



Praticando uso do DOM

Resumo


Eventos no JavaScript são ações ou ocorrências que ocorrem durante a interação com uma página web. Eles podem ser desencadeados por ações do usuário, como cliques e movimentos do mouse, ou por ações do navegador, como carregamento de página. Ao associar manipuladores de eventos a elementos HTML, podemos executar código específico quando esses eventos ocorrem. Os eventos permitem interatividade e dinamismo nas aplicações web, permitindo que os desenvolvedores respondam e tomem ações com base nas interações do usuário e nas mudanças no estado da página.

Eventos

No contexto do JavaScript, eventos são ações ou ocorrências que podem ser detectadas e manipuladas pelo código. Esses eventos podem ser desencadeados por interações do usuário, como cliques em botões, movimentos do mouse, digitação de teclado, entre outros. Além disso, eventos também podem ocorrer como resultado de ações do navegador, como carregamento de página, redimensionamento da janela, tempo decorrido, etc.

Para utilizar eventos no JavaScript, você pode seguir os seguintes passos:



1. Selecionar um elemento HTML: Primeiro, você precisa selecionar o elemento HTML ao qual deseja associar um to. Isso pode ser feito usando métodos como `getElementById`, `getElementsByClassName`, `getElementsByTagName`, `querySelector`, etc.

2. Associar um manipulador de eventos: Em seguida, você precisa associar um manipulador de eventos ao elemento selecionado. Um manipulador de eventos é uma função que será executada quando o evento ocorrer. Existem algumas maneiras de fazer isso:

- Atribuição direta: Você pode atribuir uma função diretamente a uma propriedade de evento do elemento. Por exemplo, para associar um manipulador de eventos de clique a um botão com o id “myButton”, você pode fazer o seguinte:

```
'''javascript
const myButton = document.getElementById("myButton");
myButton.onclick = function() {
    // Código a ser executado quando o botão for clicado
};
'''
```

- Método `addEventListener`: Uma forma mais flexível de associar eventos é usar o método `addEventListener`. Isso permite adicionar vários manipuladores de eventos para o mesmo tipo de evento em um elemento. Por exemplo:

```
'''javascript
const myButton = document.getElementById("myButton");
myButton.addEventListener("click", function() {
    // Código a ser executado quando o botão for clicado
});
'''
```

- Manipular o evento: Dentro do manipulador de eventos, você pode escrever o código que será executado quando o evento

ocorrer. Isso pode incluir ações como modificar o conteúdo HTML, alterar estilos, fazer solicitações AJAX, validar dados de entrada, entre outras tarefas.

É importante ressaltar que existem muitos tipos de eventos disponíveis, como `click`, `mouseover`, `keydown`, `submit`, `load`, `resize`, entre outros. Além disso, você também pode criar seus próprios eventos personalizados usando a API `CustomEvent`.

Por exemplo, para lidar com um evento de clique em um botão, você pode fazer algo como:

```
```html
<button id="myButton">Clique aqui</button>

<script>
const myButton = document.getElementById("myButton");
myButton.addEventListener("click", function() {
 console.log("O botão foi clicado!");
});
</script>
```
```

Neste exemplo, quando o botão for clicado, a mensagem “O botão foi clicado!” será exibida no console.

Conteúdo Bônus

Javascript possui uma gama de eventos que podem ser utilizados para acionar diversos comportamentos em suas aplicações. Dessa forma, é importante conhecermos quais tipos de eventos podemos utilizar.

Existem vários tipos de eventos que podem ocorrer em uma página web. Aqui estão alguns exemplos dos eventos mais comuns:

1. Eventos de Mouse:

- `click`: Acionado quando um elemento é clicado.



- `mouseover`: Acionado quando o cursor do mouse entra em um elemento.
- `mouseout`: Acionado quando o cursor do mouse sai de um elemento.
- `mousemove`: Acionado quando o cursor do mouse se move dentro de um elemento.

2. Eventos de Teclado:


- `keydown`: Acionado quando uma tecla é pressionada.
- `keyup`: Acionado quando uma tecla é liberada.
- `keypress`: Acionado quando uma tecla é pressionada e solta.

3. Eventos de Formulário:

- `submit`: Acionado quando um formulário é enviado.
- `input`: Acionado quando o valor de um elemento de input é alterado.
- `focus`: Acionado quando um elemento recebe foco.
- `blur`: Acionado quando um elemento perde o foco.

4. Eventos de Tempo:

- `load`: Acionado quando uma página ou recurso é carregado.

- unload: Acionado quando uma página é fechada ou navegada para fora. 
- resize: Acionado quando a janela do navegador é redimensionada.
- scroll: Acionado quando a barra de rolagem é movida.

5. Eventos de Touch:

- touchstart: Acionado quando um toque é iniciado em um elemento touch.
- touchend: Acionado quando um toque é finalizado em um elemento touch.
- touchmove: Acionado quando um toque se move em um elemento touch.
- touchcancel: Acionado quando um toque é cancelado.

Manipulação de Eventos:

Existem duas maneiras comuns de lidar com eventos no JavaScript:

1. Atribuição direta de manipuladores de eventos:

- Nesse método, você atribui uma função diretamente a uma propriedade de evento do elemento, como onclick, onmouseover, etc.
- Por exemplo:



```
'''javascript
const myButton = document.getElementById("myButton");
myButton.onclick = function() {
  console.log("O botão foi clicado!");
};
'''
```

2. Método addEventListener:

- Com esse método, você pode adicionar um ouvinte de evento a um elemento. Ele permite adicionar vários manipuladores de eventos para o mesmo tipo de evento em um elemento.

- Por exemplo:

```
'''javascript
const myButton = document.getElementById("myButton");
myButton.addEventListener("click", function() {
  console.log("O botão foi clicado!");
});
'''
```

Remoção de Eventos:

Você também pode remover eventos que foram previamente adicionados a um elemento usando o método `removeEventListener`. Isso é útil quando você não precisa mais do manipulador de eventos associado a um elemento.

- Por exemplo:

```
'''javascript
const myButton = document.getElementById("myButton");

function handleClick() {
  console.log("O botão foi clicado!");
}

myButton.addEventListener("click", handleClick);
```

Referência Bibliográfica

FLANAGAN, David. JavaScript: O Guia Definitivo. 6ª Ed. Porto Alegre: Bookman, 2013.



FREEMAN, Eric. Use a cabeça!: programação JavaScript. 1ª Ed. São Paulo: Alta Books, 2016.

ATIVIDADE PRÁTICA

Título da Prática: Classe, HTML e DOM

Objetivos: Compreender o uso do Javascript com objetos e elementos do DOM

Materiais, Métodos e Ferramentas: Para realizar esta prática vamos utilizar o Visual Studio Code

Prática

Você foi contratado para desenvolver uma funcionalidade em JavaScript para uma loja virtual. A funcionalidade consiste em filtrar uma lista de objetos com base na categoria informada pelo usuário e exibir os objetos correspondentes em uma lista.

1. Crie uma classe chamada Objeto que tenha as seguintes propriedades:

- nome (string): o nome do objeto.
- descricao (string): uma breve descrição do objeto.
- categoria (string): a categoria à qual o objeto pertence.
- preco (number): o preço do objeto.

2. Crie uma lista de objetos com os seguintes itens:



- Objeto 1: nome - "Samsung A71", descrição - "Celular para games", categoria - "Eletrônico", preço - 1500.
- Objeto 2: nome - "Samsung TAB G21", descrição - "Tablet para estudo", categoria - "Eletrônico", preço - 2100.
- Objeto 3: nome - "Fogão 4 bocas", descrição - "Fogão para alimentação saudável", categoria - "Eletrodoméstico", preço - 1600.
- Objeto 4: nome - "Geladeira e Freezer", descrição - "Congela rápido", categoria - "Eletrodoméstico", preço - 4000.
- Objeto 5: nome - "Air Fryer", descrição - "Agiliza seu dia", categoria - "Eletrodoméstico", preço - 950.

3. Crie uma função chamada `filtrarPorCategoria`, que será chamada quando o usuário clicar no botão "Pesquisar". Essa função deve realizar as seguintes ações:

- Obter o valor digitado pelo usuário em um campo de texto com o id "categoricalInput".
- Limpar a lista de resultados anterior.
- Filtrar a lista de objetos com base na categoria informada usando o método `filter`.
- Se não houver objetos correspondentes, exibir a mensagem "Nenhum objeto encontrado para a categoria informada." na lista de resultados.
- Para cada objeto correspondente, exibir as seguintes informações na lista de resultados:

- Nome do objeto.
- Descrição do objeto.
- Categoria do objeto.
- Preço do objeto.
- Teste a funcionalidade fornecendo diferentes categorias e verifique se os objetos correspondentes são exibidos corretamente na lista de resultados.



Resolução

Arquivo HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Lista de Objetos</title>
  <script src="scriptcategoria.js"></script>
</head>
<body>
  <h1>Pesquisa por Categoria</h1>
  <input type="text" id="categoriaInput" placeholder="Digite a categoria">
  <button onclick="filtrarPorCategoria()">Pesquisar</button>
  <ul id="resultado"></ul>
</body>
</html>
```

Arquivo JavaScript



```
class Objeto {
  constructor(nome, descricao, categoria, preco) {
    this.nome = nome;
    this.descricao = descricao;
    this.categoria = categoria;
    this.preco = preco;
  }
}

const listaObjetos = [
  new Objeto('Sansung A71', 'Celular para games', 'Eletrônico', 1500),
  new Objeto('Sansung TAB G21', 'Tablet para estudo', 'Eletrônico', 2100),
  new Objeto('Fogão 4 bocas', 'Fogão para alimentação saudável', 'Eletrônico', 1600),
  new Objeto('Geladeira e Freezer', 'Congela rápido', 'Eletrodoméstico', 4000),
  new Objeto('Air Fryer', 'Agiliza seu dia', 'Eletrodoméstico', 950)
];

function filtrarPorCategoria() {
  const categorialInput = document.getElementById('categorialInput').value;
  const resultado = document.getElementById('resultado');
  resultado.innerHTML = ""; // Limpar a lista de resultados anterior

  const objetosFiltrados = listaObjetos.filter(objeto => objeto.categoria === categorialInput);

  if (objetosFiltrados.length === 0) {
    resultado.innerHTML = 'Nenhum objeto encontrado para a categoria informada.';
    return;
  }

  objetosFiltrados.forEach(objeto => {
```

```
const li = document.createElement('li');
li.innerHTML = `
  <strong>Nome:</strong> ${objeto.nome}<br>
  <strong>Descrição:</strong> ${objeto.descricao}<br>
  <strong>Categoria:</strong> ${objeto.categoria}<br>
  <strong>Preço:</strong> ${objeto.preco}<br><br>
`;
resultado.appendChild(li);
});
}
```



[Ir para exercício](#)