Criando Campos de Texto e Senha em Formulários HTML

1. Aprimorando Formulários com Controles Essenciais

Esta apostila tem como objetivo aprimorar a construção de formulários HTML, evoluindo de estruturas básicas para a criação de campos de usuário e senha mais seguros, validados e fáceis de usar. Abordaremos problemas comuns, como o envio de dados em branco, que podem comprometer a integridade de uma aplicação.

Para isso, apresentaremos novos tipos de `input` e atributos fundamentais do HTML5 que permitem criar formulários mais robustos e inteligentes. Com essas ferramentas, o próprio navegador pode realizar validações prévias, garantindo que os dados enviados ao servidor atendam às regras de negócio estabelecidas. A seguir, veremos a estrutura base do código que utilizaremos como ponto de partida.

2. Estrutura Base do Formulário de Login

Para começar, é fundamental estabelecer uma estrutura HTML bem definida. Utilizaremos tags semânticas como `<form>` para encapsular nosso formulário e `<label>` para associar descrições textuais aos seus respectivos campos de entrada, melhorando tanto a acessibilidade quanto a usabilidade.

Para a organização dos campos, utilizaremos a tag de parágrafo (`<p>`). É importante notar que esta é uma escolha prática, especialmente para iniciantes, pois o parágrafo adiciona um espaçamento vertical padrão entre os elementos. Contudo, a estruturação também poderia ser feita com `<div>`, uma alternativa muito comum no desenvolvimento web.

O código inicial abaixo representa um formulário de login simples, com campos para "Usuário" e "Senha", além dos botões de "Enviar" e "Limpar". Análise dos componentes chave do código apresentado:

- **`<form action="cadastro.php" method="post">`**: O atributo `action` define para qual arquivo (`cadastro.php`) os dados serão enviados. Já o `method="post"` é o método de envio escolhido por ser mais seguro para o envio de dados sensíveis, como senhas.

- **`<label for="...">`**: A tag `label` cria uma etiqueta para o campo de formulário. O atributo `for` a associa a um `input` específico através do `id` correspondente, permitindo que o usuário clique no texto para focar no campo.
- **`<input type="..." name="..." id="...">`**: Esta é a tag principal para controles de formulário.
    - `type`: Define o tipo de controle (ex: texto, senha).
    - `name`: Atribui um nome ao dado, que é crucial para que a tecnologia do backend (como o PHP) possa identificá-lo e processá-lo.
    - `id`: Cria um identificador único para o elemento, usado para a associação com o `<label>` e para manipulação via JavaScript.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
          <meta  name="viewport"  content="width=device-width,
initial-scale=1.0">
    <title>Teste de Formulário</title>
</head>
<body>
    <h1>Teste de Formulário</h1>
    <form action="cadastro.php" method="post">
        <p>
            <label for="iusu">Usuário:</label>
            <input type="text" name="usu" id="iusu">
        </p>
        <p>
            <label for="isen">Senha:</label>
            <input type="password" name="sen" id="isen">
        </p>
        <p>
            <input type="submit" value="Enviar">
            <input type="reset" value="Limpar">
        </p>
    </form>
</body>
</html>
```

Agora que temos a estrutura base, vamos detalhar os tipos de `input` utilizados e a importância do método de envio escolhido.

3. Tipos de Input e a Importância do Método POST

A escolha correta do atributo `type` do `input` e do `method` do formulário impacta diretamente a funcionalidade e, principalmente, a segurança dos dados. Cada tipo de `input`

oferece um comportamento específico para o usuário, enquanto o método de envio determina como as informações trafegam até o servidor.

Tipos de `input` Apresentados:

- **`type="text"`**: Utilizado para campos de texto padrão, de uma única linha, como um nome de usuário.
- **`type="password"`**: Específico para campos de senha. Sua principal característica é mascarar os caracteres digitados, exibindo "bolinhas" ou "asteriscos" para proteger a informação de olhares indiscretos.
- **`type="submit"`**: Cria o botão que efetua o envio dos dados do formulário para o destino definido no atributo `action` do `<form>`.
- **`type="reset"`**: Cria um botão que, ao ser clicado, limpa todos os campos do formulário, restaurando-os para seus valores iniciais.

A Diferença Crítica entre os Métodos `get` e `post:`

- O método **`get`** envia os dados do formulário anexados diretamente à URL. Por exemplo, um formulário de login enviado com `get` resultaria em uma URL como: `...cadastro.php?usu=Guanabara&sen=1234`.
- Isso representa um **grave risco de segurança**, pois expõe dados sensíveis, como a senha, de forma visível na barra de endereço do navegador, no histórico e em logs do servidor.
- O método **`post`**, por outro lado, envia os dados no corpo da requisição HTTP. As informações não são visíveis na URL, tornando-o a prática recomendada para formulários de login ou qualquer outro que manipule informações confidenciais.

É fundamental entender, no entanto, que `method="post"` **não é uma forma de criptografia**. Ele apenas oculta os dados da visualização casual na URL. As informações ainda são enviadas como texto simples e podem ser facilmente interceptadas ou visualizadas por meio das ferramentas de desenvolvedor do navegador (na aba "Rede" ou "Network"). Para uma segurança real na transmissão de dados sensíveis, é indispensável o uso do protocolo **HTTPS**, que criptografa toda a comunicação entre o cliente e o servidor. Além da segurança, é preciso garantir que os dados enviados sejam válidos. Para isso, o HTML5 nos oferece atributos de validação.

4. Atributos de Validação e Regras de Negócio

A validação de dados no lado do cliente (*client-side*) é uma prática essencial. O HTML5 fornece atributos que permitem ao próprio navegador verificar os dados antes mesmo de enviá-los ao servidor. Isso melhora a experiência do usuário, fornecendo feedback imediato, e reduz a carga no servidor ao evitar requisições com dados inválidos.

Atributos de Validação:

- **`required`**: Torna o preenchimento de um campo obrigatório. Se o usuário tentar enviar o formulário com o campo vazio, o navegador o impedirá e exibirá uma mensagem padrão, como "Preencha este campo".
- **`minlength`**: Define o número mínimo de caracteres que um campo deve conter. Se o valor inserido for menor, o navegador exibirá uma mensagem de erro específica. Por exemplo, se o mínimo for 8 e o usuário digitar 3, a mensagem pode ser: *"Aumente este texto para 8 caracteres ou mais. No momento, você está usando 3 caracteres."*
- **`maxlength`**: Define o número máximo de caracteres permitidos em um campo. Isso impede que o usuário digite além do limite estabelecido, o que é útil para alinhar o formulário com as restrições do banco de dados.

Conectar a validação de dados à usabilidade é o próximo passo para criar formulários eficazes.

5. Melhorando a Experiência do Usuário (UX)

Além de garantir que os dados sejam válidos, é fundamental tornar o preenchimento do formulário uma tarefa fácil e intuitiva. O HTML oferece atributos específicos para melhorar a experiência do usuário (UX), ajudando-o a preencher os campos de forma mais rápida e com menos erros.

Atributos de Experiência do Usuário:

- **`placeholder`**: Exibe um texto de dica (uma sugestão) dentro do campo de entrada. Esse texto desaparece assim que o usuário começa a digitar. É importante notar que ele serve como uma dica, e não como um valor pré-preenchido.

- **`size`**: Controla a largura visível do campo, definida em número de caracteres. Diferente de `maxlength`, `size` não limita a quantidade de caracteres que podem ser inseridos. Se o texto digitado exceder a largura visual definida pelo `size`, ele simplesmente rolará horizontalmente dentro do campo.
- **`autocomplete`**: Ajuda os navegadores a preencherem os campos automaticamente com base em dados previamente armazenados. Para funcionar corretamente, requer uma configuração em duas partes:
    - **Habilitar no formulário:** A tag `<form>` deve ter o atributo `autocomplete="on"`.
    - **Especificar em cada campo:** Cada `<input>` deve receber um valor semântico que descreva seu propósito. Isso orienta o navegador sobre qual dado deve ser sugerido.
    - `autocomplete="username"`: Ideal para o campo de nome de usuário.
    - `autocomplete="current-password"`: Usado em formulários de login para que o navegador sugira a senha atual e salva para aquele site.
    - `autocomplete="new-password"`: Utilizado em formulários de cadastro ou de alteração de senha, indicando ao navegador que um novo valor deve ser gerado ou sugerido.

A seguir, apresentamos o código final do nosso formulário, que consolida todos os atributos aprendidos.

6. Código Completo e Comentado

O código a seguir representa a versão final do nosso formulário de login, incorporando todos os atributos de segurança (`method="post"`), validação (`required`, `minlength`, `maxlength`) e experiência do usuário (`placeholder`, `size`, `autocomplete`) discutidos nesta apostila.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
          <meta  name="viewport"  content="width=device-width,
initial-scale=1.0">
    <title>Teste de Formulário</title>
</head>
<body>
    <h1>Teste de Formulário</h1>
               <form    action="cadastro.php"    method="post"
autocomplete="on">
        <p>
            <label for="iusu">Usuário:</label>
              <input type="text" name="usu" id="iusu" required
minlength="5"  maxlength="15"  size="10"  placeholder="nome  do
usuário" autocomplete="username">
        </p>
        <p>
            <label for="isen">Senha:</label>
                  <input  type="password"  name="sen"  id="isen"
required      minlength="8"      maxlength="20"      size="8"
placeholder="mínimo 8 letras" autocomplete="current-password">
        </p>
        <p>
            <input type="submit" value="Enviar">
            <input type="reset" value="Limpar">
        </p>
    </form>
</body>
</html>
```

Para facilitar a consulta, a tabela a seguir resume os novos atributos que aprendemos.

7. Resumo dos Novos Atributos

A tabela a seguir serve como um guia de referência rápida para os principais atributos de `input` abordados nesta aula, detalhando a função de cada um.

Atributo	Função
`required`	Torna o preenchimento do campo obrigatório antes do envio do formulário.
`minlength`	Define o número mínimo de caracteres que o campo deve conter.
`maxlength`	Define o número máximo de caracteres que o campo pode conter.
`size`	Controla a largura visível do campo de texto, em número de caracteres.
`placeholder`	Exibe um texto de dica dentro do campo, que desaparece ao digitar.
`autocomplete`	Sugere ao navegador como preencher o campo com base em valores semânticos (`username`, `current-password`, etc.).