CONTROLES AVANÇADOS DE FORMULÁRIO EM HTML5

1. A EVOLUÇÃO DOS FORMULÁRIOS COM HTML5

Esta mini apostila, baseada nos ensinamentos da aula do Curso em Vídeo, explora os novos e poderosos tipos de `input` que foram introduzidos com o HTML5. Esses controles avançados, com foco especial em campos para números, datas e horas, representam um grande salto na usabilidade e na integridade dos dados. Eles aprimoram a experiência do usuário e transferem parte da validação para o próprio navegador, resultando em formulários mais inteligentes e interativos. A seguir, vamos analisar a construção prática de um formulário que incorpora esses novos elementos, começando pela sua estrutura fundamental.

2. ESTRUTURA INICIAL DO FORMULÁRIO

Antes de mergulharmos nos novos controles, é fundamental começar com uma estrutura de formulário HTML bem definida. Nesta seção, revisaremos a configuração básica que servirá como alicerce para os componentes avançados que adicionaremos. O código a seguir estabelece um formulário simples com um campo de nome e botões de envio e limpeza. A análise de seus componentes iniciais nos ajuda a reforçar conceitos importantes:

- **`<label for="inome">`**: A tag `<label>` é crucial para a acessibilidade, pois associa textualmente o rótulo ao seu respectivo campo de entrada (`input`). Isso permite que tecnologias assistivas identifiquem o propósito do campo e melhora a usabilidade, pois clicar no rótulo foca no campo correspondente.
- **`input type="text"`**: Este é o campo de texto padrão. Relembramos seus atributos essenciais, como `name` e `id` para identificação, `required` para torná-lo obrigatório, `maxlength` para limitar o número de caracteres e `autocomplete` para sugerir preenchimentos com base no histórico do navegador.
- **`input type="reset"`**: Este botão tem a função específica de limpar *todos* os campos preenchidos dentro da tag `<form>` em que está inserido, restaurando o formulário ao seu estado inicial.

```
<!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="get" autocomplete="on">
        <p>
            <label for="inome">Nome:</label>
                    <input  type="text"  name="nome"  id="inome"  required
maxlength="30" placeholder="Nome completo" autocomplete="name">
        </p>
        <p>
            <input type="submit" value="Enviar">
            <input type="reset" value="Limpar">
        </p>
    </form>
</body>
</html>
```

Com esta base sólida estabelecida, estamos prontos para explorar o primeiro controle avançado: o campo numérico.

2.1. O Input Numérico: `type="number"`

Tradicionalmente, se você precisasse que um usuário inserisse um número, usaria `type="text"`. O problema é que isso permite que ele digite letras, levando a erros de validação no seu sistema. O HTML5 fornece uma solução direta e muito mais robusta: o `input type="number"`. Ele oferece validação automática pelo navegador e uma interface aprimorada, geralmente com setas de incremento/decremento, melhorando a usabilidade.

Para configurar este campo de forma eficaz, utilizamos atributos específicos: O atributo `step` é particularmente poderoso. Imagine que você configurou o campo de média com `min="0"` e `max="10"`, mas omitiu o `step`. O navegador permitirá apenas números inteiros. Se o usuário

tentar submeter uma nota válida como `5.5`, receberá um erro de validação, pois o valor não corresponde ao passo (que por padrão é `1`). Para permitir notas com casas decimais, você deve adicionar `step="0.1"`. Isso valida valores como `5.6` ou `5.7`. Se a sua instituição usasse apenas meios pontos, você poderia usar `step="0.5"`, mas isso invalidaria uma entrada como `5.2`.

Outro benefício fundamental é a integridade dos dados. Mesmo que um usuário em uma localidade específica digite um decimal com vírgula (ex: `5,5`), o navegador converte automaticamente para um ponto (`5.5`) nos dados enviados, como você pode observar nos parâmetros da URL. Isso garante dados consistentes e padronizados para o seu processamento de back-end.

Dica de especialista: *Lembre-se que os atributos são frequentemente específicos para o tipo de input. Enquanto `maxlength` limita o número de caracteres em um campo de texto, para `type="number"`, você deve usar `min` e `max` para definir o intervalo numérico aceitável. Note também que quando você define um `value` padrão, ele sobrepõe o `placeholder`. O texto do `placeholder` só é visível quando o campo está vazio.*

```
<p>
    <label for="imedia">Média:</label>
        <input  type="number"  name="media"  id="imedia"  min="0"  max="10"
placeholder="0 a 10" step="0.1" required>
</p>
```

Como desenvolvedor, é crucial experimentar. Tente remover o atributo `step` e veja o que acontece. Teste diferentes valores para `min` e `max`. Essa prática é a melhor maneira de internalizar como esses controles se comportam. Agora que dominamos os inputs numéricos, vamos enfrentar outro desafio comum: capturar datas e horas de forma confiável, sem scripts complexos.

2.2. Capturando Datas e Horas com Controles Nativos

A captura de datas e horas é uma tarefa comum e propensa a erros. Os usuários podem digitar formatos inconsistentes (`DD/MM/AAAA`, `MM-DD-AA`, etc.), exigindo validações complexas. Os controles nativos do HTML5 resolvem esse problema ao oferecer interfaces interativas, como calendários e seletores de tempo. Isso elimina a necessidade de o usuário digitar, reduzindo drasticamente os erros de entrada e padronizando os dados enviados ao servidor.

2.2.1. 3.1 Mês e Ano com `type="month"`

Ideal para situações que exigem apenas a seleção do mês e do ano, como a definição de um "Período Letivo", o `input type="month"` apresenta uma interface simplificada para essa finalidade. Ao usar o atributo `value` com o formato `AAAA-MM` (ex: `value="2020-07"`), você pode pré-selecionar aquele mês e ano quando a página carrega. O valor final é enviado ao servidor nesse mesmo formato.

```
<p>
    <label for="imes">Período Letivo:</label>
    <input type="month" name="mes" id="imes" value="2020-07">
</p>
```

2.2.2. Data Completa com `type="date"`

O `input type="date"` é a solução para selecionar uma data completa (dia, mês e ano), renderizando um calendário interativo. Sua maior força é a validação integrada do navegador. Ele torna impossível para um usuário submeter uma data inexistente como "30 de fevereiro", poupando-lhe o trabalho de escrever scripts de validação personalizados. Os dados são enviados no formato padrão `AAAA-MM-DD`. Você pode pré-selecionar uma data usando `value="2020-07-14"`. Embora estejamos definindo isso manualmente para demonstração, em uma aplicação real, você normalmente usaria JavaScript para definir dinamicamente o valor para a data atual.

```
<p>
    <label for="idia">Dia da Prova:</label>
    <input type="date" name="dia" id="idia" value="2020-07-14">
</p>
```

2.2.3. Horário da Prova com `type="time"`

Para capturar um horário específico, o `input type="time"` é a ferramenta ideal. Ele fornece uma interface de relógio amigável, permitindo que o usuário selecione a hora e os minutos facilmente. Quando submetido, o horário é enviado no formato de 24 horas. Esteja ciente de que

caracteres especiais como os dois pontos (`:`) são codificados na URL. Por exemplo, um horário de `19:30` aparecerá na URL como `hora=19%3A30`.

```
<p>
    <label for="ihora">Horário da Prova:</label>
    <input type="time" name="hora" id="ihora">
</p>
```

Construímos nosso formulário peça por peça. Vejamos agora o código completo e consolidado para ver como todas essas entradas poderosas funcionam juntas.

2.3. Código Final Consolidado

A seguir, apresentamos o código-fonte completo do arquivo `form003.html`. Ele integra todos os controles discutidos nesta apostila — texto, número, mês, data e hora — em um único formulário funcional, pronto para consulta e teste.

```
<!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="get" autocomplete="on">
        <p>
            <label for="inome">Nome:</label>
                      <input  type="text"  name="nome"  id="inome"  required
maxlength="30" placeholder="Nome completo" autocomplete="name">
        </p>
        <p>
            <label for="imedia">Média:</label>
             <input type="number" name="media" id="imedia" min="0" max="10"
placeholder="0 a 10" step="0.1" required>
        </p>
        <p>
            <label for="imes">Período Letivo:</label>
            <input type="month" name="mes" id="imes" value="2020-07">
        </p>
        <p>
            <label for="idia">Dia da Prova:</label>
            <input type="date" name="dia" id="idia" value="2020-07-14">
        </p>
        <p>
            <label for="ihora">Horário da Prova:</label>
            <input type="time" name="hora" id="ihora">
        </p>
        <p>
            <input type="submit" value="Enviar">
            <input type="reset" value="Limpar">
        </p>
    </form>
</body>
</html>
```

Embora o código seja funcional, existe uma consideração crucial sobre a aparência e o comportamento desses controles em diferentes ambientes, que abordaremos na conclusão.

2.4. Conclusão e Alerta de Compatibilidade

Os novos tipos de `input` do HTML5 são, sem dúvida, ferramentas poderosas que modernizam a criação de formulários. Eles melhoram a experiência do usuário, padronizam a coleta de dados e adicionam uma camada de validação nativa, tornando o desenvolvimento mais eficiente.
No entanto, é fundamental estar ciente de um alerta importante: **a aparência e o comportamento desses controles não são padronizados entre as diferentes plataformas.**

A renderização dos calendários, seletores de tempo e campos numéricos pode variar significativamente dependendo de múltiplos fatores:

- **Navegador**: A renderização no Chrome, Firefox, Safari e outros pode ser visualmente distinta.
- **Sistema Operacional**: A aparência pode mudar entre Windows, macOS, Android e iOS.
- **Dispositivo**: A experiência em um desktop será diferente da de um dispositivo móvel.

Apesar dessas variações visuais, o ganho mais significativo reside na funcionalidade principal de validação e formatação de dados. Para o desenvolvedor web, compreender e aceitar essas diferenças é um passo essencial para criar formulários robustos e compatíveis, garantindo que a funcionalidade prevaleça sobre uma estética unificada.