UNINOVE

Funções

PRÁTICA E MANIPULAÇÃO COM FUNÇÕES EM JAVASCRIPT

AUTOR(A): PROF. DANIEL FERREIRA DE BARROS JUNIOR

Funções

Em continuidade ao assunto iniciado no tópico anterior, vimos que o uso das funções em JavaScript podem ser flexíveis e úteis em diversos casos e a habilidade de programar boas estruturas é essencial.
Vamos analisar algumas possibilidades de se trabalhar com funções. Vale lembrar, que o uso de funções não é recurso único das operações matemáticas. Elas são muito importantes em conjunto com a manipulação de objetos e elementos HTML.
Vejamos o exemplo a seguir:

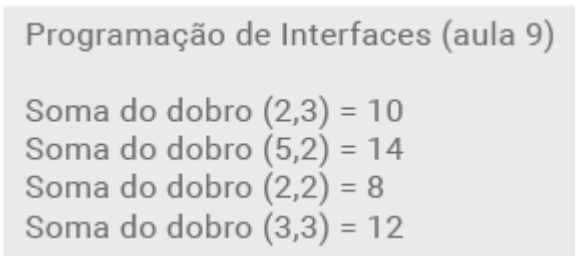
```
<!DOCTYPE html>
<html>
    <head>
        <title>Tópico 7</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <div>Programação de Interfaces (aula 9)</div>
        <script type="text/javascript">
            function somarDobro(a,b){
                function dobro(n){
                    return n * 2;
                }

                return dobro(a) + dobro(b);
            };

            document.write("<br>Soma do dobro (2,3) = " + somarDobro(2,3)
            document.write("<br>Soma do dobro (5,2) = " + somarDobro(5,2)
            document.write("<br>Soma do dobro (2,2) = " + somarDobro(2,2)
            document.write("<br>Soma do dobro (3,3) = " + somarDobro(3,3)
            //document.write("<br>Dobro de 3 = " + dobro(3) );

        </script>
    </body>
</html>
```

A imagem abaixo exemplifica o resultado do código anterior.

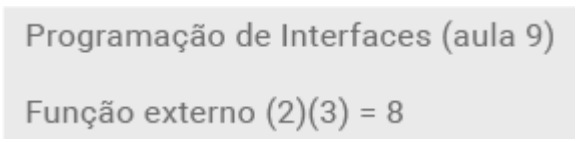


Neste exemplo, temos uma função dentro de outra função. A função mais interna é chamada apenas dentro da função mais externa.

Observe que a linha 22 está comentada, para evitar erro de compilação. Este comando tentaria acessar a função dobro(3) sem utilizar-se da função mais externa, a somarDobro().
Ocorre que isso não é permitido, pois de certa forma a função está protegida e inacessível. No entanto, há maneiras corretas de acessar e passar parâmetros, ou argumentos, para funções internas.
Veja este outro exemplo:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <title>Tópico 7</title>
5.         <meta charset="UTF-8">
6.     </head>
7.     <body>
8.         <div>Programação de Interfaces (aula 9)</div>
9.         <script type="text/javascript">
10.            function externo(a){
11.                function interno(b){
12.                    return a * (b+1);
13.                }
14.                return interno;
15.            };
16.            document.write("<br>Função externo (2)(3) = " + externo(2)(3)
17.        </script>
18.    </body>
19. </html>
```

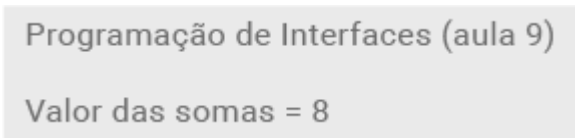
A imagem abaixo exemplifica o resultado do código apresentado.



Neste exemplo, foram passados dois argumentos, o primeiro para função mais externa e o segundo para a função mais externa.
Vamos analisar outro exemplo, agora com funções alinhadas:

```
1. <!DOCTYPE html>
2. <html>
3.     <head>
4.         <title>Tópico 7</title>
5.         <meta charset="UTF-8">
6.     </head>
7.     <body>
8.         <div>Programação de Interfaces (aula 9)</div>
9.         <script type="text/javascript">
10.             function valor1(a){
11.                 function valor2(b){
12.                     function valor3(c){
13.                         document.write("Valor das somas = " + (a + b + c)
14.                     }
15.                     valor3(5);
16.                 }
17.                 valor2(2);
18.             }
19.             var x = valor1(1);
20.
21.         </script>
22.     </body>
23. </html>
```

A imagem abaixo exemplifica o resultado do código acima.



Para o exemplo acima, temos o seguinte resultado:

Valor das somas = 8

Podemos ir além no uso de funções, com o uso de Closures. Veja este exemplo:

A imagem a seguir exemplifica o resultado do código acima.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Tópico 7</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <div>Programação de Interfaces (aula 9)</div>
        <script type="text/javascript">
           var Pessoa = function (nome){
               var idade;
               return{
                   setNome: function(newNome){
                       nome = newNome;
                   },
                   getNome: function(){
                       return nome;
                   },
                   setIdade: function(newIdade){
                       idade = newIdade;
                   },
                   getIdade: function(){
                       return idade;
                   }
               }
           }

           var aluno = Pessoa('Eduardo');
           document.write("<br>Nome do aluno: " + aluno.getNome());
           aluno.setNome('João');   // alterado o nome para João
           aluno.setIdade(20);
           document.write("<br>Idade: " + aluno.getIdade());        // ret
           document.write("<br>Nome Atualizado: " + aluno.getNome());

        </script>
    </body>
</html>
```

A imagem a seguir exemplifica o resultado do código anterior.

```
Programação de Interfaces (aula 9)

Nome do aluno: Eduardo
Idade: 20
Nome Atualizado: João
```

> ## DICA:
>
> Para saber mais sobre funções, veja em:
> https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Funções
> (https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Funções)
> https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions
> (https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions)

Neste nível de as funções são projetadas para dar uma maior segurança aos dados, onde apenas algumas funções podem alterar determinados valores.

Conclusão

Neste tópico analisamos algumas formas de construção de funções. Esta flexibilidade permite a geração de códigos mais complexos. No entanto, utilize este recurso quando for mais adequado e necessário, afim de evitar erros e um código inelegível.

ATIVIDADE

Escolha a alternativa correta para o script a seguir:

```
function somarDobro(a,b){
  function dobro(n){
    return n * 2;
  }
  return dobro(a) + dobro(b);
};
```

A. somarDobro[2,2]?;

B. somarDobro(2,2)?;

C. somarDobro(2;2)?;

D. somarDobro(2)?;

ATIVIDADE

As funções tem qual propósito básico?

A. Tornar o programa mais rápido.

B. criar blocos de códigos, segmentados e reutilizáveis.

C. cria uma maior complexidade.

D. substituir o JavaScript

ATIVIDADE

No código exemplificando o recurso de Closure, qual a finalidade do getNome() ?

A. exibir o valor da ?idade do aluno

B. exibir o valor da ?função mestre.

C. exibir o valor da variavel nome.

D. não possui função definida

REFERÊNCIA

MORRISON, M. Use a cabeça JavaScript. 5º Ed. Rio de Janeiro: Alta Books, 2012. 606 p.

OLIVIERO. C. A. J. Faça um site JavaScript orientado por projeto. 6º ed. São Paulo: Érica, 2010. 266 p.

ZAKAS, Nicholas C. JavaScript de alto desempenho. 8º Ed. São Paulo: Novatec, 2010. 245 p.