```
<!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/use.css" />
<title>Regra @use</title>
</head>
<body>

<!--
            Regra de Estilo @use

        O que faz a Regra de Estilo @use?

    A regra de estilo @use carrega mixins, funções e variáveis de outras folhas de estilo Sass e combina CSS de várias
    folhas de estilo. Folhas de estilo que são carregadas com @use são chamadas de módulos. O próprio Sass já fornece
    alguns módulos integrados que são cheios de funções úteis para os desenvolvedores.


            Como utilizar o @use?

        Podemos utilizá-la das seguintes formas:

        •   Modo Simples:

        A forma simples de utilizar essa regra é:

            @use “nomeDoArquivo”

        Dessa forma, a regra vai carregar o nome do arquivo fornecido, ou a URL dele se ele não estiver na mesma pasta ou
        local.

        Atenção! Quando utilizamos um módulo que contém underline na frente, por exemplo: “_arquivo.scss” , não precisamos
        colocar o underline, em vez disso vamos colocar: “arquivo.scss”, sempre dentro das aspas.
```

Outro detalhe importante sobre a regra @use é que todos os estilos carregados por através de @use vão fazer parte do arquivo original que estamos trabalhando. Diferente do @import onde somente as partes do arquivo seriam coladas ao arquivo atual.

Veja como utilizá-la logo abaixo:

Criando arquivo scss:

Arquivo: _corners.scss

```
$radius: 3px;

@mixin rounded {
    border-radius: $radius;
}
```

Perceba que aqui criamos um mixin com o nome “ronded” com um valor de 3px que pega esse valor da variável $radius;

Carregando arquivo com @use:

Arquivo style.scss:

```
@use “src/corners”;
.button {
    @include corners.rounded;
    Padding: 5px + corners.$radius;
}
```

Aqui nós fizemos o carregamento do arquivo “corners”. Usando a regra @include chamamos um mixin específico que é o “rounded”, por através do “rounded” conseguimos usar o valor da variável “$radius” somado ao valor de 5px. Mas não apenas isso, o include também copiou a propriedade border-radius que havia dentro do mixin, aqui não mostra por que essa formatação já está inclusa no @include, mas, poderemos ver isso claramente no nosso arquivo final;

Resultado no arquivo final:

Arquivo Final:

```
.button {
    border-radius: 3px;
    padding: 8px;
}
```

Perceba que no arquivo final, vamos ter o border-radius passado por através do @include e o padding modificado já com o valor de 5px + 3px da variável $radius. Incrível não é?

- Usando com namespaces:

Outra vantagem da regra @use é que podemos usar um alias para referenciar a um arquivo externo invés de usar o próprio nome do arquivo, a esse alias damos o nome de “namespaces”.

Veja como podemos utilizá-lo:

Criando arquivo scss:

Arquivo: _corners.scss

```
$radius: 3px;

@mixin rounded {
    border-radius: $radius;
}
```

Carregando arquivo com @use usando namespace:

Arquivo style.scss:

```
@use “src/corners” as c;  //Aqui usamos o namespace
.button {
    @include c.rounded;
```

```
                Padding: 5px + c.$radius;
            }
```

Perceba que invés de usarmos o nome “corners”, damos ao arquivo o apelido de “c”, o que facilita a nossa escrita e também a referência a um determinado arquivo que desejamos retirar um elemento;

Resultado no arquivo final:

Arquivo Final:

```
.button {
    border-radius: 3px;
    padding: 8px;
}
```

Perceba que o resultado seria o mesmo do que se usássemos o método tradicional.

OBSERVAÇÕES! Cuidado ao usar * como namespace, geralmente usamos * para se referir a todas as bibliotecas existentes dentro de uma pasta, porém isso não é uma boa prática de programação, pois poderíamos acabar esbarrando com o mesmo problema que tínhamos no @import, que é encontrar variáveis com nomes iguais.

Como usar extensões de arquivo na Regra @use?

Quando carregamos um arquivo dentro da regra @use por através da sua URL, o Sass consegue encontrar o arquivo intuitivamente sem precisar de que a extensão do arquivo esteja explicitada junto ao nome dele. Por exemplo:

Em vez de colocar: @use “arquivo.scss” ou @use “arquivo.sass”, podemos colocar somente @use “arquivo”. Mesmo se colocarmos somente o nome do arquivo, o Sass vai entender intuitivamente que deve procurar somente por arquivos nas extensões em que ele trabalha.

Usando Load Paths na regra @use:

O que são load paths? Load paths são caminhos que o sistema usa para encontrar um determinado arquivo dentro de um diretório que está no sistema. Para tanto, o Sass utiliza somente URL’s e não os caminhos convencionais que o Sistema Operacional utiliza, isso faz com que sempre que precisarmos adicionar um load path tenhamos que usar somente a barra

normal “/”, mesmo em sistemas onde o seu load path seria com barra invertida (como o Windows por exemplo), não há necessidade de fazer isso no Sass, ele utilizará somente barras normais.

Mas além disso, o Sass traz outra facilidade, que é podermos definir um caminho de carregamento padrão para os nossos arquivos Sass, isso faz com que caso precisemos carregar um arquivo que está dentro deste caminho não vamos precisar escrever o caminho completo, basta escrever o nome do arquivo – não precisa de colocar a extensão, como vimos anteriormente – e o Sass vai localizar o arquivo correto.

Por exemplo, digamos que passamos para o Sass o caminho: node_modules/susy/sass e usamos como caminho na regra @use a URL “susy”, o Sass já sabe que deverá carregar o arquivo: node_modules/susy/sass/susy.scss

Mas Atenção!! Os módulos sempre serão carregados em relação ao caminho do arquivo atual primeiro, isso significa que, somente se o Sass não encontrar nenhum arquivo “.css”, “.scss” ou “sass” com o nome “susy” dentro do diretório atual, é que ele vai procurar o arquivo no load path passado por nós anteriormente.

Sobre Partials:

O que é um partial?
Um partial é um módulo Sass que não desejamos gerar um arquivo CSS final á partir dele, ele é um módulo usado somente como referência para arquivos Sass que desejem utilizar a sua folha de estilos para gerar os seus próprios arquivos finais.

Para criar partials utilizamos o underline antes do nome do arquivo, por exemplo: _arquivo.scss. Quando o Sass gera arquivos assim, ele já sabe que não deve compilar arquivos que tenham o underline na frente do nome do arquivo.

Porém, quando carregamos um partial dentro de um arquivo Sass por através do @use não precisamos escrever o underline, podemos escrever só o nome do arquivo mesmo que vai dar tudo certo.

Grande Vantagem do uso da Regra @use:

Como @use adiciona namespaces aos nomes dos membros, é seguro escolher nomes muito simples como $radius ou $width como nome das nossa variáveis ao escrever as nossas folhas de estilo. O que é bem diferente da regra @import do CSS, que nos encoraja a escrever nomes longos como $mat-corner-radius só para evitar conflitos com outras bibliotecas que possam ter o mesmo nome declarado a uma variável. Como as nossas bibliotecas terão um namespace, quando o arquivo puxar um determinado nome de variável o Sass vai saber que aquele nome só se aplica áquela biblioteca em questão.

```
-->

    <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>
```

ARQUIVO PARTIAL...

```
$raio: 8px;

@mixin arredondado {
    border-radius: $raio;
}

@mixin circulo {
    border-radius: 50%;
}
```

ARQUIVO SCSS...

```
@use "cantos" as c; /*Perceba que importamos um módulo sem precisar de referenciar todo o load path e demos a ele um
                        apelido de "c" */
```

```scss
$color: #7f808c;
$primary: #0176ff;

*{
    box-sizing: border-box;
}

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

#chat {
    display: flex;
    flex-direction: column;
    margin: 20px;
    padding: 0;
    list-style: none;
    li {
        $background-color: #f5f6fa;
        display: flex;
        margin-bottom: 32px;
        .avatar {
            padding: 0 16px;
            display: flex;
            align-items: flex-end;
            img{
                @include c.circulo; /*Aqui fizemos um include para o valor de círculo do módulo c*/
                width: 48px;
            }
        }
        .message {
            flex: 1;
            background-color: $background-color;
            padding: 16px;
```

```
            @include c.arredondado; /*Um include para o valor da borda arredondada da biblioteca c*/
            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: $primary;
        flex-direction: row-reverse;
        color: #fff;
        .message {
            background-color: $background-color;
            border-bottom-left-radius: c.$raio; /*E somente puxamos o valor da variável de c*/
```

```
            border-bottom-right-radius: 0;
            ::selection {
                background: white;
                color: $primary;
            }
        }
        strong {
            display: none;
        }
    }
}
```

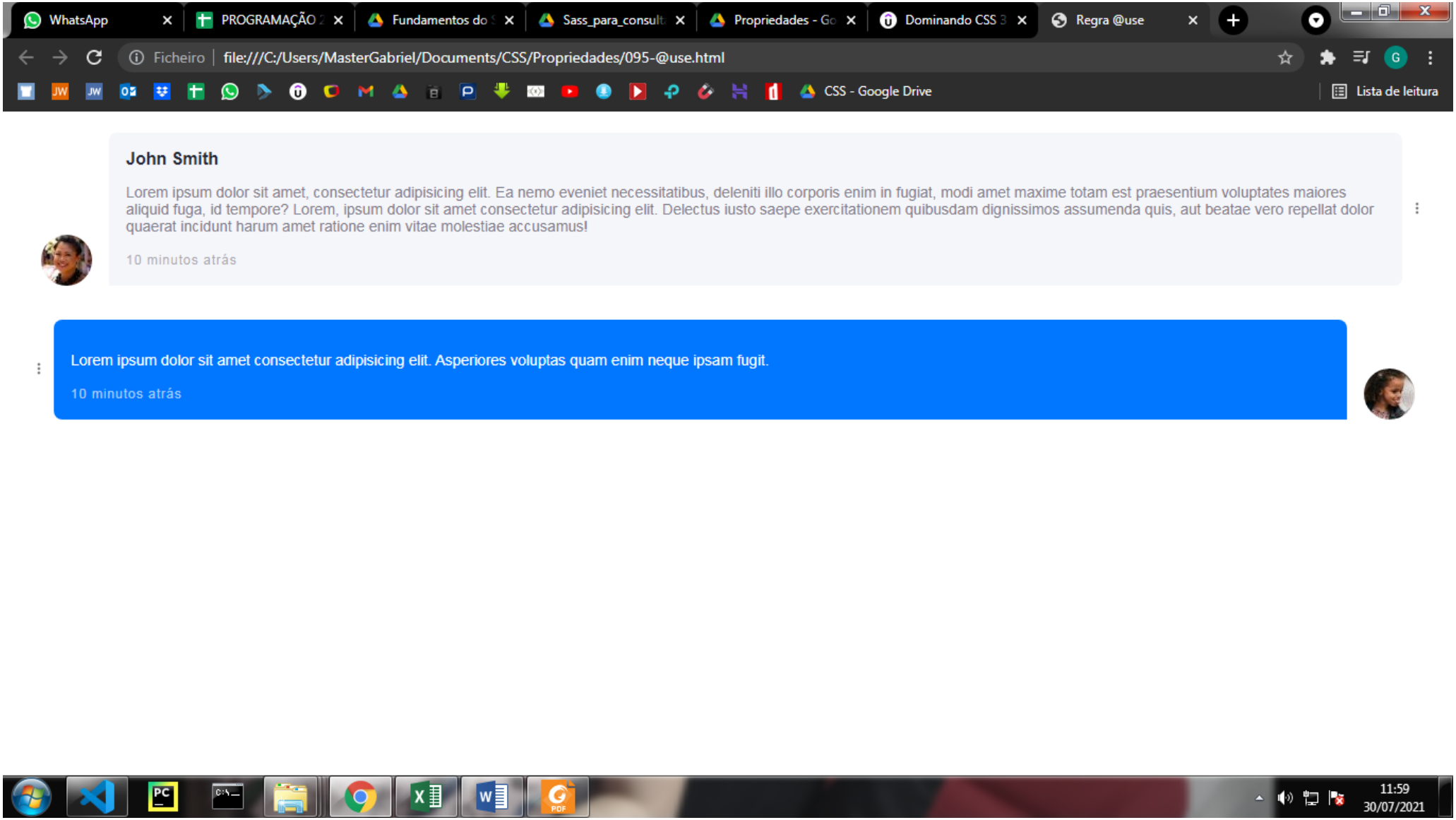
ARQUIVO CSS FINAL...

```
@charset "UTF-8";
/*Perceba que importamos um módulo sem precisar de referenciar todo o load path e demos a ele um
    apelido de "c" */
* {
  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 {
  display: flex;
```

```
  margin-bottom: 32px;
}
#chat li .avatar {
  padding: 0 16px;
  display: flex;
  align-items: flex-end;
}
#chat li .avatar img {
  border-radius: 50%;
  /*Aqui fizemos um include para o valor de círculo do módulo c*/
  width: 48px;
}
#chat li .message {
  flex: 1;
  background-color: #f5f6fa;
  padding: 16px;
  border-radius: 8px;
  /*Um include para o valor da borda arredondada da biblioteca c*/
  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;
  /*E somente puxamos o valor da variável de c*/
  border-bottom-right-radius: 0;
}
#chat .me .message ::selection {
  background: white;
  color: #0176ff;
}
#chat .me strong {
  display: none;
}

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

RESULTADO NO NAVEGADOR LOGO ABAIXO...


WhatsApp
PROGRAMAÇÃO
Fundamentos do
Sass_para_consult
Propriedades - Go
Dominando CSS 3
Regra @use
Ficheiro | file:///C:/Users/MasterGabriel/Documents/CSS/Propriedades/095-@use.html
CSS - Google Drive
Lista de leitura
John Smith
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!
10 minutos atrás
Lorem ipsum dolor sit amet consectetur adipisicing elit. Asperiores voluptas quam enim neque ipsam fugit.
10 minutos atrás
11:59
30/07/2021