FMX Academy

INFORMAÇÕES		
MÓDULO	**AULA**	**INSTRUTOR**
04 JavaScript Essencial e Consumo de APIs	**05 - Validação de Formulários**	**Pedro Henrique Amadio**

CONTEÚDO

1. Nas Últimas Aulas...

Nas aulas anteriores aprendemos:

- **Aula 03:** como modularizar o código usando **funções**, entender **escopos** e reutilizar lógicas.
- **Aula 04:** como manipular o **DOM** e responder a **eventos de usuário** (cliques, inputs, teclas).
  Esses dois conhecimentos agora se unem: Vamos capturar eventos de formulários e **chamar funções de validação** que usam condições, manipulação do DOM e funções reutilizáveis.

2. Introdução à Validação de Formulários

Formulários são o ponto de contato entre o usuário e a aplicação.
Antes de enviar os dados para o servidor, precisamos garantir que estejam corretos.

Validação client-side é feita no navegador, com JavaScript, **antes do envio.**

Exemplo de problema comum:

```
<form>
  <input type="text" placeholder="Digite seu e-mail" />
  <button>Enviar</button>
</form>
```

Se o usuário digitar “pedro” e clicar em enviar, o servidor receberá algo inválido.
Podemos evitar isso validando **no front-end.**

3. Estrutura Base de um Formulário (Simples)

```
<p id="mensagem"></p>
<form id="cadastro">
  <label>Nome:</label>
  <input type="text" id="nome" placeholder="Seu nome completo" />

  <label>E-mail:</label>
  <input type="email" id="email" placeholder="exemplo@email.com" />

  <label>Idade:</label>
  <input type="number" id="idade" placeholder="Digite sua idade" />

  <label>Senha:</label>
  <input type="password" id="senha" placeholder="Crie uma senha" />

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

Nesse exemplo temos somente um elemento responsável por renderizar os erros que validamos via JavaScript.

4. Capturando o Evento de Envio (submit)

O formulário possui o evento submit, que é disparado ao clicar em **enviar**.

```
const form = document.getElementById("cadastro");

form.addEventListener("submit", function (e) {
  e.preventDefault(); // impede o envio automático
  console.log("Formulário enviado!");
});
```

`e.preventDefault()` evita o comportamento padrão (recarregar a página).

5. Validação Campo a Campo (Com alertas)

Temos duas formas de obter as informações de um formulário

- Usando `document.getElementById("nome").value` para cada um dos elementos

  ```
  const nome = document.getElementById("nome").value.trim();
  ... Código ...
  ```

  Repare que dessa forma teremos acesso ao elemento e seus valores mas teremos que repetir a estratégia para cada elemento <input> que desejamos validar.

- Usando `new FormData(userInfo);`

```
const form = document.querySelector("#cadastro");
const formData = new FormData(form);
formData.append(“serialnumber”, 12345); //Cria um novo campo “serialnumber” e define valor 12345

const nome = formData.get(“nome”); //Pega o valor de “nome”

... Código ...
... OU ...
document.querySelector(“#cadastro”).addEventListener(“submit”, (e) => {
  e.preventDefault()
  const formData = new FormData(e.target);


  const nome = formData.get(“nome”); //Pega o valor de “nome”

});
```

5.1. Validando o Nome

Queremos garantir que o nome tenha pelo menos **3 caracteres** e não seja vazio.

```
const nome = document.getElementById("nome").value.trim();

if (nome.length < 3) {
  alert("O nome deve ter pelo menos 3 caracteres.");
  return;
}
... Código ...
```

5.2. Validando o E-mail com Regex

Uma **expressão regular (RegEx)** permite verificar padrões.

```
const email = document.getElementById("email").value.trim();
const regexEmail = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (!regexEmail.test(email)) {
  alert("Digite um e-mail válido!");
  return;
}
... Código ...
```

5.3. Validando a Idade

Vamos impedir valores negativos e menores de 18.

```
const idade = Number(document.getElementById("idade").value);

if (idade < 18 || isNaN(idade)) {
  alert("Você deve ter pelo menos 18 anos.");
  return;
}
... Código ...
```

5.4. Validando Senha

A senha deve ter pelo menos 6 caracteres e conter um número.

```
const senha = document.getElementById("senha").value;
const regexSenha = /^(?=.*[0-9]).{6,}$/;

if (!regexSenha.test(senha)) {
  alert("A senha deve ter pelo menos 6 caracteres e conter um número.");
  return;
}
... Código ...
```

6. Exibindo Mensagens no DOM (sem alert)

Podemos melhorar a UX exibindo as mensagens diretamente na página:

```
const msg = document.getElementById("mensagem");

function mostrarMensagem(texto, cor = "red") {
  msg.textContent = texto;
  msg.style.color = cor;
}
```

E substituir os alert() por:

```
mostrarMensagem("Preencha o nome corretamente!");
```

Se tudo estiver certo:

```
mostrarMensagem("Cadastro realizado com sucesso!", "green");
```

7. Juntando Tudo

```
form.addEventListener("submit", function (e) {
  e.preventDefault();

  const nome = document.getElementById(“nome”).value.trim();
  const email = document.getElementById(“email”).value.trim();
  const idade = Number(document.getElementById(“idade”).value);
  const senha = document.getElementById(“senha”).value;

  if (nome.length < 3) return mostrarMensagem(“Nome inválido!”);
  if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email)) return mostrarMensagem(“E-
mail inválido!”);
  if (idade < 18 || isNaN(idade)) return mostrarMensagem(“Idade inválida!”);
  if (!/^(?=.*[0-9]).{6,}$/.test(senha)) return mostrarMensagem(“Senha
fraca!”);

  mostrarMensagem(“Cadastro realizado com sucesso!”, “green”);
});
```

8. Dicas de Boas Práticas

- Sempre use **.trim()** para limpar espaços extras.

- Combine **validações simples + RegEx.**
- Evite validações apenas no backend - o front deve ajudar o usuário a corrigir os erros antes mesmo de chegar ao servidor.
- Prefira **mensagens amigáveis e diretas.**
- Mantenha o código modular (funções separadas para cada campo).

9. Expressões Regulares Mais Comuns

PROPÓSITO	REGEX	EXEMPLO VÁLIDO
E-mail	/^[^\s@]+@[^\s@]+\.[^\s@]+$/	pedro@email.com
Apenas letras	/^[A-Za-zÀ-ÿ\s]+$/	João Silva
Apenas números	/^[0-9]+$/	12345
Senha forte (letra, número e símbolo)	/^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&]).{6,}$/	P@ssw0rd

10.Atividade Prática em Sala

Monte um formulário de cadastro com os seguintes campos:

- Nome completo
- E-mail
- Idade
- Senha
- Confirmar senha
- Use RegEx para validar e-mail e senha.
- O botão “Cadastrar” só deve funcionar se **todas as validações** passarem.
- Abaixo de **cada campo**, crie uma pequena área de feedback:

```
<input id=”nome” placeholder=”Seu nome completo” />
<p data-error-for=”nome”></p>
```

  Essas mensagens de erro/sucesso serão controladas dinamicamente.
- Exiba mensagens coloridas usando o atributo data-error-for associado a cada input
- Aplique cores no **texto e na borda do input** conforme o tipo de mensagem.

Função sugerida

```
function exibirMensagem(id, texto, tipo = “error”) {
  const input = document.getElementById(id);
  const msg = document.querySelector(`[data-error-for=”${id}”]`);

  // Reset visual
  if (tipo === “reset”) {
    msg.textContent = “”;
    input.style.borderColor = “”;
    return;
  }

  msg.textContent = texto;
  msg.style.fontSize = “0.9rem”;
  msg.style.marginTop = “2px”;

  if (tipo === “error”) {
    msg.style.color = “red”;
    input.style.borderColor = “red”;
  } else if (tipo === “success”) {
    msg.style.color = “green”;
    input.style.borderColor = “green”;
  }
}
```

Dica:

- Chame exibirMensagem(“email”, “E-mail inválido!”) quando o campo estiver incorreto.
- Ao corrigir o valor, use exibirMensagem(“email”, “”, “reset”) para limpar o feedback.
- Ao enviar com sucesso, exiba todas as mensagens com tipo “success”.

11.Tarefa de Casa

Crie uma página chamada formulario-login.html com os seguintes campos:

- E-mail
- Senha

Abaixo de **cada campo**, adicione um parágrafo para mensagens de erro ou sucesso usando o atributo personalizado data-error-for:

```
<input id=”email” placeholder=”Digite seu e-mail” />
<p data-error-for=”email”></p>

<input id=”senha” type=”password” placeholder=”Digite sua senha” />
<p data-error-for=”senha”></p>
```

Desafios:

- Valide o **e-mail** com RegEx e exiba mensagem em vermelho se for inválido.
- A senha deve ter pelo menos **6 caracteres e 1 número**.
- Use a função exibirMensagem(id, texto, tipo) do exercício em sala para colorir o texto e a borda dos campos conforme o resultado.
- Ao corrigir o valor do campo, limpe a mensagem com o tipo 'reset'.

- Se todas as validações passarem, exiba a mensagem **“Login efetuado com sucesso!”** com cor verde.

Função reutilizável

```
function exibirMensagem(id, texto, tipo = “error”) {
  const input = document.getElementById(id);
  const msg = document.querySelector(`[data-error-for=”${id}”]`);

  if (tipo === “reset”) {
    msg.textContent = “”;
    input.style.borderColor = “”;
    return;
  }

  msg.textContent = texto;
  msg.style.fontSize = “0.9rem”;
  msg.style.marginTop = “2px”;

  if (tipo === “error”) {
    msg.style.color = “red”;
    input.style.borderColor = “red”;
  } else if (tipo === “success”) {
    msg.style.color = “green”;
    input.style.borderColor = “green”;
  }
}
```

Dica:

- Valide os campos no evento submit do formulário.
- Ao finalizar com sucesso, exiba todas as mensagens de feedback com tipo = 'success' e limpe os campos.