

# **Centro Universitário Anhanguera**

**Curso de tecnologia em Análise de Desenvolvimento de  
Sistemas**

**Cleyton Pereira dos Santos**

**R.A: 3570521902**

**JavaScript**

**RELATÓRIO: Validação de E-mail com JavaScript**

**São Paulo**

**2024**

# SUMÁRIO

<b>1. RESUMO</b>	<b>3</b>
Principais resultados:	3
<b>2. INTRODUÇÃO</b>	<b>3</b>
<b>3. Objetivo</b>	<b>4</b>
<b>4 Procedimentos</b>	<b>4</b>
<b>5 Código Desenvolvido</b>	<b>5</b>
• Observações Finais:	14
• Melhorias Futuras:	14
• Aprendizado:	14
<b>REFERÊNCIAS</b>	<b>15</b>

## 1. RESUMO

Este relatório apresenta um projeto de validação de e-mail em JavaScript, desenvolvido através de um formulário HTML com feedback visual instantâneo. A solução utiliza expressões regulares para verificar o formato do e-mail (ex: nome@dominio.com), exibindo mensagens de erro ou sucesso conforme a entrada do usuário.

### Principais resultados:

- Validação em tempo real durante a digitação
- Interface intuitiva com cores e ícones indicativos
- Cobertura de diversos formatos de e-mail (incluindo subdomínios)
- Feedback claro para o usuário

## 2. INTRODUÇÃO

Nesta aula prática da disciplina de Desenvolvimento em JavaScript, o objetivo foi criar um formulário HTML simples com um campo de entrada para e-mail e implementar uma validação desse campo utilizando JavaScript. A validação foi feita por meio de uma expressão regular, que verifica se o e-mail inserido pelo usuário está no formato correto. Além disso, foram exibidas mensagens de erro e orientação ao usuário, caso o e-mail estivesse incorreto. Essa atividade permitiu a aplicação de conceitos como manipulação do DOM e uso de expressões regulares.

### 3. Objetivo

- Criar um formulário HTML com um campo de entrada para e-mail.
- Implementar um script JavaScript para validar o formato do e-mail.
- Exibir mensagens de erro e orientação ao usuário, caso o e-mail esteja incorreto.
- Aplicar conceitos de manipulação do DOM e expressões regulares para validação de entrada de dados.

### 4 Procedimentos

- Acessei o [Playcode.io](https://playcode.io) e criei um novo projeto.
- Escolhi um ambiente de desenvolvimento vazio.
- Estructurei um formulário HTML contendo um campo de entrada para e-mail e um botão de envio.
- Desenvolvi um script JavaScript para validar o formato do e-mail digitado.
- Configurei mensagens de erro para e-mails inválidos e uma mensagem de sucesso para entradas corretas.
- Realizei testes com diferentes formatos de e-mail para verificar a funcionalidade do código.



## 5 Código Desenvolvido

```
<!DOCTYPE html>
```

```
<html lang="pt-br">
```

```
<head>
```

```
<!-- Configurações básicas do documento -->
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Validação de E-mail Moderna</title>
```

```
<!-- Importação de fontes e ícones -->
```

```
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&displ
ay=swap" rel="stylesheet">
```

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
```

```
</head>
```

```
<body>
```

```
<!-- Container principal do formulário -->
```

```
<div class="container">
```

```
<h1><i class="fas fa-envelope"></i> Formulário de Contato</h1>
```

```
<!-- Formulário com id para manipulação via JavaScript -->
```

```
<form id="contactForm">
```

```
<div class="form-group">
```

```
<label for="email">Endereço de E-mail</label>
```

```
<!-- Campo de entrada de e-mail -->
```

```
<input type="text" id="email" name="email"
placeholder="exemplo@dominio.com">
```

```
<!-- Mensagens de feedback (inicialmente ocultas) -->
```

```
<div id="emailError" class="feedback error-message" style="display: none;">
```

```
<i class="fas fa-exclamation-circle"></i>
```

```
<span></span> <!-- Espaço para mensagem de erro dinâmica -->
```

```
</div>
```

```
<div id="emailInfo" class="feedback info-message" style="display: none;">
```

```

        <i class="fas fa-info-circle"></i>

        <span></span> <!-- Espaço para mensagem informativa -->

    </div>

    <div id="emailSuccess" class="feedback success-message" style="display:
none;">

        <i class="fas fa-check-circle"></i>

        <span>E-mail válido!</span> <!-- Mensagem fixa de sucesso -->

    </div>

    <!-- Exemplo de formato de e-mail para orientar o usuário -->

    <div class="email-example">

        <strong>Formato esperado:</strong> nome@dominio.extensão (ex:
usuario@exemplo.com)

    </div>

</div>

    <!-- Botão de submit do formulário -->

    <button type="submit">

        <i class="fas fa-paper-plane"></i> Enviar

    </button>

</form>

</div>

<script>

    // Event listener para o envio do formulário

```

```
document.getElementById('contactForm').addEventListener('submit',
function(event) {

    event.preventDefault(); // Impede o envio padrão do formulário

    // Valida o e-mail antes de enviar

    if (validateEmail()) {

        // Simulação de envio do formulário

        alert('Formulário enviado com sucesso!');

        this.reset(); // Limpa o formulário

        resetFeedback(); // Reseta os feedbacks visuais

    }

});
```

```
// Validação em tempo real enquanto o usuário digita

document.getElementById('email').addEventListener('input', function() {

    if (this.value.trim() === "") {

        resetFeedback(); // Se campo vazio, reseta feedbacks

    } else {

        validateEmail(); // Caso contrário, valida o e-mail

    }

});
```

```
// Função principal de validação de e-mail

function validateEmail() {

    // Obtém elementos DOM necessários

    const emailInput = document.getElementById('email');
```



```

const emailError = document.getElementById('emailError');
const emailInfo = document.getElementById('emailInfo');
const emailSuccess = document.getElementById('emailSuccess');

// Obtém e limpa o valor do input
const email = emailInput.value.trim();

// Expressão regular para validar formato de e-mail
// Valida: texto@texto.texto (sem espaços e com @ e .)
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

// Reseta todos os estados visuais antes de nova validação
emailInput.classList.remove('error', 'success');
emailError.style.display = 'none';
emailInfo.style.display = 'none';
emailSuccess.style.display = 'none';

// Validações:
if (email === "") {
    // Caso campo vazio
    emailInput.classList.add('error');
    emailError.querySelector('span').textContent = 'Por favor, insira um endereço de e-mail.';
    emailError.style.display = 'flex';
    return false;
} else if (!emailRegex.test(email)) {

```

```

    // Caso formato inválido

    emailInput.classList.add('error');

    emailError.querySelector('span').textContent = 'Formato de e-mail inválido.';

    emailError.style.display = 'flex';

    emailInfo.querySelector('span').textContent = 'Verifique se o e-mail segue o
formato nome@dominio.extensão';

    emailInfo.style.display = 'flex';

    return false;
  } else {
    // Caso válido

    emailInput.classList.add('success');

    emailSuccess.style.display = 'flex';

    return true;
  }
}

```

// Função para resetar todos os feedbacks visuais

```

function resetFeedback() {
  const emailInput = document.getElementById('email');
  const emailError = document.getElementById('emailError');
  const emailInfo = document.getElementById('emailInfo');
  const emailSuccess = document.getElementById('emailSuccess');

  // Remove classes e oculta mensagens

  emailInput.classList.remove('error', 'success');

  emailError.style.display = 'none';

```

```

        emailInfo.style.display = 'none';
        emailSuccess.style.display = 'none';
    }
</script>
</body>
</html>

```

```

// Resetar estados
emailInput.classList.remove('error', 'success');
emailError.style.display = 'none';
emailInfo.style.display = 'none';
emailSuccess.style.display = 'none';

if (email === '') {
    emailInput.classList.add('error');
    emailError.querySelector('span').textContent = 'Por favor, insira um endereço de e-mail.';
}

```

## ✉ Formulário de Contato

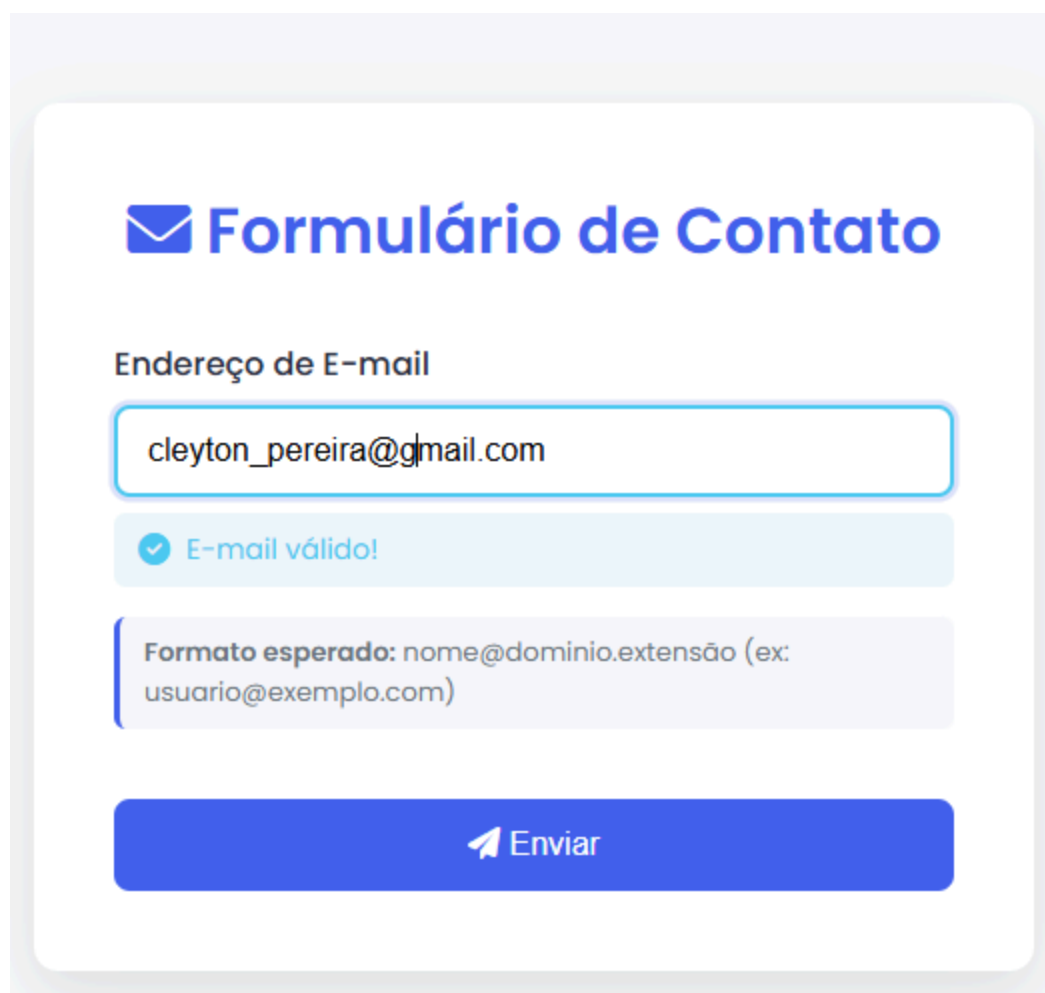
Endereço de E-mail

❗ Formato de e-mail inválido.

ℹ Verifique se o e-mail segue o formato  
nome@dominio.extensão

Formato esperado: nome@dominio.extensão (ex:  
usuario@exemplo.com)

➡ Enviar




 **Formulário de Contato**

Endereço de E-mail

✓ E-mail válido!

Formato esperado: nome@dominio.extensão (ex: usuario@exemplo.com)

 Enviar

## 6. Resultados Obtidos

Após a implementação do código, realizei testes com diferentes formatos de e-mail para verificar se a validação estava funcionando corretamente. Abaixo estão alguns exemplos dos testes realizados:

E-mail válido: usuario@dominio.com

Resultado: A mensagem "E-mail válido!" foi exibida em azul.

E-mail inválido (sem "@"): usuariodominio.com

Resultado: A mensagem "E-mail inválido. Verifique se o e-mail segue o formato (exemplo: usuario@dominio.com)." foi exibida em vermelho.

E-mail inválido (sem domínio): usuario@

Resultado: A mensagem de erro foi exibida corretamente.

E-mail inválido (com espaços): usuario @ dominio.com

Resultado: A mensagem de erro foi exibida corretamente.

O script funcionou conforme o esperado, identificando e-mails válidos e inválidos com base na expressão regular utilizada. Um desafio encontrado foi garantir que a expressão regular cobrisse a maioria dos formatos de e-mail válidos, como e-mails com subdomínios (exemplo: usuario@dominio.com.br). Após alguns ajustes, a expressão regular foi capaz de validar esses casos também.

## 6. CONCLUSÃO

A atividade foi muito proveitosa, pois permitiu colocar em prática conceitos importantes de JavaScript, como manipulação do DOM e uso de expressões regulares para validação de dados. Aprendi como criar um formulário simples e como validar entradas do usuário de forma eficiente, garantindo que os dados inseridos estejam no formato correto. Além disso, a experiência de testar diferentes cenários e ajustar o código para cobrir mais casos foi enriquecedora. O conhecimento adquirido será útil em projetos futuros, especialmente no desenvolvimento de formulários e validações de dados em aplicações web.

- **Observações Finais:**

Dificuldades Encontradas: A principal dificuldade foi ajustar a expressão regular para cobrir todos os formatos possíveis de e-mail, como e-mails com subdomínios ou caracteres especiais.

- **Melhorias Futuras:**

Uma possível melhoria seria adicionar mais validações, como verificar se o domínio do e-mail existe ou se o e-mail já foi cadastrado em um banco de dados.

- **Aprendizado:**

A atividade reforçou a importância da validação de dados no front-end para melhorar a experiência do usuário e evitar erros no back-end.

## REFERÊNCIAS

- MDN Web Docs. JavaScript Guide. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Acesso em: [data de acesso].
- W3Schools. JavaScript Form Validation. Disponível em: [https://www.w3schools.com/js/js\\_validation.asp](https://www.w3schools.com/js/js_validation.asp). Acesso em: [data de acesso].
- [Playcode.io](https://playcode.io). Online JavaScript Editor. Disponível em: <https://playcode.io>. Acesso em: [data de acesso].
- Google Fonts. Poppins Typeface. Disponível em: <https://fonts.google.com/specimen/Poppins>. Acesso em: [data de acesso].
- Font Awesome. Icon Library. Disponível em: <https://fontawesome.com>. Acesso em: [data de acesso].
- ECMAScript 2023 Language Specification. Regular Expressions. Disponível em: <https://262.ecma-international.org>. Acesso em: [data de acesso].
- 
- W3C. HTML5 Specification. Disponível em: <https://www.w3.org/TR/html52/>. Acesso em: [data de acesso].