Aula 8

Criando nossa história final

▶ Unidade

Funções: criando uma missão sobre Inteligência Artificial

Questão 1 – Sobre o método addEventListener()

Leia o código abaixo:

```
var botaoDescobrir = document.getElementById('descobrir');

function mostrarFato() {

    alert("O elefante é o maior animal terrestre!");

}

botaoDescobrir.addEventListener("clicar", mostrarFato);
```

Silvio está desenvolvendo um jogo educativo online para crianças aprenderem sobre animais. O jogo possui um botão *Descubra mais!* que, quando clicado, deve mostrar uma imagem e um fato interessante sobre o animal apresentado. No entanto, atualmente ao clicar no botão, nada acontece.
Qual é o erro no código impede que a função **`mostrarFato()`** seja executada ao clicar no botão?

a) O método **`addEventListener()`** recebe apenas o parâmetro do evento que deve ser ouvido.

b) O método **`addEventListener()`** recebe dois parâmetros: o evento que será ouvido e uma função. Essa função pode ser tanto anônima quanto uma função seta.

c) O método **`addEventListener()`** recebe dois parâmetros: o evento que será ouvido e uma função. Essa função deve ser do tipo anônima.

d) O método **`addEventListener()`** recebe dois parâmetros: o evento que será ouvido e uma função. Essa função deve ser do tipo seta.

Alternativa A, incorreta. O método *addEventListener()* recebe, além do evento, uma função.

Alternativa B, correta. O método recebe tanto o evento quanto a função. Essa função pode ser declarada de duas formas, dependendo do uso e tamanho das instruções. De um modo geral, queremos evitar que uma função anônima contenha muitas instruções complexas.

Alternativa C, incorreta. Este método pode receber, além da anônima, mais tipos de função.

Alternativa D, incorreta. Este método pode receber, além da seta, mais tipos de função.

Questão 2 – Corrigindo o código

Observe o código abaixo:

```
botao._____ = opcao.texto;

botao._____('click', () => cafeSelecionado(opcao));

containerDeOpcoes._____(botao);
```

Com esse código, algumas opções de café são criadas. Quando o usuário clica em uma dessas opções, o café se torna filho do elemento **containerDeOpcoes** e, portanto, passa a ser exibido na tela como um elemento HTML.

No entanto, estão faltando alguns métodos para que esse código funcione. Para fazer a correção, ordene **todos os blocos** abaixo que preencham corretamente as lacunas.

`appendChild` `textContent` `addEventListener`

Escreva a sequência correta de blocos nas linhas a seguir:

Sequência correta: textContent | addEventListener | appendChild

Comentário: para resolver essa situação, precisamos primeiramente definir o que o botão vai exibir (*textContent*). Depois, definimos o que acontece quando o botão é clicado (*addEventListener*), e, por último, colocamos o botão na página, dentro do elemento container (*appendChild*).

Questão 3 – Fazendo compras

Crie um algoritmo que simula uma compra de supermercado. A cada código de barras que for lido, a quantidade de itens deve ser incrementada ao total. Ao fim das compras, é necessário que o total gasto seja exibido. Considere que esse total seja calculado em uma nova função.

Para isso, ordene **todos os blocos** abaixo:

`mostraTotal();}` `itens++;` `function compras(){`

Escreva a sequência correta de blocos nas linhas a seguir:

Sequência correta: function compras(){ | itens++; | mostraTotal();}

Comentário: para representar o incremento de um elemento, podemos utilizar a nomenclatura *elemento ++* ou *elemento = elemento + 1*. Além disso, podemos chamar uma função dentro de outra sempre que necessário. Dica: há uma diferença textual quando criamos uma função e quando apenas a chamamos em um código. Em um dos casos, usamos a palavra reservada function; no outro, usamos apenas o nome dela e parênteses.