JavaScript: Fundamentos para Desenvolvimento Web Interativo - Turma 2025B

5.3 Tipos de eventos

Em JavaScript, existem diferentes tipos de eventos que podem ser acionados durante a interação do usuário com uma página da web. A seguir, estão listadas algumas das principais categorias de eventos JavaScript que você pode encontrar em uma página web.

Eventos de mouse

Estes eventos são disparados por ações relacionadas ao mouse do usuário. Abaixo estão alguns dos principais eventos de mouse em JavaScript:

- **click:** Este evento é acionado quando um usuário clica em um elemento.
- **dblclick:** Este evento é acionado quando um usuário clica duas vezes rapidamente em um elemento.
- **mousedown:** Este evento é acionado quando o botão do mouse é pressionado sobre um elemento.
- **mouseup:** Este evento é acionado quando o botão do mouse é liberado após ter sido pressionado sobre um elemento.
- **mousemove:** Este evento é acionado quando o cursor do mouse se move sobre um elemento.
- **mouseover:** Este evento é acionado quando o cursor do mouse entra na área de um elemento.
- **mouseout:** Este evento é acionado quando o cursor do mouse deixa a área de um elemento.
- **mouseenter:** Este evento é acionado quando o cursor do mouse entra na área de um elemento, mas não se propaga aos elementos descendentes.

- **mouseleave:** Este evento é acionado quando o cursor do mouse deixa a área de um elemento, mas não se propaga aos elementos descendentes.

Veja no quadro abaixo um exemplo mostrando como podemos manipular eventos de mouse usando JavaScript:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo de Eventos de Mouse</title>
  <style>
    #box {
      width: 200px;
      height: 200px;
      background-color: teal;
      color: white;
      display: flex;
      justify-content: center;
      align-items: center;
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <button id="clickButton">Clique em mim</button>
  <div id="box">Passe o mouse aqui</div>
  <script>
    document.addEventListener('DOMContentLoaded', function () {
      const button = document.getElementById('clickButton');
      const box = document.getElementById('box');

      // Evento de clique
      button.addEventListener('click', function () {
        alert('Botão clicado!');
      });
```

```
      // Evento de mouseover
      box.addEventListener('mouseover', function () {
        box.style.backgroundColor = 'purple';
        box.textContent = 'Mouse está aqui!';
      });

      // Evento de mouseout
      box.addEventListener('mouseout', function () {
        box.style.backgroundColor = 'teal';
        box.textContent = 'Passe o mouse aqui';
      });
    });
  </script>
</body>
</html>
```

Observe que:

- Usamos **document.addEventListener('DOMContentLoaded', ...)** para garantir que o script só execute depois que todo o DOM estiver completamente carregado.
- Ao clicar no botão, um alerta é mostrado. Isso demonstra o básico do evento **click**.
- Quando o mouse entra na área do **div#box** (**mouseover**) a cor de fundo muda e o texto é atualizado.
- Quando o mouse sai da área do **div#box** (**mouseout**) as propriedades originais são restauradas.

Veja o resultado no navegador, conforme figura abaixo:

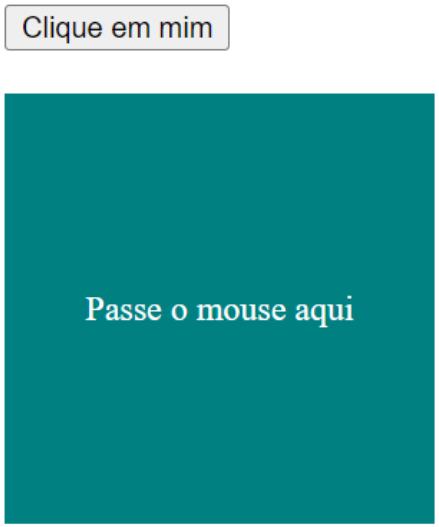

Figura 1 - Exemplo de execução de eventos de mouse

Descrição da imagem: No navegador, observamos um botão com o texto "Clique em mim" e uma div com largura e altura de 200 pixels, cor de fundo azul-petróleo (teal) e texto centralizado "Passe o mouse aqui". Quando o botão é clicado, uma mensagem de alerta aparece com o texto "Botão clicado!". Quando o usuário passa o mouse sobre a caixa, a cor de fundo muda para roxo e o texto dentro da caixa muda para "Mouse está aqui!". Mas, quando o usuário move o mouse para fora da caixa, a cor de fundo retorna ao azul-petróleo e o texto volta a ser "Passe o mouse aqui".

Eventos de teclado

Estes eventos são disparados por interações com o teclado. Abaixo estão alguns dos principais eventos de teclado em JavaScript:

- **keydown:** Este evento é acionado quando uma tecla é pressionada.
- **keypress:** Este evento é acionado quando uma tecla é pressionada e liberada (este evento está depreciado).
- **keyup:** Este evento é acionado quando uma tecla é liberada após ter sido pressionada.

Vejamos um exemplo prático que demonstra como usar eventos de teclado em JavaScript para manipular ações quando certas teclas são pressionadas.

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo de Eventos de Teclado</title>
</head>
<body>
  <input type="text" id="inputBox" size="40" placeholder="Digite algo e pressione Enter">
  <p id="feedback"></p>
  <script>
    document.addEventListener('DOMContentLoaded', function () {
      const inputBox = document.getElementById('inputBox');
      const feedback = document.getElementById('feedback');

      inputBox.addEventListener('keydown', function (event) {
        if (event.key === 'Enter') {
          feedback.textContent = 'Você pressionou Enter!';
          event.preventDefault(); // Impede que o Enter cause outra ação, como enviar um formulário
        }
      });

      inputBox.addEventListener('keyup', function (event) {
        if (event.key === 'Escape') {
          this.value = ''; // Limpa o input quando o Escape é pressionado
          feedback.textContent = 'Input limpo com a tecla Escape.';
        }
      });
    });
  </script>
</body>
</html>
```

Observe que:

- Usamos **document.addEventListener('DOMContentLoaded', ...)** para garantir que o script só execute depois que todo o DOM estiver carregado.
- Quando o usuário pressiona uma tecla dentro do campo de entrada, o evento **keydown** é acionado. Verificamos se a tecla pressionada foi a tecla "**Enter**". Se for, atualizamos o texto do elemento **<p>** e evitamos que o navegador execute o comportamento padrão do **Enter**, que poderia ser enviar um formulário, por exemplo.

- Captura de evento de tecla para cima (**keyup**): Similarmente, quando uma tecla é liberada (**keyup**), verificamos se a tecla "**Escape**" foi a pressionada. Se for, o campo de entrada é limpo e uma mensagem é mostrada ao usuário.

Veja o resultado no navegador, conforme figura abaixo:

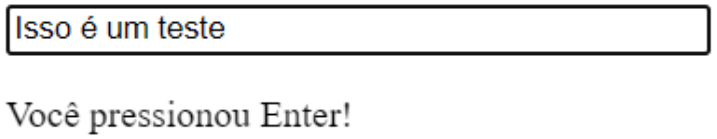

Figura 2 - Exemplo de execução de eventos do teclado

Descrição da imagem: No navegador, observamos uma caixa de entrada cujo placeholder contém o texto "Digite algo e pressione Enter". Quando o usuário digita algo e pressiona a tecla Enter, uma mensagem é exibida abaixo da caixa de entrada, dizendo "Você pressionou Enter!". Quando o usuário pressiona a tecla Escape, o texto dentro da caixa de entrada é apagado e uma mensagem é exibida abaixo da caixa de entrada, dizendo "Input limpo com a tecla Escape".

Eventos de formulário

Os eventos de formulário em JavaScript são eventos que ocorrem quando um formulário HTML é manipulado pelo usuário. Abaixo estão alguns dos principais eventos de formulário:

- **submit:** Este evento é acionado quando o usuário envia um formulário. Geralmente é usado para validar os dados do formulário antes de enviá-lo ao servidor.
- **change:** Este evento é acionado quando o valor de um elemento de formulário é alterado. Ele é comumente usado em campos de entrada (**input**), **select** e **textarea**.
- **input:** Este evento é acionado quando o valor de um elemento de formulário muda devido à entrada do usuário. Ele é semelhante ao evento change, mas é acionado imediatamente após a alteração, enquanto o change é acionado apenas quando o elemento perde o foco ou é enviado.
- **focus:** Este evento é acionado quando um elemento de formulário recebe o foco, ou seja, quando o usuário seleciona ou interage com ele.
- **blur:** Este evento é acionado quando um elemento de formulário perde o foco, ou seja, quando o usuário sai do campo de entrada após interagir com ele.

- **reset:** Este evento é acionado quando o usuário clica no botão de redefinição de um formulário. Geralmente é usado para limpar os campos do formulário e redefinir o estado inicial do formulário.

Vamos explorar um exemplo prático que demonstra como usar eventos de formulário para validar dados antes da submissão e manipular outros aspectos do formulário:

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo de Eventos de Formulário</title>
</head>
<body>
  <form id="meuFormulario">
    <label for="username">Nome de Usuário:</label>
    <input type="text" id="username" name="username" required>
    <br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <br>
    <button type="submit">Enviar</button>
  </form>
  <div id="mensagem"></div>
  <script>
    document.addEventListener('DOMContentLoaded', function () {
      const formulario = document.getElementById('meuFormulario');
      const mensagem = document.getElementById('mensagem');

      formulario.addEventListener('submit', function (event) {
        event.preventDefault(); // Impede a submissão do formulário para demonstração

        // Validação simples
        const username = formulario.querySelector('#username').value;
        const email = formulario.querySelector('#email').value;

        if (username.length < 4) {
          mensagem.textContent = 'O nome de usuário deve ter pelo menos 4 caracteres.';
          mensagem.style.color = 'red';
          return;
        }

        if (!email.includes('@')) {
          mensagem.textContent = 'Por favor, insira um email válido.';
          mensagem.style.color = 'red';
          return;
        }

        // Caso passe nas validações
```

```
        mensagem.textContent = 'Formulário válido, documento enviado!';
        mensagem.style.color = 'green';

        // Aqui você poderia realmente submeter o formulário
        // formulario.submit();
      });
    });
  </script>
</body>
</html>
```

Observe que:

- Usamos **document.addEventListener('DOMContentLoaded', ...)** para garantir que o script só execute depois que todo o DOM estiver carregado.
- Adicionamos um ouvinte ao evento **submit** do formulário. Este evento é ideal para fazer validações finais antes de enviar os dados ao servidor.
- Usamos **event.preventDefault()** para impedir que o formulário seja enviado imediatamente. Isso nos dá a chance de validar os dados manualmente.
- Verificamos se o nome de usuário tem pelo menos 4 caracteres e se o email contém um @. Se algum dos campos não atender aos requisitos, exibimos uma mensagem de erro. Se todos os campos forem validados com sucesso, exibimos uma mensagem indicando que o formulário é válido.

Veja o resultado no navegador, conforme figura abaixo:

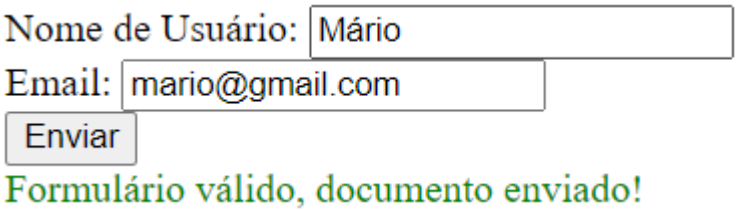

Figura 3 - Exemplo de execução de eventos de formulário

Descrição da imagem: No navegador, observamos que o formulário contém dois campos obrigatórios: um campo de texto para o nome de usuário e um campo de e-mail, ambos seguidos de um botão "Enviar". Ao tentar enviar o formulário, o sistema verifica se os dados inseridos são válidos. Se houver erros, mensagens de feedback em vermelho informarão quais correções são necessárias. Se os dados forem válidos, uma mensagem de confirmação em verde indicará que o formulário

foi enviado corretamente.

Eventos de UI (User Interface)

Os eventos de UI (User Interface) em JavaScript são eventos que ocorrem em elementos da interface do usuário, como janelas, barras de rolagem e menus. Esses eventos são úteis para lidar com interações específicas relacionadas à interface do usuário de uma aplicação web. Abaixo estão alguns dos principais eventos de UI em JavaScript:

- **load:** Este evento é acionado quando uma página ou um recurso é completamente carregado.
- **resize:** Este evento é acionado quando a janela do navegador é redimensionada pelo usuário.
- **scroll:** Este evento é acionado quando o usuário rola a página para cima ou para baixo.
- **focus:** Este evento é acionado quando um elemento da página recebe o foco, geralmente por meio de uma ação do usuário, como clicar em um campo de entrada.
- **blur:** Este evento é acionado quando um elemento da página perde o foco, ou seja, quando o usuário muda o foco para outro elemento.
- **select:** Este evento é acionado quando o usuário seleciona texto em um elemento de entrada de texto ou área de texto.

Vejamos no quadro abaixo um exemplo de uso destes eventos:

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplos de Eventos de UI</title>
</head>
<body>
  <div id="resizeInfo">Tamanho da janela: </div>
  <script>
    document.addEventListener('DOMContentLoaded', function () {
      // Evento load
      window.addEventListener('load', function () {
        console.log('Página completamente carregada!');
      });

      // Evento resize
      window.addEventListener('resize', function () {
        const resizeInfo = document.getElementById('resizeInfo');
        resizeInfo.textContent = 'Tamanho da janela: ' + window.innerWidth + ' x ' + window.innerHeight;
      });

      // Evento beforeunload
      window.addEventListener('beforeunload', function (event) {
        // Pode adicionar uma mensagem de confirmação, mas nem todos os navegadores suportam
        event.returnValue = 'Tem certeza que deseja sair?';
      });
    });
  </script>
</body>
</html>
```

Observe que:

- O evento **load** é disparado quando toda a página, incluindo todos os recursos dependentes (como imagens), é carregada. É útil para inicializações que requerem todos os recursos da página prontos.
- O evento **resize** é disparado quando a janela do navegador é redimensionada. No exemplo, atualizamos dinamicamente o conteúdo de um ** ** para mostrar as novas dimensões da janela.
- O evento **beforeunload** é usado para realizar ações ou mostrar um aviso quando o usuário está prestes a sair da página, o que é útil para prevenir que dados sejam perdidos se o usuário sair antes de salvar suas mudanças.

Veja o resultado no navegador, conforme figura abaixo:

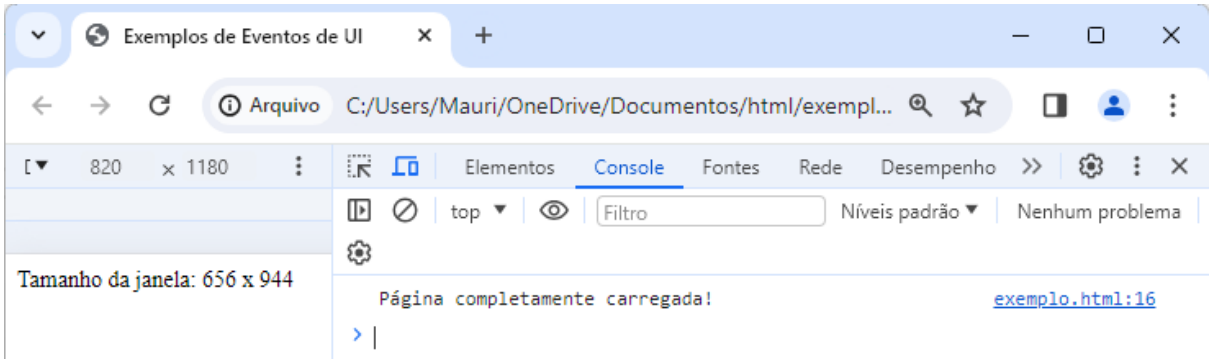

Figura 4 - Exemplo de execução de eventos de UI (User Interface)

Descrição da imagem: Quando a página é completamente carregada, uma mensagem "Página completamente carregada!" é registrada no console do desenvolvedor. Já, quando a janela do navegador é redimensionada, o texto dentro de uma div com o id "resizeInfo" é atualizado para mostrar as novas dimensões da janela (largura x altura). Quando o usuário tenta sair ou fechar a página, uma mensagem de confirmação aparece perguntando "Tem certeza que deseja sair?".

Eventos de mídia

Os eventos de mídia em JavaScript são eventos que ocorrem durante a reprodução de elementos de mídia, como áudio e vídeo, em uma página da web. Esses eventos permitem que você crie interações dinâmicas e responsivas em aplicativos que envolvem mídia. Aqui estão alguns dos principais eventos de mídia em JavaScript:

- **play:** Este evento é acionado quando a reprodução de um elemento de mídia é iniciada, seja pela ação do usuário ou programaticamente.
- **pause:** Este evento é acionado quando a reprodução de um elemento de mídia é pausada, seja pela ação do usuário ou programaticamente.
- **ended:** Este evento é acionado quando a reprodução de um elemento de mídia chega ao fim.

Vejamos no quadro abaixo um exemplo de uso destes eventos:

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo de Eventos de Mídia</title>
</head>
<body>
  <video id="videoPlayer" width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
    Seu navegador não suporta vídeos.
  </video>
  <button id="playButton">Play</button>
  <button id="pauseButton">Pause</button>
  <script>
    document.addEventListener('DOMContentLoaded', function () {
      const video = document.getElementById('videoPlayer');
      const playButton = document.getElementById('playButton');
      const pauseButton = document.getElementById('pauseButton');

      playButton.addEventListener('click', function () {
        video.play();
      });

      pauseButton.addEventListener('click', function () {
        video.pause();
      });

      video.addEventListener('play', function () {
        console.log('O vídeo começou a ser reproduzido.');
      });

      video.addEventListener('pause', function () {
        console.log('O vídeo foi pausado.');
      });

      video.addEventListener('ended', function () {
        console.log('O vídeo foi concluído.');
      });
    });
  </script>
</body>
</html>
```

Veja o resultado no navegador, conforme figura abaixo:

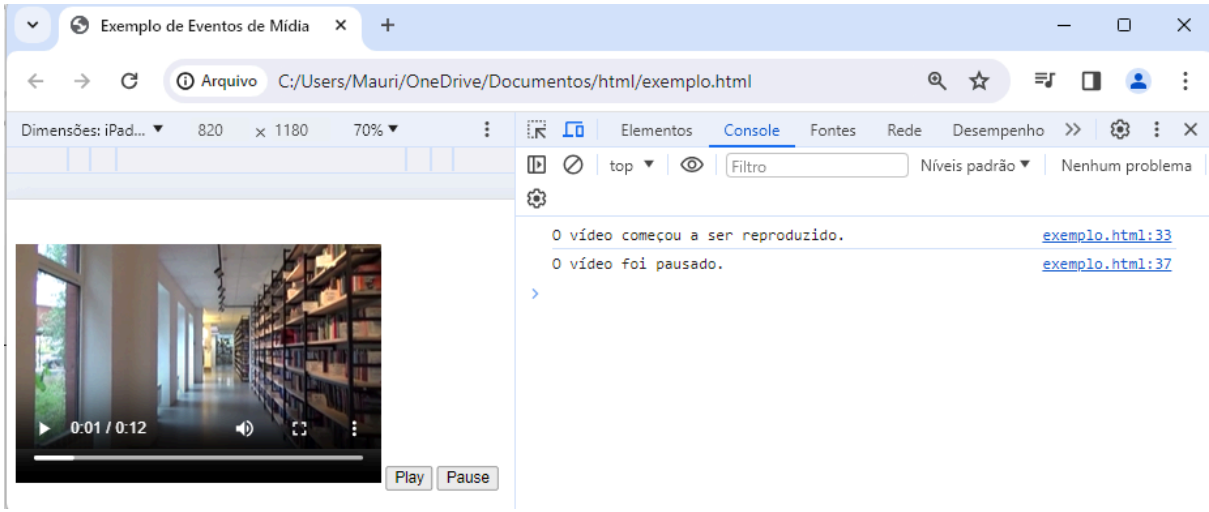

Figura 5 - Exemplo de execução de eventos de mídia

Descrição da imagem: No navegador, observamos que há um vídeo com controles de reprodução e dimensões de 320x240 pixels. O vídeo usa um arquivo chamado "movie.mp4". Se o navegador não suportar vídeos, uma mensagem será exibida dizendo "Seu navegador não suporta vídeos". O vídeo pode ser controlado usando os botões "Play" e "Pause". Mensagens no console do desenvolvedor indicarão quando o vídeo começa a ser reproduzido, é pausado ou é concluído, proporcionando feedback sobre o estado atual do vídeo.

Eventos de documento

Os eventos de documento são eventos que têm uma aplicação global em toda a página da web, em vez de estar associados a um elemento específico. Esses eventos são fundamentais para gerenciar interações e comportamentos que afetam a página como um todo. Os principais eventos deste tipo são:

- **DOMContentLoaded:** Este evento é disparado quando o HTML inicial do documento foi completamente carregado e analisado, sem esperar pelo CSS, imagens e subframes para terminar de carregar. Esse evento é crucial para scripts que necessitam que o DOM esteja pronto antes de executar.

- **load:** O evento load é disparado quando a página inteira (incluindo todos os recursos dependentes como imagens e folhas de estilo) foi completamente carregada. Este evento é útil para inicializar detalhes que dependem desses recursos externos.
- **beforeunload:** Este evento é usado para executar ações quando o usuário está prestes a sair da página. Geralmente é usado para alertar os usuários que eles podem perder dados se saírem da página antes de salvar suas alterações.
- **unload:** Disparado quando o documento está sendo descarregado (por exemplo, quando o usuário navega para outra página, fecha a janela ou recarrega a página).
- **resize:** Disparado quando a janela do navegador é redimensionada.
- **scroll:** Este evento ocorre quando o usuário rola a página.

Vejamos no quadro abaixo um exemplo de uso destes eventos:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemplo DOMContentLoaded</title>
</head>
<body>
  <h1>Exemplo de DOMContentLoaded</h1>
  <div id="message"></div>
  <script>
    document.addEventListener('DOMContentLoaded', function () {
      const message = document.getElementById('message');
      message.textContent = 'O DOM está totalmente carregado e analisado!';
    });
  </script>
</body>
</html>
```

Veja o resultado no navegador, conforme figura abaixo:

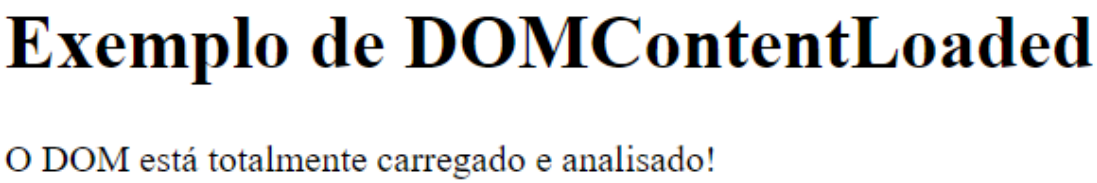

Figura 6 - Exemplo de execução do evento DOMContentLoaded

Descrição da imagem: No navegador, a página contém um título "Exemplo de DOMContentLoaded" e uma div vazia. Ao carregar a página, o texto "O DOM está totalmente carregado e analisado!" aparecerá na página, indicando que a estrutura do documento HTML foi completamente carregada e analisada.

◄ 5.2 Formas de tratamento de eventos

Seguir para...

5.4 Teste seus conhecimentos ►