos uma mensagem de alerta informando que a inscrição foi concluída e limpamos o formulário. Se a resposta não foi bem-sucedida, lançamos um erro e exibimos uma mensagem de alerta informando que ocorreu um erro ao enviar os dados.
- Se houver algum erro durante o processo, ele será capturado pelo bloco "**catch**" e uma mensagem de erro será exibida na consola.

Para quem não sabe o que é um ficheiro JSON, este é amplamente utilizado em desenvolvimento web devido à sua simplicidade, eficiência e facilidade de leitura e escrita tanto por humanos quanto por máquinas. É frequentemente utilizado para enviar dados entre o frontend e o backend de uma aplicação web, seja em requisições HTTP ou em respostas de API.

Por fim, basta apenas uma linha de código responsável por executar o método "**enviarDados()**".

Esta linha vai ser colocada na primeira parte que escrevemos anteriormente, logo abaixo da parte do cursor (onde tinha o texto "*O resto do código vai ser colado aqui* "):

```
// Espera que o DOM esteja completamente carregado antes de executar o código
document.addEventListener('DOMContentLoaded', function () {

    // Seleciona o elemento do cursor personalizado
    const cursor = document.getElementById('custom-cursor');
    // Adiciona um listener para o evento de movimento do mouse
    document.addEventListener('mousemove', function (e) {
      // Obtém as coordenadas X e Y do cursor do mouse
      const x = e.clientX;
      const y = e.clientY;

      // Atualiza a posição do cursor personalizado de acordo com as coordenadas do mouse
      cursor.style.left = x + 'px';
      cursor.style.top = y + 'px';
    })
    // Adiciona um listener para o evento de envio do formulário
    document.getElementById('formulario').addEventListener('submit', enviarDados);
});
```

E feito! Temos um formulário completamente funcional que é capaz de enviar informações para uma base de dados.

Esperamos que tenham gostado e que tenham aprendido alguma coisa sobre estas tecnologias bastante usadas no desenvolvimento Web.

Obrigado!

Referências

1. https://pt.wikipedia.org/wiki/HTML
2. https://www.w3schools.com/html/
3. https://pt.wikipedia.org/wiki/Cascading_Style_Sheets
4. https://www.w3schools.com/css/
5. https://pt.wikipedia.org/wiki/JavaScript
6. https://www.w3schools.com/js/
7. https://supabase.com/