Funções

Vamos estudar outros exemplos de funções.

```
function calcularImc(peso,altura) {
    var imc = peso / (altura * altura);
    console.log(imc);
}

var meuPeso = 62;
var minhaAltura = 1.63;

calcularImc(meuPeso, minhaAltura);
```

Função calcularIMC – Fonte: Elaborado pela autora.

```
23.335466144755166
```

Console – Fonte: Elaborado pela autora.

O valor da variável IMC é calculado pela seguinte expressão matemática: peso dividido por altura vezes altura.

Lembre-se de que o cálculo entre os parênteses deve ser resolvido primeiro.

Note que, fora do escopo da função (fora das chaves), foram criadas duas variáveis com valor fixo. Na última linha, chamamos a função calcularIMC e passamos como parâmetros as variáveis criadas fora do escopo da função.

Nas linhas iniciadas por var, deverão ser passados os valores das variáveis (meuPeso = 62, minhaAltura = 1.63) para os parâmetros (peso, altura) para, assim, dar continuidade ao cálculo. Caso a forma passada fosse invertida (minhaAltura = 1.63, meuPeso = 62), essa seria a ordem utilizada pelos parâmetros, o que resultaria em uma mudança no valor da variável do IMC, pois o cálculo seria 1.63 / (62 * 62), tendo como resultado 4,240374609781478.

```
> function areaQuadrado(lado){
      var area = lado * lado;
      return area;
  }
  console.log(areaQuadrado(4));
  16
```

Função areaQuadrado – Fonte: Elaborado pela autora.

O nome da função areaQuadrado é passado com o parâmetro (lado). Podemos adicionar diversos parâmetros, basta separá-los por vírgulas. Criamos uma variável área, que armazenará o valor de lado vezes lado e retornará o valor da variável. *Return* é um comando padrão das funções, sua função é retornar o valor da variável pedida.

Na penúltima linha, na opção do console.log, passamos o valor do parâmetro (4), cujo resultado será 16, pois o parâmetro lado recebeu (4) como valor. Veja o resultado no console ao inspecionar a página na última linha.

```
var custoCarro = function(portas){
    var precoInicial = 10000;
    if(portas){
        return portas * precoInicial;
    } else {
        return 2 * precoInicial;
    }
}
console.log(custoCarro(400));
```

Função custoCarro – Fonte: Elaborado pela autora.

Para usar uma função, não é necessário apenas criar variáveis dentro do seu escopo. As variáveis podem ser criadas antes da função, como no exemplo acima. Você pode usar laços condicionais dentro da função, como mostrado.

Alguns tipos de função:

- ***Function constructor***: usada para criar objetos.

```
function multiply(x, y) {
    return x * y;
}
```

Exemplo de função *constructor* – Fonte: Elaborado pela autora.

- ***Function declaration***: atribuída a uma variável; trata-se de uma função anônima, ou seja, não possui nome.

```
var multiply = function(x, y) {
    return x * y;
};
```

Exemplo de função *declaration* – Fonte: Elaborado pela autora.

- ***Function expression*:** atribuída a uma variável e possui um nome.

```
var multiply = function func_name(x, y) {
    return x * y;
};
```

Exemplo de função *expression* – Fonte: Elaborado pela autora.

Exemplo de função para criar menu para mobile

HTML

```
<div class="menuMobile">
  <button onclick="myFunction()" class="dropbtn"><img src="img/bars.png" alt=""></button>
  <div id="subMenuMobile" class="subMenuItens">
    <ul class="menu">
      <a class="linkMenu" href="jogos.html">Jogos</a>
      <a class="linkMenu" href="hardware.html">Hardware</a>
      <a class="linkMenu" href="celulares.html">Celulares</a>
      <a class="linkMenu" href="pc-gamer.html">Pc gamer</a>
      <a class="linkMenu" href="contato.html">Contato</a>
      <a class="linkMenu login" href="login.html">Login</a>
    </ul>
  </div>
</div>
```

HTML – Fonte: Elaborado pela autora.

Em *button*, adicione o evento *onclick* e o nome da função, que será chamada do arquivo JavaScript. O seletor *onclick* determina que haverá uma resposta quando o usuário clicar no botão. Nesse caso, ao clicar, o id="subMenuItens", que está oculto, ficará visível.

CSS

```css
.dropbtn {border: none;cursor: pointer;height: 50px;}
.dropbtn img{height: 50px;}
.subMenuItens .menu {display:flex;}
.dropdown {position: relative;display: inline-block;}
.subMenuItens {display: none;z-index: 1;}
.subMenuItens a {color: black;padding: 12px 16px;text-decoration: none;display: block;}
.subMenuItens a:hover {background-color: #f1f1f1}
.show {display:block;}
```

CSS – Fonte: Elaborado pela autora.

JS

```javascript
function myFunction() {
    document.getElementById("subMenuMobile").classList.toggle("show");
}
```

JavaScript – Fonte: Elaborado pela autora.

A função chama o menu oculto pelo id, que no CSS está como display:none (oculto). Quando o usuário clicar (o *onclick* do HTML), o menu passará a ser visível (display:block, no CSS). O método *toggle* verifica a visibilidade do elemento.

Exemplo de função para criar carrossel

HTML

```html
<div class="slideshow-container">
  <div class="mySlides">
    <img src="https://www.w3schools.com/howto/img_nature_wide.jpg" style="width: 100%;">
  </div>
  <div class="mySlides">
    <img src="https://www.w3schools.com/howto/img_fjords_wide.jpg" style="width: 100%;">
  </div>
  <div class="mySlides">
    <img src="https://www.w3schools.com/howto/img_lights_wide.jpg" style="width: 100%;">
  </div>
  <div class="mySlides">
    <img src="https://www.w3schools.com/howto/img_mountains_wide.jpg" style="width: 100%;">
  </div>
  <a class="prev" onclick="plusSlides(-1)">&#10094;</a>
  <a class="next" onclick="plusSlides(1)">&#10095;</a>
</div>
<br>
```

HTML – Fonte: Elaborado pela autora.

CSS

```css
.slideshow-container {max-width: 1000px; position: relative;margin: auto;}
.prev,.next {cursor: pointer;position: absolute;top: 0;top: 50%;width: auto;
    margin-top: -22px;padding: 16px;color: white;font-weight: bold;
    font-size: 18px;transition: 0.6s ease;border-radius: 0 3px 3px 0;}
.next {right: 0;border-radius: 3px 0 0 3px;}
.prev:hover, .next:hover {background-color: rgba(0, 0, 0, 0.8);}
```

CSS – Fonte: Elaborado pela autora.

JS

```
function showSlides(n) {
  var i;
  var slides = document.getElementsByClassName("mySlides");
    if (n > slides.length) {slideIndex = 1}
    if (n < 1) {slideIndex = slides.length};
        for (i = 0; i < slides.length; i++) {
            slides[i].style.display = "none";
        }
  slides[slideIndex-1].style.display = "block";
}
```

JavaScript – Fonte: Elaborado pela autora.

A função chama a *class* das imagens (mySlides). A primeira condição a ser verificada é se a variável n é menor que a quantidades de slides; caso seja, a variável slideIndex passará a receber 1.

A segunda condição a ser verificada é se n é menor que um; caso seja, ele entra em um loop, no qual a estilização das demais imagens adquirem a característica de display:none e a atual, display:block.