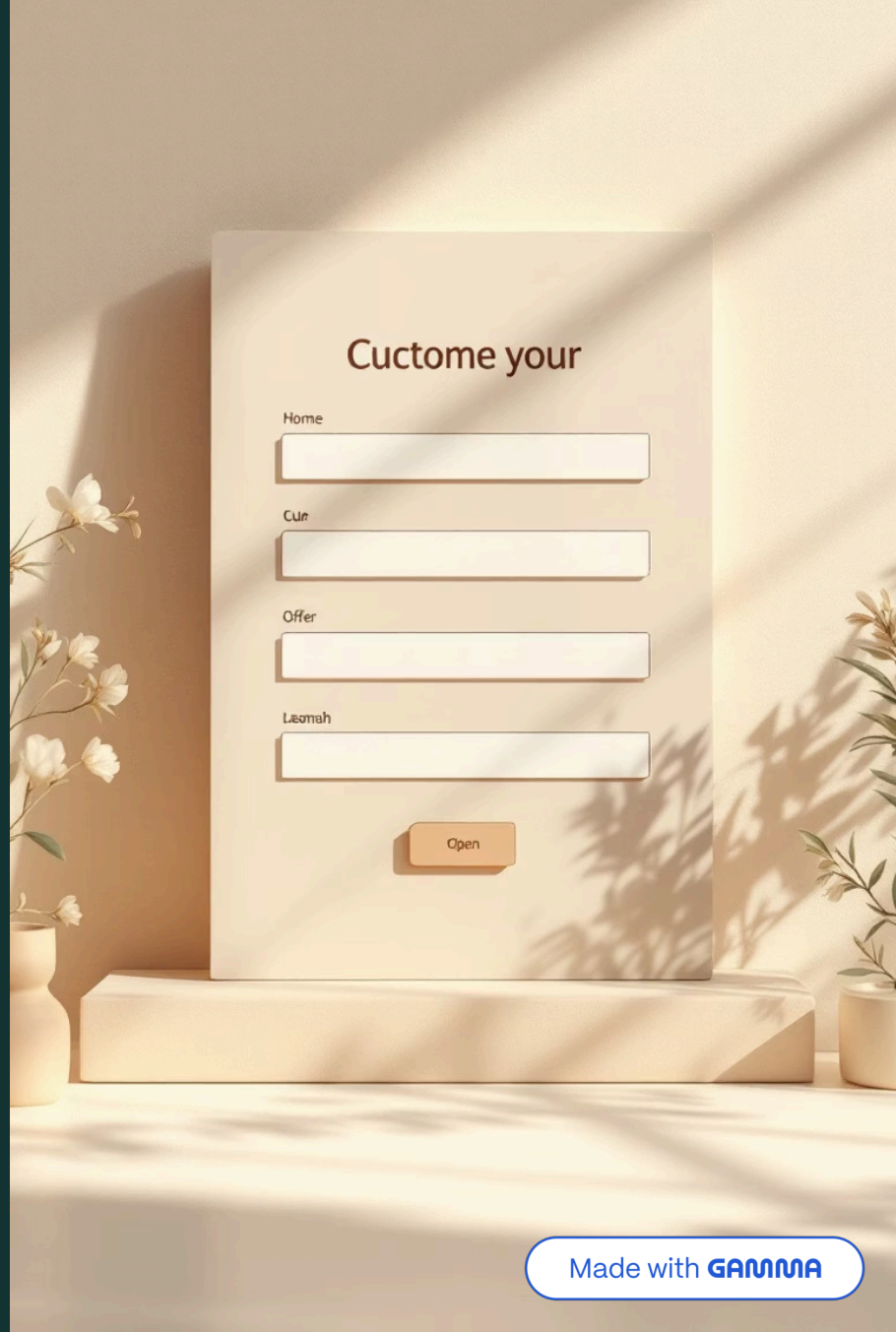
MÓDULO 4 - Aula 02: Formulários e Controles de Entrada

Aula 2 de 8

Aprenda a criar formulários HTML para capturar e processar dados dos usuários de forma profissional.



Revisão e Transição

Da Estrutura à Interação

Formulários são a principal forma de **interação** entre o usuário e o servidor web. Na Aula 1, construímos a **estrutura semântica** do conteúdo usando Headings e Parágrafos.

Agora, vamos além: chegou o momento de capturar dados do usuário, criar experiências interativas e estabelecer comunicação bidirecional entre o navegador e o servidor.

O Elemento <form>

Contêiner Fundamental

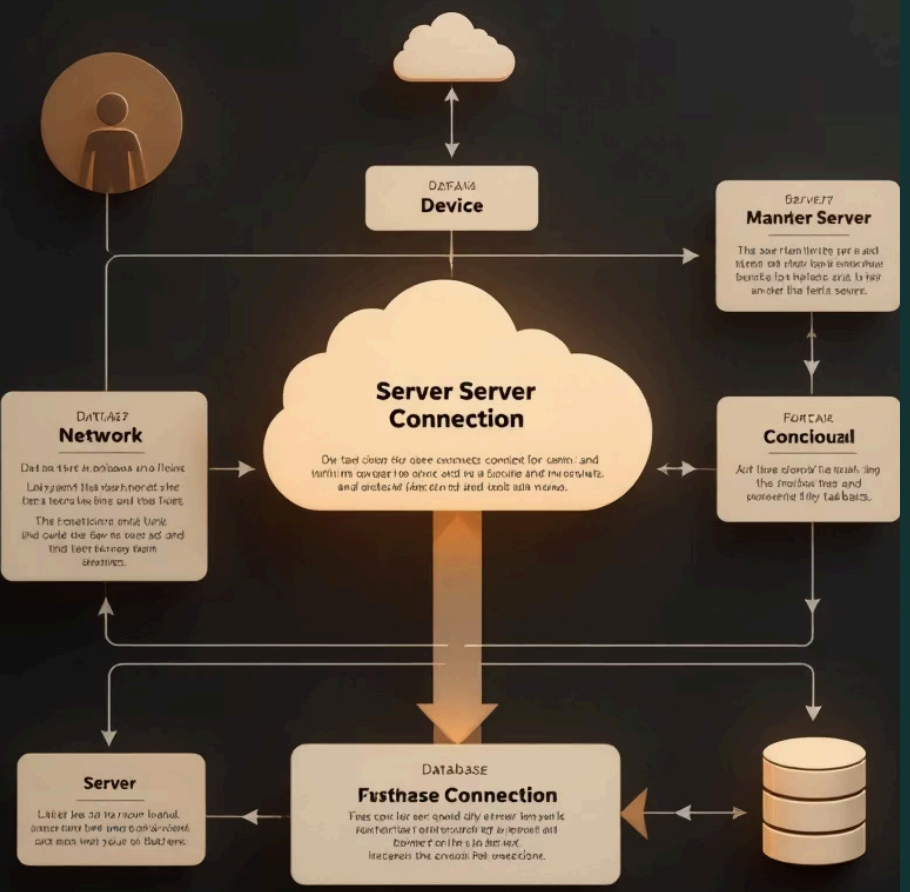
O elemento `<form>` é o recipiente essencial que agrupa todos os campos de entrada, botões e controles de um formulário HTML.

Ponte com Servidor

Define como e para onde os dados coletados serão enviados, estabelecendo a comunicação entre front-end e back-end.

Configuração Completa

Possui atributos específicos que controlam o comportamento, método de envio e formato dos dados transmitidos.



Atributo action: Definindo o Destino

Para Onde os Dados Vão?

O atributo `action` especifica o **URL do programa no servidor** que receberá e processará os dados enviados pelo formulário. É essencialmente o endereço de destino das informações.

Pode ser um arquivo HTML, um endpoint de API REST, ou qualquer script server-side. Exemplo: `action="processa.html"` ou `action="/api/usuarios"`.

Dica: Se o atributo action for omitido, o formulário enviará os dados para a mesma página atual (comportamento padrão).

Atributo method: GET vs POST

Método GET

Características:

- Dados visíveis na URL (query string)
- Ideal para buscas e filtros
- Pode ser favoritado/compartilhado
- Limite de tamanho de dados

Uso: Formulários de pesquisa, filtros de produtos, páginas com parâmetros.

Método POST

Características:

- Dados no corpo da requisição (invisíveis na URL)
- Mais seguro para informações sensíveis
- Sem limite prático de tamanho
- Ideal para registros e login

Uso: Cadastros, login, upload de arquivos, alteração de dados no servidor.

Comparação Visual: GET vs POST

Exemplo GET

```
<form action="/busca" method="GET">
  <input name="q" value="HTML">
</form>
```

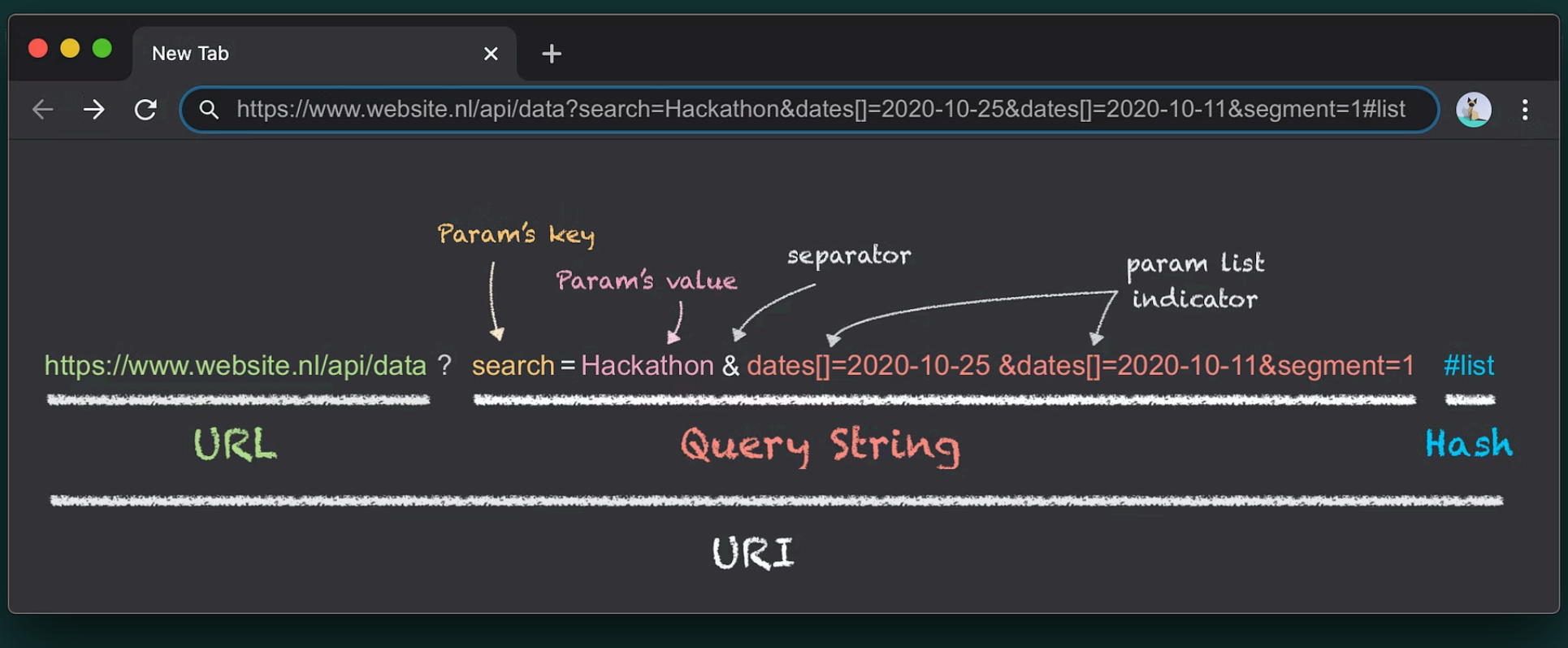
URL resultante:
exemplo.com/busca?q=HTML

Exemplo POST

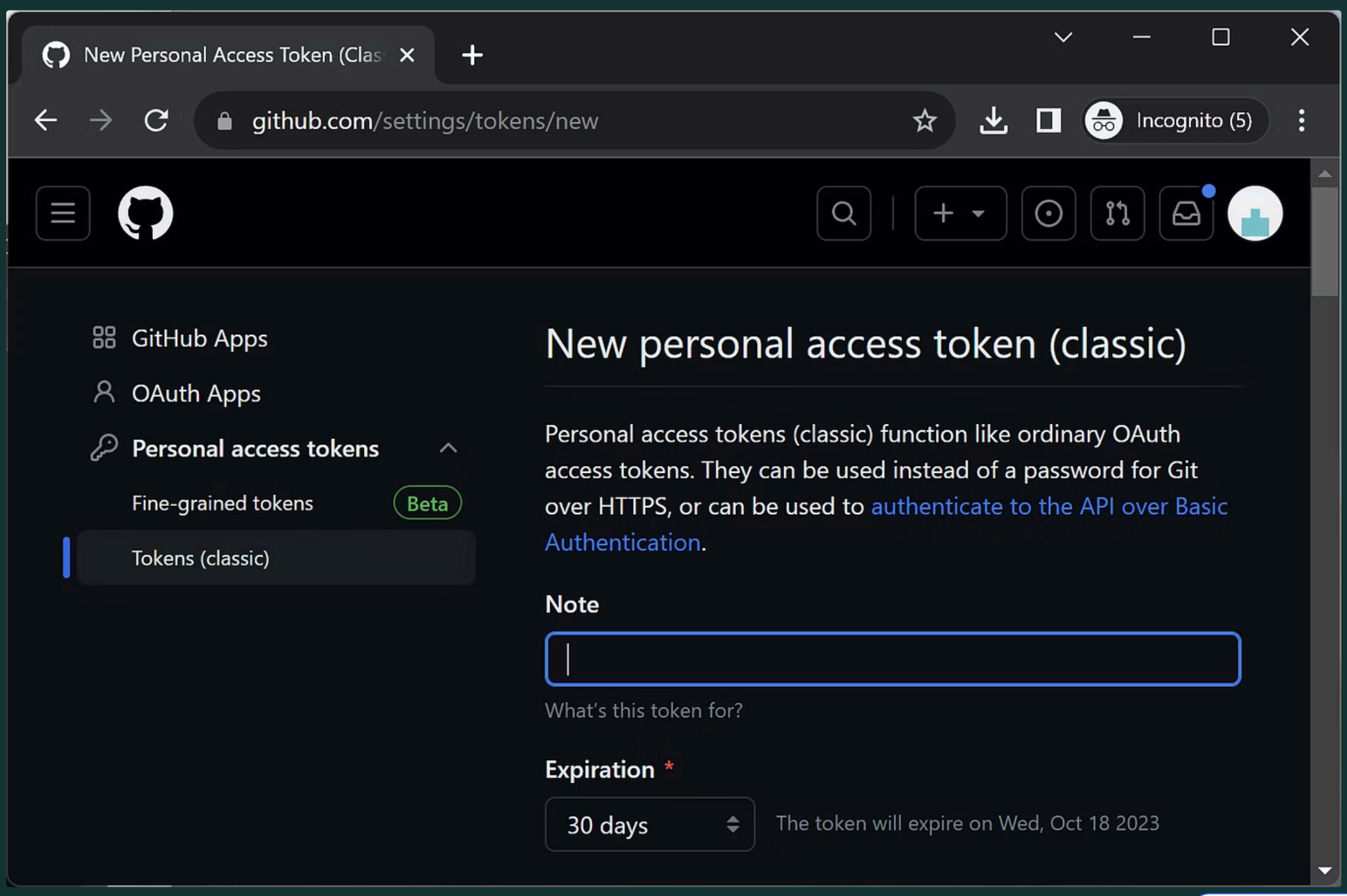
```
<form action="/login" method="POST">
  <input name="email">
  <input name="senha">
</form>
```

URL resultante:
exemplo.com/login
(dados no corpo HTTP)

Exemplo GET



Exemplo POST



Atributo enctype: Codificação dos Dados

Definindo o Formato de Envio

O atributo `enctype` especifica como os dados do formulário devem ser codificados antes do envio ao servidor. É especialmente importante quando trabalhamos com uploads de arquivos.

01

application/x-www-form-urlencoded

Codificação padrão. Converte espaços em + e caracteres especiais em códigos hexadecimais. Usado para formulários simples sem arquivos.

02

multipart/form-data

Essencial para upload de arquivos. Divide os dados em partes, permitindo envio de binários como imagens, PDFs e vídeos sem corrupção.

03

text/plain

Raramente usado. Envia dados sem codificação, apenas para debugging. Não recomendado para produção.

04

Vamos ver um exemplo

www.w3schools.com

W3Schools online HTML editor

The W3Schools online code editor allows you to edit code and view the result in your browser

Upload de Arquivos: Configuração Completa

Configuração Necessária

Para permitir upload de arquivos (imagens, documentos, vídeos), você **DEVE** usar `method="POST"` e `enctype="multipart/form-data"`.

```
<form action="/upload" method="POST"
      enctype="multipart/form-data">
  <label for="arquivo">Escolha um arquivo:</label>
  <input type="file" id="arquivo" name="arquivo">
  <button type="submit">Enviar</button>
</form>
```

Importante: Sem o enctype correto, arquivos não serão enviados adequadamente e chegarão como strings vazias no servidor.

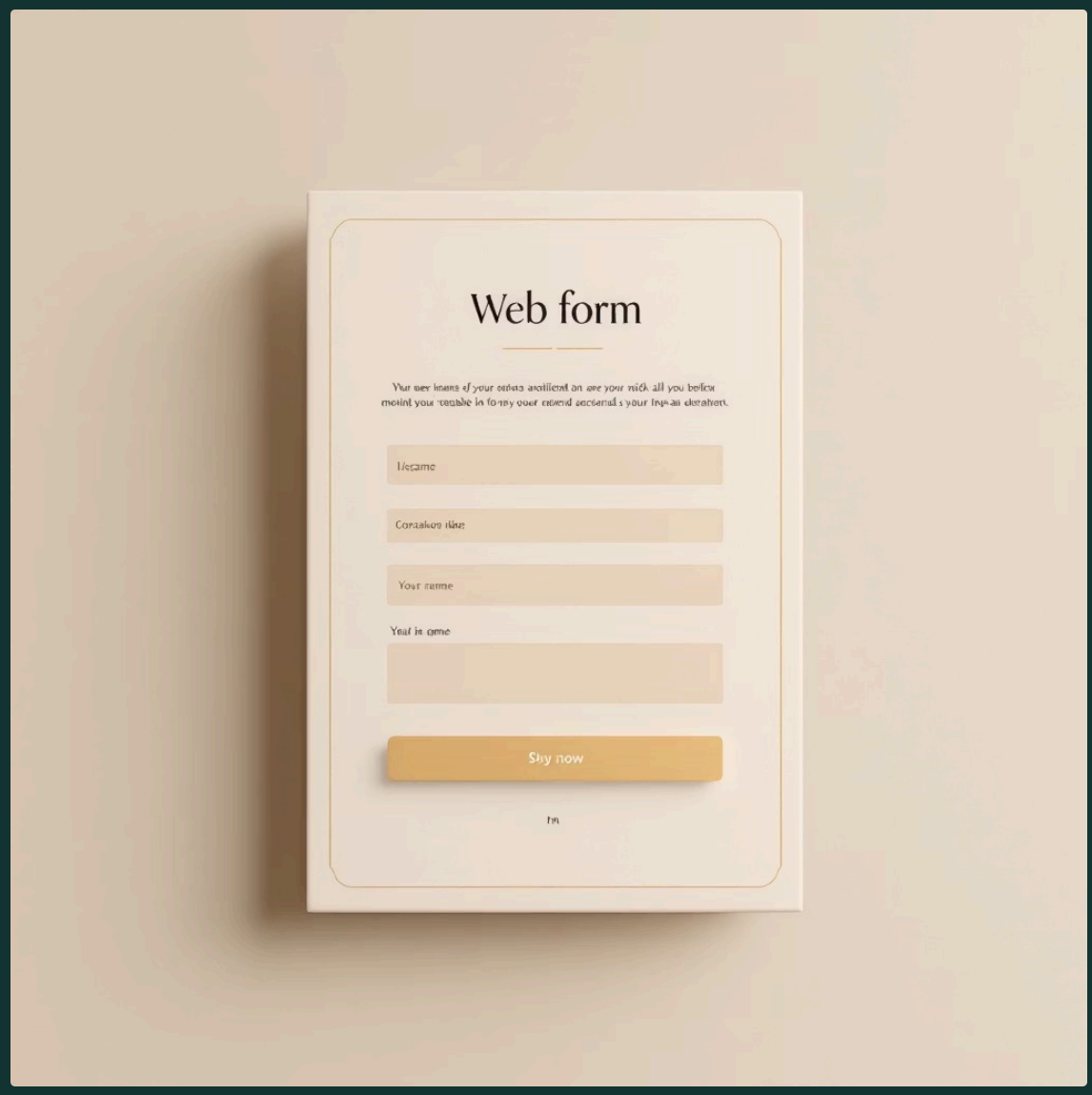
O Elemento <label> e Acessibilidade

Rótulos Semânticos

O `<label>` cria rótulos descritivos para campos de entrada, melhorando acessibilidade e usabilidade.

Atributo for: Conecta o label ao input através do ID, permitindo que clicar no texto foque o campo automaticamente.

Essencial para leitores de tela e navegação por teclado.



Exemplo Prático

```
<form action="processa.php" method="POST">
 <label for="nome">Nome Completo:</label>
 <input type="text" id="nome" name="nome">

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

Estrutura Completa de um Formulário

Exemplo Integrado

Veja como todos os conceitos se conectam em um formulário funcional e bem estruturado:

```
<form action="processa.php" method="POST" enctype="multipart/form-data">

  <label for="usuario">Usuário:</label>
  <input type="text" id="usuario" name="usuario" required>

  <label for="senha">Senha:</label>
  <input type="password" id="senha" name="senha" required>

  <label for="foto">Foto de Perfil:</label>
  <input type="file" id="foto" name="foto" accept="image/*">

  <button type="submit">Cadastrar</button>
</form>
```

ELEMENTO <input>: A ESPINHA DORSAL DO FORMULÁRIO

Tipos e Atributos de Usabilidade

O `<input>` é a tag mais versátil do HTML para coletar dados do usuário. Sendo um elemento vazio, sua função específica é determinada pelo atributo `type`. É fundamental para a interação em qualquer formulário web, permitindo diversas formas de entrada.

Tipos Básicos

- `type="text"` (padrão)
- `type="password"` (oculta a entrada)
- `type="checkbox"` (seleção binária)
- `type="radio"` (seleção única)
- `type="submit"` (botão de envio)

Tipos HTML5

- `type="email"` (valida formato de e-mail)
- `type="url"` (valida formato de URL)
- `type="number"` (apenas números)
- `type="date"` (seletor de data)
- `type="range"` (controle deslizante)

Estes novos tipos do HTML5 oferecem **validação implícita** no navegador, melhorando a experiência do usuário e a robustez do formulário.

Atributos de Usabilidade Essenciais

- **placeholder:** Texto de sugestão dentro do campo antes da digitação.
- **autofocus:** Foca o campo automaticamente ao carregar a página.
- **required:** Torna o preenchimento do campo obrigatório para o envio.
- **min/max/step:** Define limites e incrementos para campos numéricos.

Exemplo HTML

```
<form action="/cadastro" method="POST">
  <label for="nome">Nome Completo:</label>
  <input type="text" id="nome" name="nome" placeholder="Seu nome aqui" autofocus>

  <label for="email">E-mail:</label>
  <input type="email" id="email" name="email" placeholder="seu@email.com" required>

  <input type="checkbox" id="aceite" name="aceite">
  <label for="aceite">Aceito os termos de uso</label>

  <button type="submit">Cadastrar</button>
</form>
```

Elementos <select> e <textarea>: LISTAS E TEXTO MULTILINHA

Controles para Opções Fixas e Entradas Extensas

Além dos diversos tipos de `<input>`, o HTML oferece elementos específicos para gerenciar entradas de texto mais longas ou seleções a partir de listas pré-definidas, garantindo flexibilidade e boa usabilidade.

Elemento <textarea>

Permite ao usuário inserir múltiplas linhas de texto. Ideal para comentários, mensagens ou descrições detalhadas.

- `rows` e `cols`: Definem as dimensões visíveis iniciais do campo.
- `maxlength` **(HTML5):** Limita o número máximo de caracteres permitidos.

Elemento <select>

Cria uma lista suspensa de opções. É composto por elementos `<option>` e pode agrupar opções com `<optgroup>`.

- `multiple`: Permite ao usuário selecionar mais de uma opção simultaneamente.
- `size`: Especifica o número de opções visíveis sem a necessidade de rolagem.

Exemplo HTML

```
<form action="/dados" method="POST">
  <label for="mensagem">Sua Mensagem:</label>
  <textarea id="mensagem" name="mensagem" rows="5" cols="40" maxlength="250"
            placeholder="Digite sua mensagem aqui..."></textarea>

  <label for="cidade">Selecione sua Cidade:</label>
  <select id="cidade" name="cidade">
    <optgroup label="Sudeste">
      <option value="sp">São Paulo</option>
      <option value="rj">Rio de Janeiro</option>
    </optgroup>
    <optgroup label="Sul">
      <option value="poa">Porto Alegre</option>
      <option value="cwb">Curitiba</option>
    </optgroup>
  </select>

  <button type="submit">Enviar</button>
</form>
```

VALIDAÇÃO EM HTML5: ATRIBUTOS NATIVOS

Validação Implícita e Obrigatória

O HTML5 trouxe um conjunto poderoso de atributos nativos que permitem implementar validação de formulários diretamente no navegador, reduzindo a necessidade de JavaScript e melhorando a usabilidade e acessibilidade.

Validação por Tipo (`type`)

Tipos como `email`, `url`, `number`, `date` e `color` forçam automaticamente o formato correto da entrada do usuário, garantindo dados válidos e melhorando a experiência sem código adicional.

Atributo `required`

Este atributo booleano, quando presente, impede o envio do formulário se o campo estiver vazio. É fundamental para garantir que informações essenciais sejam sempre preenchidas antes da submissão.

Validação de Limite e Restrição

Atributos como `maxlength` (limita caracteres), `min` e `max` (definem valores mínimo/máximo para numéricos) e `step` (incremento mínimo aceitável) controlam a amplitude e precisão da entrada.

Desabilitando Validação

Para desativar a validação nativa, use o atributo `novalidate` na tag `<form>`. Alternativamente, `formnovalidate` em um botão de submit desabilita a validação apenas para aquele envio específico (ex: "Salvar Rascunho").

Exemplo Prático

Veja como aplicar alguns desses atributos para controlar a entrada de dados em campos numéricos e de texto.

```
<form action="/processa-dados" method="POST">
  <label for="idade">Idade:</label>
  <input type="number" id="idade" name="idade" min="18" max="99" required>

  <label for="comentario">Comentário (máx. 100 caracteres):</label>
  <textarea id="comentario" name="comentario" rows="4" maxlength="100" required></textarea>

  <button type="submit">Enviar</button>
</form>
```

VALIDAÇÃO AVANÇADA: pattern E LÓGICA CUSTOMIZADA

Expressões Regulares e API de Validação

O HTML5 expande as capacidades de validação, permitindo controle fino sobre o formato dos dados de entrada e até mesmo validações complexas com a ajuda do JavaScript, mantendo a experiência do usuário consistente com o navegador.

Validação de Formato com `pattern`

Permite definir **Expressões Regulares** para validar o formato exato da entrada (ex: CEP, telefone regional). Se o padrão não for atendido, o campo é considerado inválido.

Uso com Atributo `title`

Use `title` em conjunto com `pattern` para informar o usuário sobre o formato esperado. Isso melhora a usabilidade e acessibilidade, exibindo uma dica útil em caso de erro.

Exemplo HTML com `pattern` e `title`

```
<form action="/contato" method="POST">
  <label for="telefone">Telefone (XX) XXXXX-XXXX:</label>
  <input type="tel" id="telefone" name="telefone"
         pattern="^\(\d{2}\) \d{5}-\d{4}$"
         title="Formato: (XX) XXXXX-XXXX" required>

  <button type="submit">Enviar</button>
</form>
```

Validação Customizada com JavaScript

Para validações que dependem de algoritmos complexos (ex: CPF/CNPJ), o JavaScript é essencial. A API de Validação do HTML5 permite integrar sua lógica customizada com a interface nativa do navegador.

- **O Evento `oninput`**
  É usado para checar a validade em tempo real, disparado a cada modificação do valor do campo. Ideal para dar feedback instantâneo ao usuário.
- **O Método `setCustomValidity()`**
  Esta API permite marcar um campo como inválido e definir uma mensagem de erro customizada. Se a string estiver vazia, o campo é considerado válido.
- **Integração com o Navegador**
  A grande vantagem é que o navegador cuida da interface de erro (ex: tooltip) e impede o envio do formulário, mesmo que a lógica de validação seja totalmente em JS.

Exemplo HTML e JavaScript para CPF

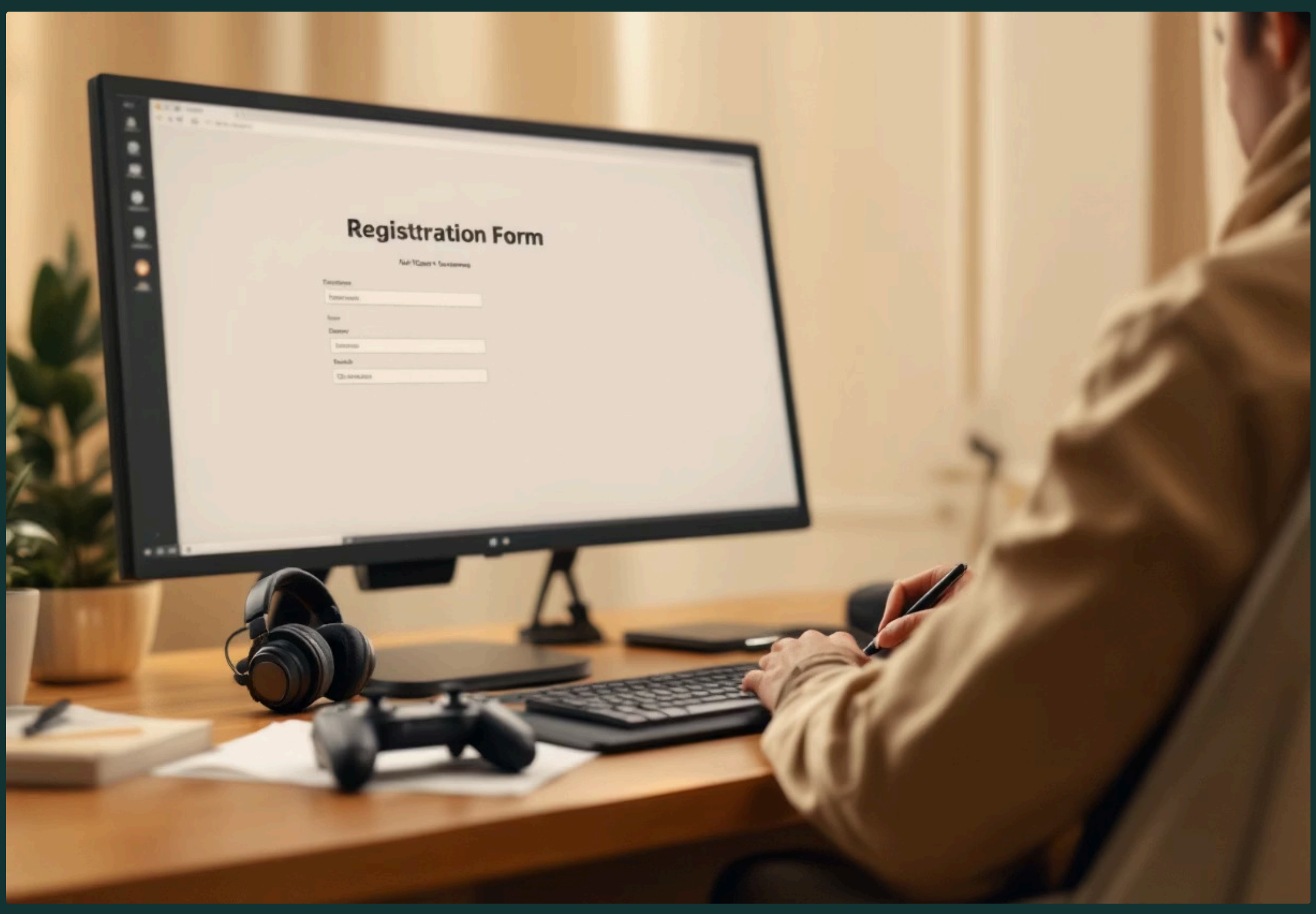
```
<form action="/cadastro-doc" method="POST">
  <label for="cpf">CPF:</label>
  <input type="text" id="cpf" name="cpf"
         placeholder="000.000.000-00" required
         oninput="validarCpf(this)">

  <button type="submit">Cadastrar</button>
</form>

<script>
  function validarCpf(input) {
    const cpf = input.value.replace(/\D/g, ''); // Remove caracteres não numéricos
    if (cpf.length !== 11 || !/^\d{11}$/.test(cpf)) {
      input.setCustomValidity('CPF inválido: Deve conter 11 dígitos.');
    } else {
      // Lógica de validação de CPF mais complexa aqui
      // Por simplicidade, vamos considerar válido se tiver 11 dígitos
      input.setCustomValidity(''); // Campo válido
    }
    input.reportValidity(); // Exibe ou remove a mensagem de erro
  }
</script>
```

PRÁTICA ORIENTADA: Formulário de Registro em Campeonato E-Sports

Desenvolver um formulário de registro para um campeonato de e-sports é uma excelente maneira de aplicar os conceitos de HTML5 sobre formulários e validações. Este exercício integra diferentes tipos de inputs, selects e textareas, garantindo que os dados sejam coletados de forma eficiente e validada.



Cenário: Formulário de Inscrição para a "Copa DevMaster"

O objetivo é criar a marcação HTML para o formulário, utilizando os atributos de validação nativa do HTML5 e simulando uma validação customizada via JavaScript para campos complexos como o Documento de Identificação. A semântica correta e a acessibilidade são fundamentais.

Tarefas e Requisitos de Implementação

1. **Estrutura Base e Envio de Dados**:
   a. Crie o elemento <**form**> no <**body**> do documento.
   b. Defina os atributos action (simule processar-inscricao.html) e **method** como **POST**.
2. **Identificação do Jogador (Inputs Básicos e Novos Tipos)**:
   a. Crie um campo de Nome Completo (type="text") e o torne obrigatório (usando o atributo required).
   b. Crie um campo de Apelido/Nickname (type="text"). ◦
   c. Crie um campo de E-mail (type="email") e o torne obrigatório. O navegador deve validar o formato user@domain.com automaticamente..
   d. **IMPORTANTE**: Associe um elemento a cada campo de entrada usando o atributo for.
3. **Configurações do Torneio (Controles Adicionais)**:
   a. Crie um campo de Senha (type="password").
   b. Use o atributo pattern e title na senha para exigir pelo menos 8 caracteres, incluindo uma letra maiúscula e um número (Simule a expressão regular no pattern e use o title para explicar a regra ao usuário).
   c. Crie uma lista suspensa para que o jogador escolha seu "Jogo Principal" (com pelo menos 3 opções, como "League of Legends", "Valorant", "Fortnite").
   d. Adicione o atributo required ao .
4. **Comentários e Data de Nascimento:**
   a. Crie um campo de Comentário (usando) para que o jogador insira uma breve biografia ou mensagem para a equipe organizadora.
   b. Use o atributo maxlength no <textarea> para limitar a 200 caracteres.
   c. Crie um campo de Data de Nascimento (type="date").
5. **Validação Customizada (Simulação de CPF/CNPJ):**
   a. Crie um campo de texto (type="text") rotulado "Documento de Identificação".
   b. Explique que este campo exige uma validação de lógica (algoritmo) (como CPF/CNPJ) que o HTML5 não pode fazer sozinho.
   c. Simule a integração JS: Adicione o atributo oninput="verificaDocumento(this)" (não é necessário escrever a função JavaScript real, apenas o hook). Explique que essa função chamaria setCustomValidity() se a lógica falhasse, e setCustomValidity('') se fosse válida.
6. **Botão de Submissão:**
   a. Adicione um botão de envio (type="submit") com o texto "Inscrever-se na Copa DevMaster".

Dicas:

- Lembre-se do HTML Semântico: use a tag correta para o propósito correto.
- O atributo name é crucial para que os dados sejam enviados ao servidor.
- Os Atributos Globais (id, class) devem ser usados em alguns elementos (como a classe em um grupo de inputs para estilização futura).

PRÓXIMOS PASSOS: ESTRUTURA PARA ESTILO

Onde Chegamos em HTML

- **Formulários (`<form>`)**

  Contêiner fundamental para coletar dados, definindo o destino (`action`) e o método de envio (`method`).

- **Controles de Entrada**

  Cobrimos elementos essenciais de interação como `<input>` (com seus diversos tipos), `<select>` e `<textarea>`.

- **Validação HTML5**

  Usamos validações nativas importantes como `required`, `type="email"` e `pattern` para garantir a qualidade dos dados no cliente.

- **Acessibilidade**

  Reforçamos o uso do `<label>` para rotular corretamente cada campo, essencial para uma boa experiência do usuário.

Próxima aula: CSS e Seletores

- **HTML e CSS**

  O HTML define a **estrutura** (a "alma" da página); o **CSS** (Cascading Style Sheets) controla a **formatação** (o visual).

- **O Desafio do CSS**

  Para formatar um elemento (cor, layout, etc.), o CSS precisa saber qual elemento do HTML deve ser afetado.

- **Tema da Próxima Aula**

  A Aula focará em **Seletores** (por nome, id, class) e **Especificidade** (como o navegador resolve conflitos de estilo).

Destaque: O CSS é responsável por controlar o visual da informação exibida pelo HTML. Nosso conteúdo está pronto; agora vamos deixá-lo bonito!