



# Aprofundando em JavaScript

**N**esta aula, vamos aprofundar nossos conhecimentos em JavaScript, focando na manipulação de eventos e na validação de elementos em páginas web. Entender como os eventos funcionam é essencial para criar interfaces dinâmicas e responsivas, permitindo que a página reaja às ações do usuário. Além disso, a validação de dados é crucial para garantir que as informações inseridas estejam corretas antes de serem processadas. Ao dominar esses conceitos, você estará apto a desenvolver aplicações web mais interativas, seguras e eficientes.

## Definição de Eventos

Eventos em JavaScript são elementos centrais na criação de interatividade e dinamismo em páginas web. Eles são desencadeados por ações do usuário ou pelo navegador e permitem que o desenvolvedor responda a essas ações com códigos específicos. A compreensão e manipulação de eventos é fundamental para criar experiências de usuário envolventes e funcionais.

Um evento pode ser algo simples como um clique em um botão, a movimentação do mouse sobre um elemento, a rolagem da página, ou até o carregamento completo do conteúdo de uma página. Quando um evento é acionado, ele pode disparar uma função em JavaScript que executa uma ação pré-determinada.

Por exemplo, imagine uma página onde há um botão que, ao ser clicado, exibe uma mensagem de boas-vindas. Esse comportamento é possível graças a um evento de clique associado ao botão. O código JavaScript poderia ser algo como:

```
```javascript
```

```
document.getElementById("meuBotao").addEventListener("click", function() {  
    alert("Bem-vindo à nossa página!");  
});  
```
```




Neste exemplo, utilizamos o método `addEventListener` para “escutar” o evento de clique no botão com o ID “meuBotao”. Quando o usuário clica no botão, a função anônima dentro do `addEventListener` é executada, mostrando uma mensagem de alerta.

Além de cliques, eventos podem ser aplicados a teclas pressionadas, movimento do mouse, mudança de valor em um formulário, entre muitos outros. A lista de eventos possíveis é extensa, permitindo que o desenvolvedor controle praticamente qualquer interação que o usuário tenha com a página.

Outro exemplo prático seria o uso de eventos de formulário. Suponha que você tenha um formulário de cadastro e queira validar os dados do usuário antes de enviar as informações ao servidor. Você poderia utilizar o evento `blur`, que é disparado quando o usuário sai de um campo de texto. Veja um exemplo de código:

```
```javascript
```

```
document.getElementById("email").addEventListener("blur", function() {  
    var email = document.getElementById("email").value;  
    if (!email.includes("@")) {  
        alert("Por favor, insira um endereço de e-mail válido.");  
    }  
});  
```
```

Neste caso, o código verifica se o campo de e-mail contém o caractere “@” quando o usuário sai do campo. Se não contiver, uma mensagem de alerta é exibida, impedindo o usuário de enviar um e-mail inválido. 

Portanto, a manipulação de eventos é uma ferramenta poderosa em JavaScript que permite ao desenvolvedor criar páginas altamente interativas, controlando tudo, desde simples cliques até interações mais complexas, como validação de formulários e controle de navegação.

## Utilizando Eventos em JavaScript

Uma vez que entendemos o que são eventos em JavaScript, o próximo passo é aprender a utilizá-los de maneira eficaz para manipular elementos do DOM (Document Object Model) e criar interações dinâmicas na página. Para isso, o método `addEventListener` é amplamente utilizado.

O `addEventListener` é uma função que “escuta” um determinado evento em um elemento da página e executa uma função quando esse evento ocorre. A flexibilidade do `addEventListener` permite que seja usado em uma variedade de situações, como cliques de botão, movimentação do mouse, eventos de teclado, entre outros.

Vamos aprofundar um exemplo prático em que utilizamos `addEventListener` para adicionar uma nova linha em uma tabela sempre que um botão de submissão for clicado. Suponha que temos um formulário de cadastro de produtos, onde o usuário insere o código do produto, a descrição, o preço e a quantidade. Ao clicar em “Cadastrar”, uma nova linha deve ser inserida na tabela de produtos.

Aqui está como isso pode ser feito:

```
```javascript
```

```
document.getElementById("cadastrar").addEventListener("click", function() {  
    // Captura os valores inseridos nos campos do formulário  
    const codigoProduto = document.getElementById("codigoProduto").value;  
    const descricaoProduto = document.getElementById("descricaoProduto").value;  
    const precoProduto =  
parseFloat(document.getElementById("precoProduto").value).toFixed(2);  
    const quantidadeProduto =  
document.getElementById("quantidadeProduto").value;  
  
    // Acessa a tabela onde os produtos serão inseridos  
    const tabela =  
document.getElementById("tabelaProdutos").getElementsByTagName("tbody")[0];  
  
    // Cria uma nova linha na tabela  
    const novaLinha = tabela.insertRow();  
  
    // Adiciona células à nova linha com os valores inseridos  
    novaLinha.insertCell(0).textContent = codigoProduto;  
    novaLinha.insertCell(1).textContent = descricaoProduto;  
    novaLinha.insertCell(2).textContent = `R$ ${precoProduto}`;  
    novaLinha.insertCell(3).textContent = quantidadeProduto;  
    // Reseta os campos do formulário  
    document.getElementById("formularioProduto").reset();  
});  
```
```



Nesse exemplo, utilizamos o `getElementById` para capturar os valores dos campos de entrada do formulário. Depois, criamos uma nova linha na tabela com o método `insertRow`, e, em seguida, inserimos células nessa linha com o método `insertCell`, preenchendo-as com os valores capturados. Após a inserção da nova linha, o formulário é resetado para permitir novas inserções.

Um aspecto importante desse código é o uso do método `parseFloat` e `toFixed` para formatar o preço como um valor decimal com duas casas, garantindo que seja exibido corretamente na tabela. Isso demonstra como JavaScript pode ser utilizado

para realizar manipulações mais complexas dos dados, assegurando que sejam apresentados de forma consistente e apropriada.



Esse exemplo ilustra como eventos podem ser usados para capturar interações do usuário e manipular o DOM em tempo real, criando interfaces dinâmicas e responsivas. Ao dominar o uso de eventos, o desenvolvedor ganha o controle total sobre a interação do usuário com a página, podendo implementar funcionalidades avançadas e experiências de usuário sofisticadas.

## Validação de Elementos

A validação de elementos é uma prática crucial em qualquer aplicação web, pois garante que os dados inseridos pelos usuários estejam corretos antes de serem enviados ao servidor. A validação não apenas melhora a experiência do usuário ao evitar erros, mas também aumenta a segurança da aplicação, prevenindo o envio de dados inválidos ou maliciosos.

Em JavaScript, a validação pode ser realizada de várias maneiras, desde verificações simples, como se um campo está vazio, até validações mais complexas, como comparar datas ou números. Um exemplo simples de validação é garantir que um campo obrigatório seja preenchido:

```
```\javascript
document.getElementById("enviar").addEventListener("click", function(event) {
    const nome = document.getElementById("nome").value;
    if (nome === "") {
        alert("O campo nome é obrigatório.");
        event.preventDefault(); // Impede o envio do formulário
    }
});
...`
```

Neste código, utilizamos `addEventListener` para capturar o evento de clique no botão de envio do formulário. Se o campo “nome” estiver vazio, uma mensagem de alerta é exibida, e a função `preventDefault()` é chamada para impedir o envio do formulário até que o campo seja preenchido.


A validação de números também é comum, especialmente em campos que exigem que o usuário insira valores monetários ou quantidades. Veja um exemplo onde validamos se o preço de um produto é maior que zero:

```
```javascript
document.getElementById("enviar").addEventListener("click", function(event) {
  const preco = parseFloat(document.getElementById("preco").value);
  if (isNaN(preco) || preco <= 0) {
    alert("Por favor, insira um valor válido para o preço.");
    event.preventDefault();
  }
});
```
```

Aqui, utilizamos `parseFloat` para converter o valor inserido em um número decimal e verificamos se ele é maior que zero. Se não for, ou se o valor não for um número (`isNaN`), o formulário não é enviado, e o usuário é alertado para corrigir o erro.

Outra forma comum de validação é a de datas, onde podemos garantir que a data de nascimento inserida seja válida e não uma data futura:

```
````javascript
document.getElementById("enviar").addEventListener("click", function(event) {
    const dataNascimento = new
Date(document.getElementById("dataNascimento").value);
    const hoje = new Date();
    if (dataNascimento >= hoje) {
        alert("A data de nascimento não pode ser no futuro.");
        event.preventDefault();
    }
});
````
```



Neste exemplo, comparamos a data de nascimento inserida com a data atual, e se a data inserida for igual ou maior que a data de hoje, o formulário não é enviado. Essa validação impede que datas inválidas sejam submetidas, garantindo a consistência dos dados.

A validação em JavaScript, portanto, é uma técnica poderosa que permite garantir a integridade dos dados em tempo real, antes que eles cheguem ao servidor. Isso não apenas melhora a usabilidade da aplicação, mas também oferece uma camada adicional de segurança contra entradas inválidas ou potencialmente perigosas.

## Utilizando a Validação de Elementos

Aplicar validação de elementos em JavaScript envolve a implementação de verificações específicas no código para garantir que os dados inseridos pelo usuário atendem aos critérios esperados. Isso pode incluir a validação de campos obrigatórios, o controle de comprimento de entradas de texto e a comparação entre valores. Esses mecanismos são essenciais para evitar erros e garantir a consistência dos dados.

Para validar um campo obrigatório, por exemplo, podemos usar uma estrutura simples de `if` para verificar se o campo foi preenchido:

```
````javascript
document.getElementById("enviar").addEventListener("click", function(event) {
    const campoObrigatorio = document.getElementById("campoObrigatorio").value;
    if (campoObrigatorio === "") {
        alert("Este campo é obrigatório.");
        event.preventDefault();
    }
});
````
```



Neste código, a função `preventDefault()` é utilizada para impedir que o formulário seja enviado caso o campo obrigatório esteja vazio, garantindo que o usuário preencha todos os campos necessários antes de prosseguir.

Além da validação de preenchimento, a validação de tamanho é outra técnica importante. Ela garante que o conteúdo de um campo não exceda ou seja inferior a um determinado número de caracteres. Por exemplo, ao validar um código de produto:

```
````javascript
document.getElementById("enviar").addEventListener("click", function(event) {
    const codigoProduto = document.getElementById("codigoProduto").value;
    if (codigoProduto.length < 5 || codigoProduto.length > 10) {
        alert("O código do produto deve ter entre 5 e 10 caracteres.");
        event.preventDefault();
    }
});
````
```

Aqui, a validação assegura que o código do produto tenha um comprimento aceitável, evitando erros de entrada que poderiam comprometer o processamento posterior.



A comparação entre valores também é uma prática comum em validação. Um exemplo clássico é a validação de senhas:



```
```\njavascript\n\ndocument.getElementById("enviar").addEventListener("click", function(event) {\n  const senha = document.getElementById("senha").value;\n  const confirmacaoSenha = document.getElementById("confirmacaoSenha").value;\n  if (senha !== confirmacaoSenha) {\n    alert("As senhas não coincidem.");\n    event.preventDefault();\n  }\n});\n```\n
```

Neste caso, o código compara o valor da senha com o da confirmação de senha. Se os valores não coincidirem, o formulário não é enviado, e o usuário é avisado do erro.

Um exemplo mais avançado envolve a validação de preços e quantidades, onde não apenas validamos os valores inseridos, como também controlamos o fluxo do código baseado nessa validação:

```
```javascript
```

```
document.getElementById("enviar").addEventListener("click", function(event) {  
    const preco = parseFloat(document.getElementById("preco").value);  
    const quantidade = parseInt(document.getElementById("quantidade").value);  
  
    if (preco <= 0) {  
        alert("O preço deve ser maior que zero.");  
        event.preventDefault();  
        return;  
    }  
  
    if (quantidade < 0) {  
        alert("A quantidade não pode ser negativa.");  
        event.preventDefault();  
        return;  
    }  
  
    if (quantidade === 0 && !confirm("Você está adicionando um produto sem  
estoque. Deseja continuar?")) {  
        event.preventDefault();  
        return;  
    }  
});  
```
```



Nesse código, utilizamos diferentes validações: primeiro, verificamos se o preço é maior que zero; depois, se a quantidade é positiva ou igual a zero. Se a quantidade for zero, solicitamos uma confirmação do usuário para prosseguir. Essa abordagem não apenas valida os dados, mas também interage com o usuário para garantir que as informações sejam precisas.

Com essas validações implementadas, podemos garantir que os dados inseridos pelo usuário sejam consistentes e válidos, prevenindo problemas futuros na aplicação. A validação de elementos em JavaScript é, portanto, uma técnica essencial para a criação de aplicações web robustas e seguras.

## Conteúdo Bônus



Um excelente conteúdo complementar gratuito pode ser encontrado na MDN Web Docs (Mozilla Developer Network). A MDN é uma referência respeitada para desenvolvedores web e oferece uma vasta gama de tutoriais, guias e documentações.

Aqui estão dois artigos que podem ser muito úteis:

### 1. MDN Web Docs - JavaScript Guide:

Este guia cobre desde conceitos básicos até tópicos avançados em JavaScript, incluindo manipulação de eventos e validação de formulários.

### 2. MDN Web Docs - Form data validation:

Esta página é dedicada à validação de formulários em JavaScript, oferecendo exemplos práticos e explicações detalhadas.

## Referência Bibliográfica

ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. Fundamentos da programação de computadores. 3.ed. Pearson: 2012.

BRAGA, P. H. Teste de software. Pearson: 2016.

GALLOTTI, G. M. A. (Org.). Arquitetura de software. Pearson: 2017.

GALLOTTI, G. M. A. Qualidade de software. Pearson: 2015.

MEDEIROS, E. Desenvolvendo software com UML 2.0 definitivo. Pearson: 2004.

PFLEEGER, S. L. Engenharia de software: teoria e prática. 2.ed. Pearson: 2003.



## **Prática Integradora Desenvolvimento de Software**

### **Atividade Prática 13 – Interatividade e Validação de Formulários com JavaScript**

#### **Objetivos:**

- Aplicar eventos em JavaScript para criar interações dinâmicas em uma página web.
- Implementar validação de dados em formulários para garantir a entrada correta de informações.
- Desenvolver habilidades para manipular o DOM com base nas interações do usuário.


#### **Materiais, Métodos e Ferramentas:**

- Navegador web com ferramentas de desenvolvimento (como Google Chrome, Firefox, etc.).
- Editor de texto ou IDE (como VS Code, Sublime Text, etc.).
- Arquivo HTML e JavaScript para implementação da atividade.

#### **Atividade Prática**

Primeiramente, leia atentamente o texto a seguir:

Você foi contratado para melhorar a funcionalidade de um formulário de cadastro de produtos em um sistema de gerenciamento de inventário. A

página já possui um formulário com campos para código do produto, descrição, preço e quantidade. Além disso, há uma tabela onde os produtos cadastrados serão exibidos. 

### **Enunciado da Atividade:**

- **Criação do Formulário e da Tabela:**


- Crie uma página HTML com um formulário contendo os seguintes campos: Código do Produto, Descrição, Preço e Quantidade.
- Adicione um botão “Cadastrar” que, quando clicado, deve adicionar uma nova linha na tabela de produtos com os dados preenchidos no formulário.
- A tabela deve ter cabeçalhos correspondentes aos campos do formulário.

- **Manipulação de Eventos:**

- Utilize JavaScript para escutar o evento de clique no botão “Cadastrar”.
- Ao clicar no botão, os dados do formulário devem ser capturados e uma nova linha deve ser inserida na tabela com esses dados.
- Após adicionar a linha, o formulário deve ser resetado.

- **Validação dos Dados:**

- Implemente validação para os seguintes casos:

- O campo “Código do Produto” deve ter entre 5 e 10 caracteres. 
- O campo “Preço” deve ser um número maior que zero.
- O campo “Quantidade” não pode ser negativa.

- Se algum dos campos não atender aos critérios, exiba uma mensagem de alerta e impeça o envio dos dados para a tabela.

Agora, vamos praticar!

### **Passo a Passo Detalhado da Atividade:**

- **HTML e Estrutura da Página:**

- Crie um arquivo HTML com a estrutura básica e adicione um formulário e uma tabela. Utilize o seguinte código como exemplo:

```
<!DOCTYPE html>
```

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

```
<head>
```

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

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

```
<title>Cadastro de Produtos</title>
```

```
</head>
```

```
<body>
```

<form id="formularioProduto">



<label for="codigoProduto">Código do Produto:</label>

<input type="text" id="codigoProduto" name="codigoProduto">

<br>

<label for="descricaoProduto">Descrição:</label>

<input type="text" id="descricaoProduto" name="descricaoProduto">

<br>

<label for="precoProduto">Preço:</label>

<input type="text" id="precoProduto" name="precoProduto">

<br>

<label for="quantidadeProduto">Quantidade:</label>

<input type="text" id="quantidadeProduto"  
name="quantidadeProduto">

<br>

<button type="button" id="cadastrar">Cadastrar</button>

</form>

<table id="tabelaProdutos" border="1">

<thead>

<tr>

<th>Código</th>



<th>Descrição</th>

<th>Preço</th>

<th>Quantidade</th>

</tr>

</thead>

<tbody>

<!-- Linhas da tabela serão adicionadas aqui -->

</tbody>

</table>

<script src="script.js"></script>

</body>

</html>

- **JavaScript para Manipulação e Validação:**

- Crie um arquivo script.js e implemente o código JavaScript para manipulação de eventos e validação de dados conforme os requisitos. Utilize o exemplo de código a seguir:

```
document.getElementById("cadastrar").addEventListener("click", function() {
```

```
// Captura os valores dos campos
```



```
const      codigoProduto      =  
document.getElementById("codigoProduto").value;  
  
const      descricaoProduto    =  
document.getElementById("descricaoProduto").value;  
  
const      precoProduto        =  
parseFloat(document.getElementById("precoProduto").value);  
  
const      quantidadeProduto    =  
parseInt(document.getElementById("quantidadeProduto").value);  
  
// Valida os dados  
  
if (codigoProduto.length < 5 || codigoProduto.length > 10) {  
  
    alert("O código do produto deve ter entre 5 e 10 caracteres.");  
  
    return;  
  
}  
  
if (isNaN(precoProduto) || precoProduto <= 0) {  
  
    alert("O preço deve ser um número maior que zero.");  
  
    return;  
  
}  
  
if (isNaN(quantidadeProduto) || quantidadeProduto < 0) {  
  
    alert("A quantidade não pode ser negativa.");  
  
    return;
```

```
}
```



```
// Acessa a tabela e cria uma nova linha
```

```
const tabela =  
document.getElementById("tabelaProdutos").getElementsByTagName("tbody")[0];
```

```
const novaLinha = tabela.insertRow();
```

```
// Adiciona células à nova linha
```

```
novaLinha.insertCell(0).textContent = codigoProduto;
```

```
novaLinha.insertCell(1).textContent = descricaoProduto;
```

```
novaLinha.insertCell(2).textContent = R$ ${precoProduto.toFixed(2)};
```

```
novaLinha.insertCell(3).textContent = quantidadeProduto;
```


```
// Reseta os campos do formulário
```

```
document.getElementById("formularioProduto").reset();
```

```
});
```

## Gabarito Esperado

- **HTML:** a estrutura HTML deve conter um formulário com campos para código do produto, descrição, preço e quantidade, e uma tabela para exibir os produtos.
- **JavaScript (Eventos):** o botão "Cadastrar" deve acionar a função que captura os valores dos campos e adiciona uma nova linha na tabela.

- **JavaScript (Validação):** as validações devem garantir que o código do produto tenha entre 5 e 10 caracteres, o preço seja um número maior c,  zero, e a quantidade não seja negativa. Mensagens de alerta devem ser exibidas e o envio dos dados para a tabela deve ser impedido se as validações falharem.

Essa estrutura cobre a criação de uma página interativa e validação de dados, abordando tanto a manipulação do DOM quanto a interação com o usuário.

**[Ir para exercício](#)**