Objetivos:

I. Media queries;

II. Sintaxe das media queries.

I. Media queries

O principal objetivo das media queries é criar layouts flexíveis que se ajustem dinamicamente às características do dispositivo. Elas nos permitem adaptar o estilo de uma página web com base nas características do dispositivo ou na largura da viewport do usuário, garantindo uma experiência consistente e agradável em diferentes dispositivos, desde desktops até dispositivos móveis. Isso é essencial para proporcionar uma experiência de usuário otimizada em uma ampla variedade de dispositivos e tamanhos de tela.

Para mais detalhes

https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_media_queries/Using_media_queries.

II. Sintaxe das media queries

Uma media query consiste de um `media type` e pelo menos uma expressão que limita o escopo das folhas de estilo usando `media features`:

```
@media tipo-de-mídia and (característica de mídia) {
   Corpo formado por folhas de estilo
}
```

A estrutura básica de uma media query começa com a regra `@media`, seguida pelo `tipo de mídia (media type)` e pelo menos uma expressão que limita o escopo das folhas de estilo usando `media features (características de mídia)`. Quando a media query é verdadeira, os estilos ou folhas de estilos definidas do corpo são aplicados em cascata.

Tipos de mídia (media types):

- `all`: aplica-se a todos os dispositivos. O estilo será aplicado a qualquer tipo de mídia:

```
@media all {
  body {  background-color: lightgreen; }
}
```

- `screen`: aplica-se a dispositivos com tela (desktops, tablets e smartphones). O estilo será aplicado a somente a exibição em dispositivos com tela:

```
@media screen {
  body {  background-color: lightgreen; }
}
```

- `print`: aplica-se à visualização e impressão de documentos. O estilo será aplicado somente para impressão:

```
@media print {
  body {
    font-style: italic;
    color: red;
 }
}
```

- speech: aplica-se a leitores de tela.

Características de mídia (media features): especificam condições como largura, altura, orientação e resolução. A seguir as características mais usadas:

- width: especifica a largura da viewport. O estilo será aplicado quando a viewport tiver exatamente 800px de largura:

```
@media (width: 800px) {
  body {  background-color: lightgreen; }
}
```

- height: especifica a altura da viewport. O estilo será aplicado quando a viewport tiver exatamente 400px de altura:

```
@media (height: 400px) {
  body {  background-color: lightgreen; }
}
```

- orientation: orientação do dispositivo no modo paisagem (landscape - o visor é mais largo do que mais alto) ou retrato (portrait - o visor é mais alto do que mais largo). O estilo com cor de fundo ciano será aplicado quando a tela estiver na orientação paisagem e o estilo com a cor de fundo azul claro será aplicado quando a tela estiver na orientação retrato:

```
@media (orientation: landscape) {
  body {  background-color: cyan; }
}
@media (orientation: portrait) {
  body {  background-color: lightblue; }
}
```

- min-width e max-width: especificam uma faixa para a largura. O estilo será com cor de fundo laranja será aplicado quando a tela tiver largura de até 700px e o estilo com a cor de fundo ciano será aplicado quando a tela tiver largura de 900px ou mais:

```
@media (min-width: 900px) {
  body {  background-color: cyan; }
}
@media (max-width: 700px) {
  body {  background-color: orange; }
}
```

Combinação de características de mídia: os operadores lógicos (and, or e not) podem ser usados para criar condições complexas:

- Operador not: o estilo a seguir será aplicado para telas que não possuem largura mínima de 600px:

```
@media screen and ( not (min-width: 600px)) {
  body { background-color: lightgreen;  }
}
```

- Operador and: o estilo a seguir será aplicado para viewport com largura mínima de 600px e largura máxima de 700px, ou seja, telas no intervalo [600,700px]:

```
@media screen and (min-width: 600px) and (max-width: 700px) {
  body { background-color: lightgreen;  }
}
```

Hierarquia: as regras CSS dentro de uma media query substituem as regras fora dela, seguindo a hierarquia normal de estilos. No exemplo a seguir o estilo definido no corpo da média query substituíra o estilo definido fora, quando a media query se tornar verdadeira:

```
body {
  background-color: yellow;
}
@media (min-width: 700px) and (max-width: 900px) {
  body {  background-color: orange; }
}
```

O resultado da media query é verdadeiros se o media type especificado corresponde ao tipo do documento exibido no dispositivo e todas as expressões são verdadeiras. No exemplo a seguir não existe regra para largura de tela superior a 700px, então será usado o estilo definido fora das media queries:

```
body { background-color: yellow; }
```

Quando a largura de tela estiver entre 601 e 700px, a query `screen and (max-width: 700px)` é verdadeira e, desta forma, o estilo a seguir substituíra o estilo definido fora das queries:

```
body {  background-color: orange; }
```

Quando a largura de tela estiver em 600px ou menos, a query `screen and (max-width: 600px)` é verdadeira e, desta forma, o estilo a seguir substituíra o estilo definido fora das queries:

```
body {  background-color: red; }
```

O exemplo a seguir utiliza a largura da viewport para definir o estilo aplicado:

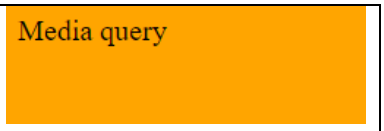
```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport"
```

Tela com largura de tela > 700px:

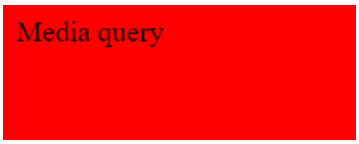
Media query

Tela com largura de tela <= 700px:

```
        content="width=device-width, initial-scale=1" />
    <style>
      body { background-color: yellow; }
      @media screen and (max-width: 700px) {
        body {  background-color: orange; }
      }
      @media screen and (max-width: 600px) {
        body {  background-color: red; }
      }
    </style>
  </head>
  <body>
    <div>Media query</div>
  </body>
</html>
```



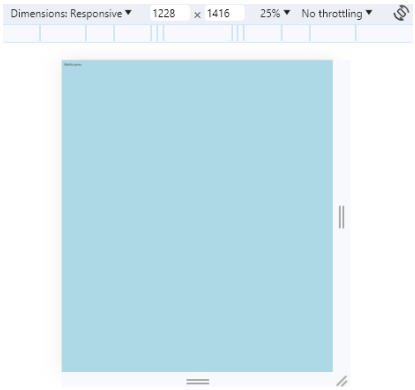
Tela com largura de tela <= 600px:



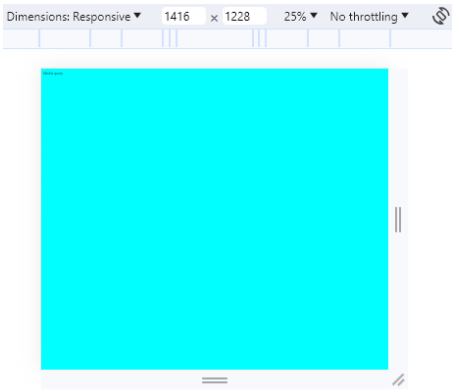
O exemplo a seguir utiliza a orientação da tela para definir o estilo aplicado:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport"
        content="width=device-width, initial-scale=1" />
    <style>
      @media screen and (orientation: landscape) {
        body {  background-color: cyan; }
      }
      @media screen and (orientation: portrait)  {
        body {  background-color: lightblue; }
      }
    </style>
  </head>
  <body>
    <div>Media query</div>
  </body>
</html>
```

Orientação portrait:



Orientação landscate:



O exemplo a seguir utiliza o tipo de mídia para definir o estilo aplicado:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport"
```

Exibição em tela:

```
        content="width=device-width, initial-scale=1" />
    <style>
      @media screen {
        body {
          background-color: #555;
          color: #fff;
        }
      }
      @media print {
        body {
          font-style: italic;
          color: red;
        }
      }
    </style>
  </head>
  <body>
    <div>Media query</div>
  </body>
</html>
```

Exibição para imprimir:

Media query

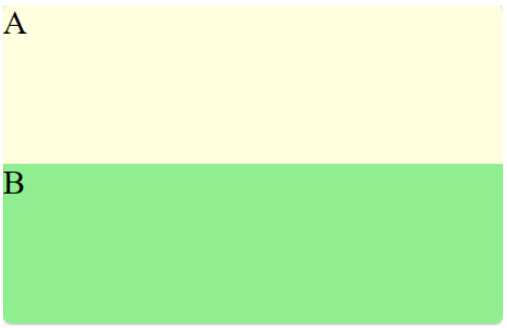
Exercícios

Utilize as imagens fornecidas para fazer os exercícios.

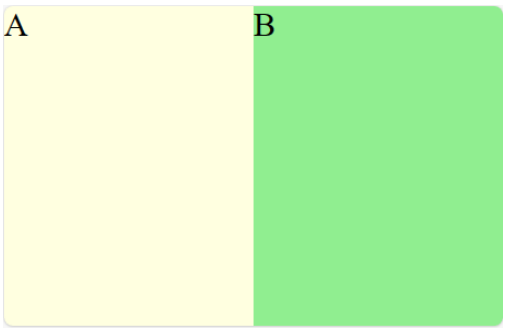
Veja o vídeo se tiver dúvidas nos exercícios: https://youtu.be/56pag2kvgHk

Exercício 1: O código a seguir produz as seguintes situações de acordo com o valor da propriedade flex-direction da classe container.

Com a propriedade flex-direction: column



Com a propriedade flex-direction: row



Adicionar uma media query para que o valor da propriedade flex-direction seja column quando a visualização estiver na orientação portrait (retrato) e que o valor da propriedade flex-direction seja row na orientação landscape (paisagem).

Código do exercício:

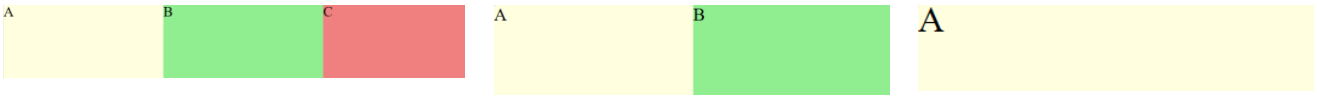
```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
```

```
    <title>Exercício 1</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      body {
        display: flex;
        margin: 0px;
        height: 100vh; /* ocupa a altura total da viewport */
      }
      .container {
        display: flex;
        flex: 1; /* ocupa toda a altura/largura disponível */
        flex-direction: row; /* utilize column para testar o layout na coluna */
      }
      .um {
        flex: 1;
        font-size: 30px;
        background-color: lightyellow;
      }
      .dois {
        flex: 1;
        font-size: 30px;
        background-color: lightgreen;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="um">A</div>
      <div class="dois">B</div>
    </div>
  </body>
</html>
```

Exercício 2: O código a seguir produz o seguinte resultado independente da largura da viewport.

Adicionar media queries para que o elemento com a classe tres seja retirado do layout quando a largura da viewport for inferior a 900px e que o elemento com a classe dois seja retirado do layout quando a largura da viewport for inferior a 600px.

Largura >=900px: Largura <900px: Largura de até 600px:



Adicionar uma media query para que o valor da propriedade flex-direction seja column quando a visualização estiver na orientação portrait (retrato) e que o valor da propriedade flex-direction seja row na orientação landscape (paisagem). Dica: use a propriedade `display: none` para retirar o elemento do layout.

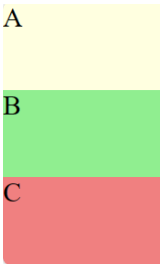
Código do exercício:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Exercício 2</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      body {
        display: flex;
        margin: 0px;
        height: 100vh;
      }
      .container {
        display: flex;
        flex: 1;
        flex-direction: row;
      }
      .um {
        flex: 1;
        font-size: 30px;
        background-color: lightyellow;
      }
      .dois {
        flex: 1;
        font-size: 30px;
        background-color: lightgreen;
      }
      .tres {
        flex: 1;
        font-size: 30px;
        background-color: lightcoral;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="um">A</div>
      <div class="dois">B</div>
      <div class="tres">C</div>
    </div>
  </body>
```

```
</html>
```

Exercício 3: Alterar o código do Exercício 2 para que os elementos sejam apresentados nos seguintes formados de acordo com a largura da viewport.

Largura <=800px:



Largura >800px:

Exercício 4: Alterar o código do Exercício 3 para que o tamanho da fonte seja de 14px para larguras da viewport de até 700px e o tamanho da fonte seja de 20px para larguras de até 1000px. Nas demais larguras deverá ser mantido o tamanho de fonte original, que é 30px.

Exercício 5: Adicionar uma media query no código a seguir para produz as seguintes situações de acordo com a largura da viewport.

Com largura <= 800px:

Com largura >800px:

Código do exercício:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Exercício 5</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      body {
        display: flex;
        margin: 0px;
```

```
        height: 100vh; /* ocupa a altura total da viewport */
      }
      .container {
        display: flex;
        flex: 1; /* ocupa toda a altura/largura disponível */
        flex-direction: row;
        justify-content: space-around; /* alinhamento horizontal dos filhos */
        align-items: center; /* alinhamento vertical dos filhos */
      }
    </style>
  </head>
  <body>
    <div class="container">
      <img src="./imagens/Fatec.png" alt="" width="87" height="41" />
      <img src="./imagens/CPS.png" alt="" width="85" height="56" />
      <img src="./imagens/Brasao.png" alt="" width="81" height="54" />
    </div>
  </body>
</html>
```