```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="sass/css/function.css" />
<title>Regra @function</title>
</head>
<body>

<!--

            Regra de Estilo @function

        O que faz a Regra de Estilo @function?

    A regra de estilo @function do Sass é usada para trabalharmos com funções, ou seja, trechos de código que utilizam
    valores como: number, string, colors, lista de valores e etc, para gerar operações que resultem na saída de um
    determinado valor que desejamos utilizar no nosso código Sass.

    As funções são muito úteis para gerar abstrações complexas de serem implementadas, fazendo isso de uma maneira
    simples e tornando o código muito mais legível.


        Como utilizar @function?

    Para criar uma função devemos usar a regra @function, seguida pelo nome da função e os parênteses que receberão
    seus parâmetros, á partir daí damos início a função por através dos colchetes, onde colocamos as operações que irão
    acontecer sequencialmente. E ao final temos que usar a regra @return dentro da função para que ela retorne um
    resultado ao final da sua execução. Depois, para utilizar a função, basta chamarmos por ela normalmente, assim
    como chamamos as funções built-ins do CSS, desde que o arquivo módulo da função já tenha sido importado no
    documento atual que estivermos mexendo.

    Fazemos isso da seguinte forma:
```

Primeiro criamos uma função dentro de um módulo:

Arquivo: _calc.scss

```
@function sum($numbers...) {
    $sum: 0;
    @each $number in %numbers {
        $sum: %sum + %number;
}
@return $sum;
}
```

Acima temos uma função que soma o total de valores que forem passados a ela separados por vírgula. Perceba que, após a regra @function demos um nome á função, o nome “sum”, após o nome temos os parênteses, que resguardam os parâmetros que a função deverá receber, esses parâmetros serão alocados na variável $numbers que é um array que vai receber uma sequência de valores. E dentro da função nós declaramos uma variável chamada $sum, essa variável será usada futuramente na regra @return para apresentar o valor da somatória ao usuário.

A seguir, temos uma regra dentro da função, que é a regra @each – ou “cada” – essa regra faz com que, para cada valor índice dentro do array em $numbers, ela calcule esse valor somado ao total que estiver em $sum, ou seja, está fazendo a soma de todos os valores passados.

Ao final da função, temos uma regra @return, essa regra é obrigatória para que possamos entregar um resultado ao usuário final.

Importando o módulo para um arquivo scss:

Arquivo style.scss:

```
@use “calc”;

.width-total {
    Width: calc.sum(50px, 30px, 100px);
}
```

Veja que importamos o módulo “calc.scss” dentro do nosso Sass gerador do nosso css final. Lá dentro do arquivo Sass nós chamamos a função “sum” dentro do seletor “width-total” para calcular a largura total de um elemento segundo 3 parâmetros passados;

Resultado no arquivo final:

Arquivo Final:

```
.width-total {
    width: 180px;
}
```

Perceba que no arquivo final, o que aparecerá para ele é o valor já calculado e passado por através da regra @return dentro da própria função “sum”.

Cuidados ao nomear funções!!!

Quando damos nomes as nossa funções devemos tomar o cuidado de não escolher nomes que comecem com números, caracteres especiais ou que tenham o mesmo nome que uma função built-in do Sass, pois, podemos acabar usando um seletor universal para renomear nossos módulos e acabar designando uma função com o mesmo nome de uma função built-in, o que poderia gerar um conflito.

Além disso devemos tomar cuidado com o uso de hífen (-) e underline (_), afinal ambas são entendidas da mesma forma pelo Sass, então se tivermos uma função cujo nome é: “scale-color” e outra “scale_color”, para o Sass elas serão a mesma função.

Recomendação importante!!!

Embora alguns se sintam tentados a usar funções para definir formatação sobre um trecho de código, essa prática é fortemente desencorajada, pois, o papel principal das funções é gerar cálculos e apresentar resultados calculados, enquanto a definição de formatações é um trabalho dos mixins.

Por isso, é bom deixarmos cada coisa no seu lugar, @function deve ser usado somente para fazer cálculos e @mixin para definição de efeitos sobre os nossos códigos.

```
        A Regra @return:

    A regra @return indica o valor a ser usado como resultado da chamada de uma função. Só é permitido dentro de um
    corpo de @function, e cada @function deve terminar com um @return.

    Quando um @return é encontrado, ele termina imediatamente a função e retorna o resultado para o código solicitante.

    Podemos até utilizar mais de @return em casos especiais, como em casos de desvio condicional com @else por exemplo,
    porém o resultado sempre vai sair de um único @return.


-->

    <ul id="chat">

        <li>

            <div class="avatar">
                <img src="https://i.pravatar.cc/48" alt="Avatar">
            </div>
            <div class="message">
                <strong>John Smith</strong>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ea nemo eveniet necessitatibus, deleniti illo
                    corporis enim in fugiat, modi amet maxime totam est praesentium voluptates maiores aliquid fuga, id
                    tempore?
                    Lorem, ipsum dolor sit amet consectetur adipisicing elit. Delectus iusto saepe exercitationem
                    quibusdam dignissimos assumenda quis, aut beatae vero repellat dolor quaerat incidunt harum amet
                    ratione enim vitae molestiae accusamus!
                </p>
                <time>10 minutos atrás</time>
            </div>
            <div class="menu">
                <button type="button">
                    <img src="more_vert_black_24dp.svg" alt="More">
                </button>
```

```
            </div>

        </li>

        <li class="me">

            <div class="avatar">
                <img src="https://i.pravatar.cc/48?1" alt="Avatar"> <!--Para que as imagens viessem diferentes colocamos
                    "?1" ao final da URL-->
            </div>
            <div class="message">
                <strong>John Smith</strong>
                <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores voluptas quam enim neque ipsam
                    fugit.</p>
                <time>10 minutos atrás</time>
            </div>
            <div class="menu">
                <button type="button">
                    <img src="more_vert_black_24dp.svg" alt="More">
                </button>
            </div>

        </li>

    </ul>

</body>
</html>
```

MÓDULO SASS...

```
@use "sass:color"; //Módulo nativo do Sass para trabalhar com cores

//Função criada para inverter uma cor passada...
@function invert($color, $amount: 100%){ //Uma variavel que recebe a cor, e outra que define a intensidade da cor
    $invert: color.change($color, $hue: color.hue($color) + 180); /*hue é a matiz, o quando ela se aproxima do branco
```

```
                                                                    ou do preto, quando mais alto o valor, mais se
                                                                    aproxima do preto, quando mais baixo, do branco.
                                                                    Como a matiz é medida em 360°, a matiz de 180°
                                                                    equivale a inversão de uma cor.*/
    /*Perceba a variável $invert recebe a função change, que recebe a cor e a matiz, sendo que a matiz recebe a cor
    com o hue invertido por estar sendo calculado + 180, ou seja, a cor invertida.*/
    @return color.mix($invert, $color, $amount);
    /*Por fim temos o return, que retorna a cor, porém mixamos a cor já existente com o valor de intensidade*/
}
```

ARQUIVO SASS...

```
@use "bootstrap-forward" as *;

@use "invertColor";

*{
    box-sizing: border-box;
}

body {
    margin: 0;
    font-family: sans-serif;
    color: $cores-color;
}

#chat {
    display: flex;
    flex-direction: column;
    margin: 20px;
    padding: 0;
    list-style: none;
    li {
        $background-color: orangered; /*Perceba que a cor que está sendo passada é laranja*/
        display: flex;
        margin-bottom: 32px;
```

```scss
.avatar {
    padding: 0 16px;
    display: flex;
    align-items: flex-end;
    img{
        @include cantos-circulo;
        width: 48px;
    }
}
.message {
    flex: 1;
    background-color: invertColor.invert($background-color); /*Porém, como estamos usando invert, a cor foi
                                                               trocada por uma cor invertida*/
    padding: 16px;
    @include cantos-arredondado;
    border-bottom-left-radius: 0;
    strong {
        color: #242939;
    }
    p {
        font-size: 14px;
    }
    time {
        font-size: 12px;
        letter-spacing: 1px;
        opacity: .65;
    }
}
.menu {
    display: flex;
    align-items: center;
    button {
        border: none;
        background: none;
        outline: none;
        img {
```

```
                    width: 16px;
                    opacity: .5;
                    cursor: pointer;
                }
            }
        }
    }
    & .me {
        $background-color: $cores-primary;
        flex-direction: row-reverse;
        color: #fff;
        .message {
            background-color: $background-color;
            border-bottom-left-radius: $cantos-raio;
            border-bottom-right-radius: 0;
            ::selection {
                background: white;
                color: $cores-primary;
            }
        }
        strong {
            display: none;
        }
    }
}
```

ARQUIVO CSS FINAL...

```
@charset "UTF-8";
/*Veja que importamos os módulos e na importação já demos apelidos á eles*/
* {
  box-sizing: border-box;
}

body {
  margin: 0;
```

```
  font-family: sans-serif;
  color: #7f808c;
}

#chat {
  display: flex;
  flex-direction: column;
  margin: 20px;
  padding: 0;
  list-style: none;
}
#chat li {
  /*Perceba que a cor que está sendo passada é laranja*/
  display: flex;
  margin-bottom: 32px;
}
#chat li .avatar {
  padding: 0 16px;
  display: flex;
  align-items: flex-end;
}
#chat li .avatar img {
  border-radius: 50%;
  width: 48px;
}
#chat li .message {
  flex: 1;
  background-color: #00baff;
  /*Porém, como estamos usando invert, a cor foi
  trocada por uma cor invertida*/
  padding: 16px;
  border-radius: 8px;
  border-bottom-left-radius: 0;
}
#chat li .message strong {
  color: #242939;
```

```
}
#chat li .message p {
  font-size: 14px;
}
#chat li .message time {
  font-size: 12px;
  letter-spacing: 1px;
  opacity: 0.65;
}
#chat li .menu {
  display: flex;
  align-items: center;
}
#chat li .menu button {
  border: none;
  background: none;
  outline: none;
}
#chat li .menu button img {
  width: 16px;
  opacity: 0.5;
  cursor: pointer;
}
#chat .me {
  flex-direction: row-reverse;
  color: #fff;
}
#chat .me .message {
  background-color: #0176ff;
  border-bottom-left-radius: 8px;
  border-bottom-right-radius: 0;
}
#chat .me .message ::selection {
  background: white;
  color: #0176ff;
}
```

```css
#chat .me strong {
  display: none;
}

/*# sourceMappingURL=function.css.map */
```

RESULTADO NO NAVEGADOR...