EVENTOS

Unidade 02

Sumário

Olá, **Fullturist,**

Seja bem-vindo(a)!

Olá, Fullturist! Bem-vindo(a) à trilha de Javascript!

Ao navegar pela internet, seja pelo seu computador ou utilizando seu smartphone, você já deve ter acessado sites como Facebook e Instagram e ficado admirado(a) com algumas funcionalidades fantásticas. Além disso, provavelmente, deparou-se com jogos divertidos e interativos que prendeu a sua atenção por um bom tempo. Nesse sentido, perguntou-se: como tudo isso é feito?

Para desenvolver algo parecido, é preciso dominar linguagens que vão além do HTML e CSS. Assim, nesta Trilha, vamos apresentar a linguagem de programação Javascript, que possibilitará o desenvolvimento de sites e aplicativos semelhantes ao Facebook, Instagram e Tik Tok.

A partir de agora, será possível entender e desenvolver funcionalidades mais complexas. Quando uma página web, como de redes sociais, faz mais do que apenas mostrar informações estáticas, ao invés disso, mostram em tempo real conteúdos atualizados, mapas interativos, animações gráficas em 2D/3D, entre outros, tenha certeza de que o Javascript está por trás de tudo. Ficou empolgado(a)? Então, vamos lá!

Unidade 02

Eventos

Olá, Fullturist!

Até o momento, você aprendeu vários conceitos da linguagem Javascript. Esses conhecimentos básicos irão lhe permitir ir ainda mais longe. Então, chegou a hora de aprender a aplicar os conhecimentos aprendidos para permitir que as pessoas interajam com os layouts já desenvolvidos. Nesta Unidade, vamos estudar como manipular os eventos em uma página. Ficou animado(a)? Está curioso(a) para saber como isso pode ser feito? Antecipamos dizendo que pode ser vinculando códigos Javascript a eventos de mouse, de teclado, de janela ou de formulários.

Portanto, nesta Unidade, você irá conhecer, testar e aprender tudo sobre eventos em Javascript.

Vamos nessa?

Objetivos de aprendizagem

Ao final do estudo desta unidade, você será capaz de:

- Compreender o que são eventos em Javascript.
- Conhecer os tipos de eventos mais utilizados em Javascript e saber como manipulá-los.
- Desenvolver layouts/páginas que possibilitam a interatividade.

2.1 Eventos

Quando você desenvolve uma página utilizando a linguagem de marcação HTML e a linguagem CSS, cria um layout bonito, estiloso e bastante atrativo. Porém, o site continua estático, imóvel, como se fosse uma foto que apenas pode ser visualizada e sem interação. O Javascript permite que profissionais desenvolvam a parte que estava faltando: a interatividade! No entanto, como isso pode ser feito?

Veremos que a interatividade pode ser programada através dos eventos! Desse modo, a seguir, vamos explicá-los, de modo que você entenda melhor o que isso significa.

2.1.1 O que são Eventos?

Eventos são muito fáceis de entender e de serem manipulados. Diante disso, analise o seguinte exemplo: você acessa o Instagram em seu computador, por meio do seu navegador favorito. Ao utilizá-lo, em toda postagem que você curte ou salva, um evento de clique do mouse acontece. Quando uma pessoa comenta uma publicação de um amigo e pressiona a tecla enter para confirmar, um evento relacionado ao pressionar de uma tecla acontece. Quando você posiciona o mouse sobre a foto do um contato, as informações sobre este contato são exibidas, nesse momento, outro tipo de evento acontece. Acho que agora ficou mais fácil de entender, não é?

De uma forma mais técnica, eventos são basicamente ações que estão vinculadas a algum conteúdo/tag HTML, como um botão, que são acionadas dentro do navegador. Esses eventos podem estar associados a um ou mais elementos HTML, ou seja, podem estar vinculados a botões, textos, imagens, entre outros.

Agora que vimos o que são eventos, que tal conhecermos alguns deles?

O Javascript possui uma lista grande de eventos. A seguir, vamos conhecer os mais comuns, mais utilizados.

Eventos	Ações Executadas pelos Eventos
	Eventos de Mouse
onclick	É acionado no clique do mouse, ou seja, quando clicamos em um elemento HTML, por exemplo, um botão. Ele é completo quando clicamos e soltamos o botão esquerdo do mouse.
ondblclick	É disparado no clique duplo do mouse. É o famoso duplo clique!
onmousedown	É executado no clique do mouse, a diferença deste evento para o onclick é que é acionado soltando ou não o botão esquerdo do mouse.
onmouseover	É acionado quando o cursor do mouse sobrepõe o elemento HTML. Um exemplo seria quando você posiciona o mouse sobre uma foto no Facebook e as informações aparecem.
onmouseout	É executado quando o cursor do mouse sai da área de um elemento, por exemplo, quando o cursor sai da área de uma imagem.
	Eventos de Teclado
onkeydown	Este evento é disparado quando qualquer tecla no teclado é pressionada, soltando ou não a tecla.
onkeypress	É invocado quando uma tecla no teclado é pressionada, soltando-a ou não. No entanto, não funciona para as teclas alt, ctrl, esc, shift etc., sendo utilizado apenas para teclas úteis, ou seja, letras, números, vírgula, acentos etc.
onkeyup	É executado quando a tecla é liberada, ou seja, no movimento completo, quando você clica e solta a tecla. Se você pressionar a tecla e ficar segurando, nada acontece!

Eventos de Janela	
onload	É invocado quando toda a página HTML for carregada no navegador.
onresize	Este evento é disparado quando redimensionamos a janela do navegador.
Eventos de Formulários	
onfocus	É executado quando um elemento de formulário, por exemplo, um <input>, recebe o foco do cursor.
onblur	É disparado quando o elemento do formulário perde o foco do cursor. Em outras palavras, quando você sai de um campo <input>.
onchange	Ocorre quando o conteúdo de um elemento de formulário é alterado. É muito utilizado pelo elemento <select>. Toda vez que você selecionar uma informação diferente na lista, o evento ocorrerá.
onSubmit	É disparado em um formulário quando é necessário enviar os dados informados nos campos para o backend.

Fonte: Elaborada pelo autor (2021).

Até agora vimos que existem eventos de mouse, de teclado, de formulários e, até, para a janela do navegador. Nesse momento, você deve estar curioso(a) para saber como programar e ansioso(a) para botar a mão na massa. Então, vamos lá!

2.2 Como manipular os eventos em JavaScript

Como já foi dito, um evento ocorre quando o usuário interage com a página que você desenvolveu, passando o mouse sobre uma imagem, clicando em um botão, preenchendo dados em um formulário. Isso quer dizer que os elementos, as tags HTML, estão relacionadas com esses eventos. Então, para programá-los, é necessário fazer um link com o elemento HTML, que irá acionar o evento. Nesse sentido, existem duas maneiras de programar e vincular esses eventos aos elementos HTML, a saber:

- Inline: eventos programados diretamente na própria tag HTML, através de um atributo.

- Em um arquivo externo: onde você deverá programar utilizando um manipulador de eventos do Javascript.

Vamos começar a programar os eventos da forma inline.

2.2.1 Manipulação de Eventos Inline

A forma de programar os eventos inline é executando códigos Javascript nas próprias tags HTML, as quais possuem um atributo que executa um evento. Vamos começar com um exemplo, onde programaremos os eventos de mouse.

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
    <title>Testando os Eventos de Mouse</title>
  </head>
  <body>
    <button onclick=”alert(‘Evento de click executado’);”>Clique aqui</
button>
    <p ondblclick=”alert(‘Duplo click do mouse executado’);”>Clique aqui
para testar o duplo clique!</p>
    <button onmousedown=”alert(‘Evento de click executado’);”>Clique aqui
para testar o evento onmousedown</button>
    <br>
    <br>
    <img onmouseover=”alert(‘Executando o evento onmouseover’);”
src=”https://www.fullture.com/wp-content/uploads/2020/07/Simbolo-
Square-1.svg” />
    <br>
```

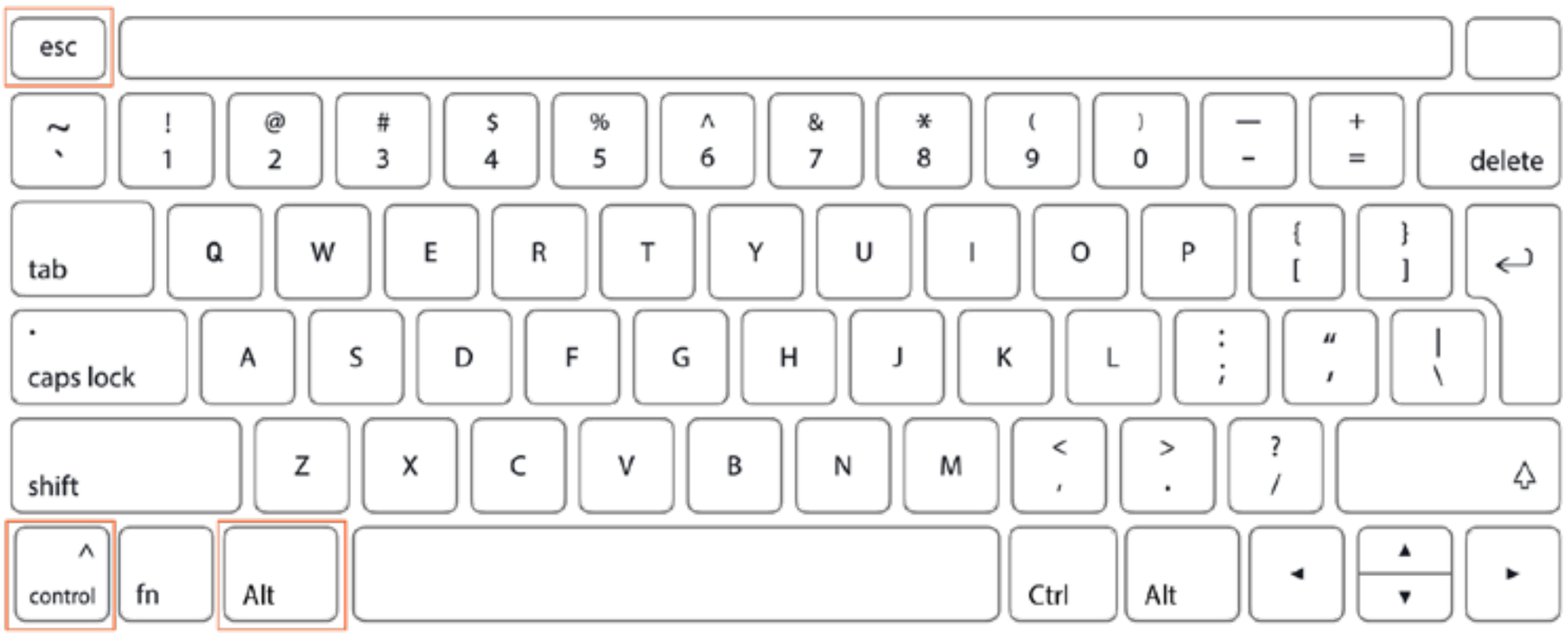
```
        <br>
        <img onmouseout=”alert(‘Executando o evento onmouseout’);”
src=”https://www.fullture.com/wp-content/uploads/2020/07/Simbolo-
Square-1.svg” />
    </body>
  </html>
```

Fonte: Elaborada pelo autor (2021).

A partir disso, vamos testar os eventos de teclado. Para isso, no próximo exemplo, utilizaremos a tag <input> para programar os eventos. No código a seguir, estão sendo utilizadas três tags <input>, e em cada uma delas está programado um evento diferente.

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
    <title>Javascript em Arquivo Externo</title>
  </head>
  <body>
    <label for=”campo1”>Campo 1</label>
    <input onkeydown=”alert(‘evento onkeydown’);” type=”text” id=”campo1”
/>
    <br><br>
    <label for=”campo2”>Campo 2</label>
    <input onkeypress=”alert(‘evento onkeypress’);” type=”text” id=”campo2”
/>
    <br><br>
    <label for=”campo3”>Campo 3</label>
    <input onkeyup=”alert(‘evento onkeyup’);” type=”text” id=”campo3” />
  </body>
</html>
```

O input da linha 8 vai executar o evento onkeydown. Quando você pressionar qualquer tecla no teclado, um alerta será exibido. Para testá-lo, pressione qualquer tecla, inclusive as teclas alt, ctrl, esc, conforme figura que segue.



Fonte: Printerest, ID 624241198332929294.

Na linha 11, o input está programado com o evento onkeypress. Teste pressionando qualquer tecla, mas perceba que este evento não é acionado ao pressionar as teclas alt, ctrl, esc.

Por fim, o input da linha 14 está sendo programado com o evento onkeyup, que só será acionado quando pressionar e soltar a tecla. Para testar, pressione a tecla A, segure por um segundo e solte. Perceba que o evento só vai ocorrer quando você soltar.

Para conhecer mais sobre os eventos de teclado estudados até agora e conhecer outros, visite o site da MDN.

Os eventos de janela são vinculados ao navegador. No código de exemplo a seguir estão sendo programados os eventos onload e onresize, os quais são associados à tag <body> na linha 7.

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
    <meta charset=”utf-8” />
    <title>Javascript em Arquivo Externo</title>
  </head>
  <body onload=”alert(‘Pagina totalmente carregada!’);”
onresize=”alert(‘Janela redimencionada’);”>
    <img rc=”https://www.fullture.com/wp-content/uploads/2020/07/
Simbolo-Square-1.svg” />
    <h1>Dev Full Stack</h1>
    <h2>Torne-se o profissional que o mundo precisa!</h2>
  </body>
</html>
```

Fonte: Elaborada pelo autor (2021).

Quando testar, perceba que a mensagem “Página totalmente carregada!” será exibida e, logo depois, o conteúdo será apresentado. Isso acontece porque todo o código HTML foi carregado na memória e formou o DOM (Document Object Model), o qual estudaremos na próxima unidade.

Para testar o outro evento, o resize, aumente e diminua a largura ou a altura da tela do navegador. Toda vez que a tela for redimensionada, a mensagem “Janela redimensionada” será emitida.

O evento de Janela load é o mais importante. Com ele, os demais eventos e códigos Javascript serão vinculados aos elementos HTML somente depois que estiverem carregados na memória, evitando, assim, uma série de problemas que poderiam ocorrer.

Por fim, vamos a um exemplo para testarmos e entendermos melhor como funcionam os eventos de formulários. Como a própria definição sugere, você deverá programar esses eventos em um formulário, vinculando os elementos, como inputs, textareas e selects, a ações que necessitem que sejam executadas.

No código que segue está sendo programado os eventos onfocus, onblur, onchange e onSubmit. Confira!

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
    <meta charset=”utf-8” />
    <title>Javascript em Arquivo Externo</title>
  </head>
  <body>
    <h1>Formulário de Contato</h1>
    <h2>Entre em contato conosco enviando uma mensagem!</h2>
    <form action=”#” method=”GET” onsubmit=”alert(‘Enviando dados!’);”>
      <p>
        <lable for=”nome”>Nome </lable>
        <input onfocus=”this.value=’recebeu o foco’;” type=”text” id=”nome” />
      </p>
      <p>
        <lable for=”nome”>e-mail </lable>
        <input onblur=”alert(‘Sai do campo e-mail!’);” type=”text” id=”email” />
      </p>
      <p>
        <lable for=”tipo”>Tipo do contato </lable>
        <select id=”tipo” onchange=”alert(‘alterou o valor’);”>
          <option>Sugestão</option>
          <option>Reclamação</option>
          <option>Outros</option>
        </select>
      </p>
      <p>
        <input type=”submit” value=”Enviar Contato” />
      </p>
    </form>
  </body>
</html>
```

Analisando o código anterior, começamos pelo evento onsubmit que está sendo programado na tag <form>, na linha 10. O alerta com a mensagem “Enviando dados!” será exibido quando pressionar o botão de “Enviar contato” da linha 28. Na linha 13, está sendo implementado o evento onfocus, o qual será acionado quando o campo nome receber o foco.

Para testar, no navegador, pressione o mouse sobre o campo nome. O texto “recebeu o foco” irá ser preenchido no campo. O próximo evento implementado na linha 17 é o onblur, que será ativado quando o foco estiver no campo e-mail e você mudar para qualquer outro campo, ou seja, quando o foco sair do campo. Para finalizar, na linha 21, está sendo implementado o evento onchange, que será usado toda vez que selecionar um valor diferente na lista de opções.

Os navegadores como o Chrome executam a leitura do arquivo HTML e analisam todos os conteúdos nele contido. É dessa forma que ele busca as palavras-chave que você digita no campo de pesquisa e encontra os conteúdos. Além de facilitar a manutenção e evolução das páginas, a separação do HTML, CSS e Javascript, é também uma técnica de SEO (Search Engine Optimization). Para saber mais sobre SEO, clique aqui.

Vamos recapitular o que você aprendeu até agora? Eventos são ações que são executadas quando os usuários interagem com a página que você está desenvolvendo. Com Javascript, é possível manipular uma série de eventos.

Estudamos até aqui os eventos mais utilizados que são os eventos de mouse, teclado, formulário e janela. Nos exemplos, os eventos foram manipulados dentro das próprias tags HTML, através das propriedades referentes a cada evento, por exemplo, a propriedade onclick. Porém, essa forma de manipulá-los não é uma boa prática, pois viola a separação do HTML, CSS e Javascript, o que dificulta a manutenção e evolução dos sistemas e a leitura dos conteúdos realizada pelos sites de buscas. Por esse motivo, a manipulação de eventos inline não é recomendada.

A manipulação de eventos em arquivos externos é a forma recomendada, além de ser uma boa prática de programação. Vamos ver como isso é feito?

2.2.2 Manipulação de Eventos em Arquivo Externo

A manipulação de eventos utilizando um arquivo de script externo é programada separando os códigos Javascript em um arquivo separado do arquivo HTML. Então, será necessário criar um arquivo, como, por exemplo, index.js, e programar um link para esse arquivo no HTML. Nele, você pode manipular os eventos de duas formas: capturando os elementos HTML (tags) e acessando as propriedades de eventos ou adicionando escutadores de eventos.

2.2.2.1 Manipulando Eventos Através das Proprieda-des dos Elementos

Para que fique mais simples o entendimento, vamos alterar em duas partes o exemplo do Código 1, onde estão sendo manipulados os eventos do mouse: primeiro, vamos mudar o código HTML e, depois, criaremos um arquivo com o nome de index. js, em que programaremos os códigos Javascript. Implementamos o exemplo a seguir e salvamos o arquivo como index.html.

```
<!DOCTYPE html>
<html lang=”pt-br”>
  <head>
    <title>Testando os Eventos de Mouse</title>
  </head>
  <body>
    <button id=”botao1”>Clique aqui</button>
    <p>Clique aqui para testar o duplo clique!</p>
    <button id=”botao2”>Clique aqui para testar o evento onmousedown</
button>
    <br>
    <br>
```

```
        <img id="img1" src="https://www.fullture.com/wp-content/
uploads/2020/07/Simbolo-Square-1.svg" />
        <br>
        <br>
        <img id="img2" src="https://www.fullture.com/wp-content/
uploads/2020/07/Simbolo-Square-1.svg" />
         <script src="index.js"></script>
    </body>
  </html>
```

Fonte: Elaborada pelo autor (2021).

No código anterior, foram realizadas algumas alterações importantes. A primeira, removeu das tags todos os códigos Javascript e as propriedades referentes aos eventos. Outra alteração, é que foram atribuídos identificadores (id) para os botões nas linhas 7 e 9, e para as imagens nas linhas 12 e 15, os quais serão utilizados para capturar os elementos HTML no arquivo externo index.js. Por último, a alteração mais importante, na linha 17 está sendo feito um link do arquivo externo index.js, que permitirá que o navegador encontre os códigos Javascript e execute os eventos. Esse link deverá ser colocado no final do arquivo HTML, pois, dessa forma, será garantido que todos os elementos da página estejam carregados. É possível colocá-lo no início da página entre as tags <head></head>. Entretanto, para que não ocorram bugs, será necessário incluir todo o código dentro do evento window.onload.

Com o código HTML alterado, vamos agora manipular os eventos. Primeiramente, você aprenderá a manipular os eventos, programando através da captura dos elementos e utilizando as propriedades. Nesse sentido, analise, implemente e teste o código a seguir. Para isso, escreva o código em um novo arquivo e salve como index.js.

```
const btn1 = document.querySelector(“#button1”);
const p = document.querySelector(“p”);
const btn2 = document.querySelector(“#button2”);
const img1 = document.querySelector(“#img1”);
const img2 = document.querySelector(“#img2”);
btn1.onclick = function() {
   alert(“Evento de click executado”);
}
p.ondblclick = function(){
   alert(“Duplo click do mouse executado”);
}
btn2.onmousedown = function(){
   alert(“Evento mousedown do mouse executado”);
}
img1.onmouseover = function(){
   alert(“Executando o evento onmouseover”);
}
img2.onmouseout = function(){
   alert(“Executando o evento onmouseout”);
}
```

Fonte: Elaborada pelo autor (2021).

Ao testar os códigos, você conseguiu perceber que a funcionalidade continua a mesma dos primeiros exemplos? Se não, vamos entender melhor o código: no início do arquivo código, entre as linhas 1 e 5, estão sendo capturados os elementos HTML. Para realizar a captura desses elementos, utiliza-se o DOM, o qual você aprenderá mais na unidade 3. É possível acessá-lo usando do objeto document. Para selecionar os elementos, o método querySelector() foi empregado. Através dele, os elementos podem ser selecionados e capturados da mesma forma utilizada no CSS, a partir do nome da própria tag, como no parágrafo na linha 2, pelo id, como está sendo feito para as demais tags, e com base no nome de uma classe.

Como dito anteriormente, vamos abordar o DOM em detalhes na Unidade 3, mas se você estiver ansioso(a) e quer saber mais sobre o DOM clique aqui.

Uma vez capturados os elementos, temos acesso a tudo, inclusive suas propriedades e eventos. Sendo assim, o comportamento dos eventos foi programado utilizando a propriedade referente a cada evento, como pode ser visto nas linhas 7, 11, 15, 19 e 23.

Na linha 7, então, foi adicionada a ação para o evento de clique de mouse, o onclick. Para inclui-la, foi criada uma função anônima, que é atribuída a propriedade onclick do elemento. Quando o botão for acionado (clicado) pelo usuário, essa função será executada. Para todos os demais eventos, a forma e a lógica de manipulação é a mesma da explicada no evento onclick.

Manipular eventos através das propriedades é uma boa prática de programação, pois permite separar o código Javascript dos elementos HTML. Se você está começando agora, essa forma, provavelmente, é mais aconselhável, pois é mais simples de entender. Todavia, tem menos poder e opções de manipulação, não é a forma mais atual, além de não ser a utilizada em aplicações maiores.

A manipulação de eventos adicionando listeners (escutadores de eventos) é a forma mais moderna de manipular eventos em Javascript, além de ser mais poderoso e com mais opções. Vamos conhecer essa forma?

2.2.2.2 Manipulando eventos adicionando Listeners

Manipular eventos através de listeners foi definido nas especificações atuais do DOM e traz uma série de benefícios para o dev Javascript, dentre eles estão: a remoção do código do manipulador de eventos (o elemento HTML) e a possibilidade de os eventos executarem duas funções (ações) diferentes. Para começar, vamos reescrever o código 6 contido no arquivo index.js, adicionando listeners aos elementos.

```
const btn1 = document.querySelector(“#button1”);
const p = document.querySelector(“p”);
const btn2 = document.querySelector(“#button2”);
const img1 = document.querySelector(“#img1”);
const img2 = document.querySelector(“#img2”);

btn1.addEventListener(‘click’, function(){
        alert(“Evento de click executado”);
});

p.addEventListener(‘dblclick’, function () {
        alert(“Duplo click do mouse executado”);
});

btn2.addEventListener(‘mousedown’, function () {
        alert(“Evento mousedown do mouse executado”);
});

img1.addEventListener(‘mouseover’, function () {
        alert(“Executando o evento onmouseover”);
});

img2.addEventListener(‘mouseout’, function () {
        alert(“Executando o evento onmouseout”);
});
```

Fonte: Elaborada pelo autor (2021).

Como é possível perceber, para manipular os eventos e adicionarmos listeners, é necessário capturar os elementos, assim como foi feito para manipular eventos através das propriedades. Então, entre as linhas 1 e 5 nada mudou. O que alterou foi a forma de programação das ações ao evento, em que se passa a utilizar o método addEventListener().

Agora, vamos analisar como é adicionado o evento de clique do mouse ao elemento button1. Para os demais elementos, a lógica e a forma de adição de listener é a mesma. Vamos lá!

Na linha 6, está sendo adicionado um um listener ao button1, salvo na constante btn1. Para acrescentá-lo no evento de clique, é necessário utilizar o método addEventListener(), que necessita de dois parâmetros obrigatórios: o primeiro é o evento, nesse caso, o 'click' do mouse, e o segundo é uma função que contém o código que será executado quando se clica no botão. Até agora, o que mudou foi apenas a sintaxe, concorda? Então, vamos às mudanças mais importantes.

Manipular eventos com listeners permite que se remova um código que será executado e que se adicione mais de uma função no mesmo evento. Para entender melhor, edite o arquivo index.html, adicionando o código em destaque no exemplo que segue.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Testando os Eventos de Mouse</title>
  </head>
  <body>
    <button id="botao1">Clique aqui</button>
    <p>Clique aqui para testar o duplo clique!</p>
    <button id="botao2">Clique aqui para testar o evento onmousedown</
button>
    <br>
    <br>
    <img id="img1" src="https://www.fullture.com/wp-content/
uploads/2020/07/Simbolo-Square-1.svg" />
    <br>
    <br>
    <img id="img2" src="https://www.fullture.com/wp-content/
```

```
uploads/2020/07/Simbolo-Square-1.svg" />
        <button id="button3">Trocar evento do button1</button>
        <script src="index.js"></script>
    </body>
</html>
```

Fonte: Elaborada pelo autor (2021).

A ideia é, quando clicar sobre o button3, um código Javascript irá remover a ação programada ao evento de click do button1 e adicionará um novo código para o evento de click do botao1. Então, altere o arquivo index.js, adicionando as linhas em destaque no código a seguir.

```
const btn1 = document.querySelector("#button1");
const p = document.querySelector("p");
const btn2 = document.querySelector("#button2");
const img1 = document.querySelector("#img1");
const img2 = document.querySelector("#img2");
const btn3 = document.querySelector("#button3");
const clique = function () {
   alert("Evento de click executado");
}
btn1.addEventListener('click', clique );
p.addEventListener('dblclick', function () {
   alert("Duplo click do mouse executado");
});
btn2.addEventListener('mousedown', function () {
   alert("Evento mousedown do mouse executado");
});
img1.addEventListener('mouseover', function () {
   alert("Executando o evento onmouseover");
});
```

```
img2.addEventListener('mouseout', function () {
    alert("Executando o evento onmouseout");
 });
 btn3.addEventListener('click', function(){
    btn1.removeEventListener('click', clique);
    btn1.addEventListener('click', function(){
       alert("Executando novo codigo do click do botao1");
    });
 });
```

Fonte: Elaborada pelo autor (2021).

Para perceber a alteração realizada, abra o arquivo html no navegador, clique no primeiro botão e o alerta vai ser exibido com a mensagem "Evento de click executado". Agora, clique no novo botão e depois, novamente, no primeiro botão. Uma nova mensagem de alerta "Executando novo código do click do botao1". Vamos entender como essa nova alteração no código funciona.

Primeiramente, analisaremos a alteração no código entre as linhas 8 e 12: foi modificada a forma de atribuir a função ao listener do evento de click do btn1, onde, primeiro, criamos e conferimos uma função a constante clique (linhas 8 a 10) por adicionar o listener ao botão. Por que isso? Isso foi feito para que uma função seja removida do listener e o Javascript precisa saber quem é ela, qual o seu link.

Então, no listener do botão 3 (linhas 30 a 37), é possível remover a função do botão 1 (linha 32), pois o Javascript usa o atalho para a função com a constante clique. Para finalizar, na linha 34, um novo código é adicionado ao evento de click do botão 1.

Nesse momento, vocês já deve entender melhor como funcionam os eventos e como programá-los inline e em arquivo externo utilizando as propriedades ou listeners.

Na próxima unidade, vamos continuar estudando sobre eventos e você irá praticar ainda mais, manipulando-os através de listeners. Ficou curioso(a)? Então, vamos para a próxima Unidade!

Síntese

Nesta Unidade, você conheceu os principais eventos programados em Javascript e suas finalidades. Confira os itens mais importantes:

- Aprendeu que é possível programar diferentes eventos para o mouse e qual o comportamento de cada um.

- Compreendeu que existem diferentes eventos para o teclado, alguns bem semelhantes, porém, com finalidades diferentes.

- Vimos o que são os eventos de Janela e a importância do evento load para evitar erros e bugs.

- Ao estudar sobre os eventos de formulário, verificou que os eventos são diretamente relacionados a campos de um formulário, inclusive, no próprio formulário, quando seus dados estão sendo enviados para o backend.

- Conheceu três diferentes formas de manipular eventos: inline, através das propriedades e adicionando listeners.

- Aprendeu que manipular eventos inline não é recomendado e que, definitivamente, você não deve utilizar.

- Constatou que utilizar listener é a forma mais recomendada para manipular eventos, pois permite remover ações de eventos já programados e adicionar mais de uma função para um evento.

Fullture Insights

- MALDONADO, Leonardo. Entendendo o DOM (Document Object Model). Tableless, [S.l.], 8 fev. 2018. Disponível em: https://tableless.com.br/entendendo-o-dom-document-object-model/. Acesso em: 6 set. 2021.

- MDN Web Docs. Documentação sobre Javascript. [S.l]: MDN Web Docs, 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/Events. Acesso em: 25 ago. 2021.

- W3SCHOOLS. Javascript Events. [S.l]: W3 Scholls, 2021. Disponível em: https://www.w3schools.com/js/js_events.asp. Acesso em: 18 ago. 2021.

FU
LL
TU
RE

www.fullture.com