

Documentação do Código de Validação e Sanitização do Formulário

Objetivo do Código

O código tem como objetivo **proteger e validar os dados** digitados em um formulário de contato antes de enviá-los para o servidor.

Isso é importante porque:

- Evita **erros de usuário** (e-mails inválidos, mensagens muito curtas etc).
- Reduz o risco de **ataques XSS** (inserção de código malicioso no navegador).
- Ajuda a evitar **SQL Injection** no back-end, garantindo que apenas dados “limpos” cheguem ao servidor.

1. Sanitização (Proteção contra XSS e caracteres perigosos)

```
function sanitizeInput(input) {  
    return input.replace(/[<>]/g, "").trim();  
}
```

- Remove os caracteres < e > para que ninguém consiga injetar código HTML/JavaScript no formulário.
- Usa .trim() para eliminar espaços extras no início e no fim.

 Essa é a **primeira camada de segurança** contra XSS.

2. Validadores (SRP – cada função faz uma única tarefa)

Essas funções testam se o valor digitado é válido:

```
function isEmpty(value) {  
    return value.length > 0; // Garante que o campo não esteja vazio  
}
```

```
function isValidEmail(email) {  
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;  
    return regex.test(email); // Verifica se o e-mail segue o formato padrão  
}
```

```
function isValidPhone(phone) {
```

```
const regex = /^\\+?\\d{8,15}$/;
return regex.test(phone); // Aceita números entre 8 e 15 dígitos
(com ou sem +)
}

function isValidMessage(message) {
  return message.length >= 5; // Mensagem precisa ter pelo menos 5
  caracteres
}
```

👉 Usamos o **princípio SRP (Single Responsibility Principle)**:
Cada função tem **apenas uma responsabilidade**, o que facilita manutenção e leitura.

3. Tratamento de Erros e Alertas

```
function showError(inputElement, message) {
  inputElement.classList.add("error"); // Destaca o campo com erro
  showFormAlert("error", message);     // Mostra alerta em vermelho
}

function clearError(inputElement) {
  inputElement.classList.remove("error"); // Remove destaque se
  corrigido
}
```

O **alerta visual** é exibido logo abaixo do formulário:

- **Vermelho claro** → quando há erro.
 - **Verde claro** → quando o formulário é enviado com sucesso.
 - O alerta tem botão de fechar e desaparece sozinho após 10 segundos.
-

4. Manipulação do Formulário

```
function handleFormSubmit(event) {
  event.preventDefault(); // Impede envio automático (recarregar
  página)

  const nameInput = document.getElementById("nome");
  const emailInput = document.getElementById("email");
  const phoneInput = document.getElementById("tel");
  const messageInput = document.getElementById("msg");

  // Sanitização
  const name = sanitizeInput(nameInput.value);
  const email = sanitizeInput(emailInput.value);
  const phone = sanitizeInput(phoneInput.value);
  const message = sanitizeInput(messageInput.value);

  let isValid = true;

  // Validação campo a campo
```

```

    if (!isEmpty(name)) { showError(nameInput, "Name is required.");
    isValid = false; }
    else { clearError(nameInput); }

    if (!isValidEmail(email)) { showError(emailInput, "Invalid email
format."); isValid = false; }
    else { clearError(emailInput); }

    if (!isValidPhone(phone)) { showError(phoneInput, "Phone must
contain only numbers (8-15 digits)."); isValid = false; }
    else { clearError(phoneInput); }

    if (!isValidMessage(message)) { showError(messageInput, "Message
must be at least 5 characters long."); isValid = false; }
    else { clearError(messageInput); }

    // Envio final
    if (isValid) {
        console.log("Form submitted:", { name, email, phone, message });
        showFormAlert("success", "Form successfully submitted!");
        event.target.reset(); // Limpa o formulário
    }
}

```

👉 Aqui acontece o **fluxo principal**:

1. O código lê os valores digitados.
2. Passa por **sanitização**.
3. Passa pelas **validações**.
4. Se houver erro → alerta vermelho.
5. Se tudo ok → alerta verde + console.log (pode ser substituído por envio ao servidor).

5. Inicialização

```
document.getElementById("send").addEventListener("click",
handleFormSubmit);
```

- Quando o usuário clicar no botão **Enviar**, a função `handleFormSubmit` será executada.

Segurança e Boas Práticas


1. **Front-end (este código):**
 - Remove caracteres perigosos (<, >, {, }, ;, --, /* */).
 - Impede que o usuário insira HTML/JS malicioso no campo.
 - Valida formato de e-mail e telefone.
 - Limita tamanho mínimo da mensagem.
2. **Back-end (servidor):**

- Sempre usar **prepared statements** (consultas parametrizadas) para evitar **SQL Injection**.
 - Nunca confiar apenas na validação do front-end.
-

Resumo

Este código é uma **primeira camada de defesa**:

- **Sanitiza** → remove caracteres estranhos.
- **Valida** → garante formato correto dos dados.
- **Informa** → mostra ao usuário mensagens de erro/sucesso de forma clara.

 O próximo passo seria integrar com um **back-end seguro**, que continue aplicando regras contra **SQL Injection e XSS**.