CRIANDO CAMPOS DE E-MAIL E TELEFONE EM FORMULÁRIOS HTML5

1. NOVOS CONTROLES DE FORMULÁRIO ESSENCIAIS

O HTML5 introduziu uma série de novos tipos de `input` projetados para simplificar a coleta de informações comuns e aprimorar a experiência do usuário em formulários web. Entre os mais importantes estão os controles específicos para e-mail e telefone, dados frequentemente solicitados em qualquer cadastro online.

Coletar e-mails e números de telefone de forma correta é crucial para a comunicação e o engajamento com os usuários. Utilizar os tipos de `input` adequados — `type="email"` e `type="tel"` — não apenas facilita o preenchimento em dispositivos móveis, mas também introduz uma camada inicial de validação diretamente no navegador. Esta abordagem melhora a qualidade dos dados recebidos e previne erros comuns de digitação antes mesmo que o formulário seja enviado. A seguir, exploraremos como implementar o primeiro desses controles: o campo de e-mail.

1.1. 2. O Campo de E-mail: `<input type="email">`

O `input type="email"` é um controle estratégico para qualquer formulário que necessite do endereço de e-mail do usuário. Sua principal vantagem é a validação nativa oferecida pelos navegadores, que verificam automaticamente se o valor inserido contém o caractere "@". Embora simples, essa verificação funciona como uma primeira linha de defesa contra dados mal formatados. É importante notar, no entanto, que esta é uma verificação estrutural, não uma confirmação da existência ou validade do e-mail. Como alertado na aula, é um primeiro passo útil, mas não se deve esperar uma validação completa apenas deste atributo. Para criar um campo de e-mail básico e acessível, utilizamos o seguinte código HTML:

```
<p>
    <label for="id-email">E-mail:</label>
            <input   type="email"   name="email"   id="id-email"   required
autocomplete="email">
</p>
```

Análise dos Atributos:

- **`type="email"`**: Define o campo como um controle de e-mail, ativando a validação básica do navegador e, em dispositivos móveis, otimizando o teclado virtual para incluir caracteres como "@" e ".".
- **`required`**: Torna o preenchimento do campo obrigatório. O navegador exibirá uma mensagem de erro se o usuário tentar enviar o formulário com este campo em branco.
- **`autocomplete="email"`**: Melhora a usabilidade ao permitir que o navegador sugira e-mails que o usuário já utilizou em outros formulários, agilizando o preenchimento.

Assim como existe um tipo de `input` específico para e-mails, o HTML5 também fornece um controle otimizado para a coleta de números de telefone.

1.2. O Campo de Telefone: `<input type="tel">`

O `<input type="tel">` foi projetado para a inserção de números de telefone. Diferente do campo de e-mail, ele não possui uma validação de formato incorporada. Isso ocorre porque os formatos de número de telefone variam drasticamente entre países e regiões. No entanto, seu grande benefício é a otimização para dispositivos móveis: ao focar neste campo, o sistema operacional tende a exibir um teclado numérico, facilitando a digitação e podendo se integrar à lista de contatos.

Veja como implementar um campo de telefone básico, utilizando o atributo `placeholder` para orientar o usuário:

```
<p>
    <label for="id-tel">Telefone:</label>
      <input type="tel" name="tel" id="id-tel" autocomplete="tel" required
placeholder="(xx) xxxxx-xxxx">
</p>
```

A principal limitação do `type="tel"` é, portanto, a ausência de uma validação de formato. Para mitigar isso, o atributo `placeholder` se torna uma ferramenta valiosa, pois exibe um exemplo visual do formato esperado. Além disso, o atributo `autocomplete` pode ser mais específico para telefones, com valores como `tel-national` para um número nacional, dando dicas mais precisas ao navegador. Para impor uma validação de formato mais rigorosa, precisamos de uma solução mais avançada: o uso de Expressões Regulares com o atributo `pattern`.

1.3. Validação Avançada com o Atributo `pattern` e Expressões Regulares (RegEx)

Para garantir que os dados inseridos sigam um formato específico, o HTML5 oferece o atributo `pattern`. Ele permite que o desenvolvedor defina uma Expressão Regular (RegEx) — uma sequência de caracteres que define um padrão de busca — para validar o conteúdo de um campo antes do envio do formulário. Este é um recurso poderoso para garantir a consistência dos dados do lado do cliente, melhorando a experiência do usuário ao fornecer feedback imediato.

Nota: É importante estudar mais sobre **RegEx (Regular Expressions)** para dominar a criação de padrões de validação personalizados.

A seguir, veremos alguns exemplos de como aplicar o `pattern` para validar diferentes formatos de telefone.

1.3.1. Padrão Fixo (Ex: `xxxx-xxxx`)

Este padrão exige um formato fixo com quatro dígitos, um hífen e mais quatro dígitos.

```
<input type="tel" name="tel" id="id-tel" required pattern="[0-9]{4}-[0-9]
{4}">
```

- **Análise da RegEx:**
    - `[0-9]`: É um **conjunto de caracteres (character set)** que corresponde a qualquer dígito de 0 a 9. Um atalho comum para `[0-9]` é `\d`.
    - `{4}`: É um **quantificador** que especifica que o conjunto de caracteres anterior deve ocorrer exatamente quatro vezes.
    - `-`: É o caractere literal de hífen.

4.2. Padrão Flexível (Ex: Celular com 8 ou 9 dígitos)

Para acomodar tanto números de telefone fixo (8 dígitos) quanto de celular (9 dígitos), podemos tornar o primeiro grupo de dígitos flexível.

```
<input type="tel" name="tel" id="id-tel" required pattern="[0-9]{4,5}-[0-9]
{4}">
```

- **Análise da RegEx:**
    - `{4,5}`: É um **quantificador** que define um intervalo, especificando que o padrão anterior (`[0-9]`) pode ocorrer no mínimo 4 vezes e no máximo 5 vezes.

4.3. Padrão Completo com DDD

Para um formato mais completo, incluindo o DDD entre parênteses, a expressão regular se torna um pouco mais complexa.

```
<input type="tel" name="tel" id="id-tel" required pattern="\([0-9]{2}\) [0-
9]{4,5}-[0-9]{4}">
```

- **Análise da RegEx:**
    - `\(` e `\)`: Os parênteses são caracteres especiais em RegEx. Para que eles sejam interpretados como os caracteres literais "(" e ")", precisamos "escapá-los" com uma barra invertida (`\`).
    - `[0-9]{2}`: Exige exatamente dois dígitos para o DDD.

Aviso Crucial: Validação no Cliente é Apenas para Usabilidade. A validação via HTML, como o atributo `pattern`, é uma excelente ferramenta para melhorar a experiência do usuário, fornecendo feedback imediato no navegador. No entanto, ela **nunca deve ser considerada uma medida de segurança**. Um usuário mal-intencionado pode facilmente contornar qualquer validação no lado do cliente. A validação definitiva, que garante a integridade e a segurança dos dados, **deve**

obrigatoriamente ser reimplementada no lado do servidor (com tecnologias como PHP, Python, Java, etc.) antes de qualquer processamento ou armazenamento. Agora que sabemos como criar e validar os campos, vamos aprender a organizá-los visualmente no formulário.

1.4. Agrupando Campos com `<fieldset>` e `<legend>`

Para melhorar a organização visual e semântica de um formulário, especialmente os mais longos, é uma boa prática agrupar campos relacionados. O HTML fornece as tags `<fieldset>` e `<legend>` para este propósito. O `<fieldset>` cria um agrupamento visual (geralmente com uma borda) em torno dos campos, enquanto o `<legend>` fornece um título ou legenda descritiva para esse grupo. Essa estrutura não só melhora a clareza para o usuário, mas também a acessibilidade para leitores de tela. O código abaixo demonstra um formulário completo onde os campos de nome, e-mail e telefone são agrupados sob a legenda "Dados Pessoais".

```
<form action="cadastro.php" method="get" autocomplete="on">
    <fieldset>
        <legend>Dados Pessoais</legend>
        <p>
            <label for="id-nome">Nome:</label>
                    <input  type="text"  name="nome"  id="id-nome"  required
minlength="5" maxlength="20">
        </p>
        <p>
            <label for="id-email">E-mail:</label>
                  <input  type="email"  name="email"  id="id-email"  required
autocomplete="email">
        </p>
        <p>
            <label for="id-tel">Telefone:</label>
               <input type="tel" name="tel" id="id-tel" autocomplete="tel"
required   pattern="\([0-9]{2}\)   [0-9]{4,5}-[0-9]{4}"   placeholder="(xx)
xxxxx-xxxx">
        </p>
    </fieldset>

    <p>
        <input type="submit" value="Enviar">
        <input type="reset" value="Limpar">
    </p>
</form>
```

O resultado é um formulário mais limpo e intuitivo, onde o `<fieldset>` desenha uma borda em torno dos campos de dados pessoais e o `<legend>` exibe o título "Dados Pessoais". Note que o exemplo também inclui um campo de texto padrão para o nome, utilizando os atributos `minlength` e `maxlength` para limitar o tamanho da entrada, demonstrando como diferentes tipos de input podem ser logicamente agrupados.

2. 6. CONCLUSÃO E PRÓXIMOS PASSOS

Neste guia, aprendemos a implementar os controles de formulário `input type="email"` e `input type="tel"`, essenciais para a coleta de dados de contato. Vimos como o `type="email"` oferece validação estrutural e como podemos usar o atributo `pattern`

com Expressões Regulares (RegEx) para impor formatos específicos no campo de telefone. Além disso, exploramos como as tags `<fieldset>` e `<legend>` podem ser usadas para agrupar campos e melhorar a estrutura e a acessibilidade do formulário.

A recomendação final é a prática constante, especialmente com as Expressões Regulares. Sua sintaxe é extremamente sensível: como alertado na aula, um único caractere fora do lugar, um espaço extra ou um erro de digitação fará com que a validação simplesmente pare de funcionar.

Os estudos sobre formulários HTML5 continuam. Nas próximas etapas, exploraremos outros controles importantes, como botões de rádio (`radio`) e caixas de seleção (`checkbox`), expandindo ainda mais seu conhecimento para criar formulários web robustos e eficazes.