Objetivo:

Introduzir o Flexbox e Media Queries

1. O que é o Flexbox?

O Flexbox é uma técnica de layout muito poderosa e popular que permite posicionar e organizar elementos em um contêiner de forma flexível e responsiva. Com o Flexbox, podemos alinhar, distribuir e redimensionar elementos em diferentes tamanhos de tela.

2. Funcionamento Básico do Flexbox.

Para criar um layout utilizando o Flexbox, é necessário definir um contêiner que será o pai dos elementos flexíveis. Isso é feito definindo a propriedade ***display*** do elemento pai como ***flex***.

```
display: flex;
```

Assim que essa propriedade é definida, imediatamente os elementos descendentes modificam para o comportamento padrão do "***flex***", que será visto a seguir. Esse comportamento é diferente, por exemplo, do layout estilo "***grid***" (visto no Exercício 3) que não aplica efeito imediato.

2.1 – Eixos

Para trabalharmos com modelo Flexbox precisamos nos preocupar com os dois eixos, o **principal** e **transversal**. Existem propriedades diferentes que são utilizadas para controlar cada um desses e irão definir o posicionamento dos elementos dentro do contêiner marcado como "***flex***".

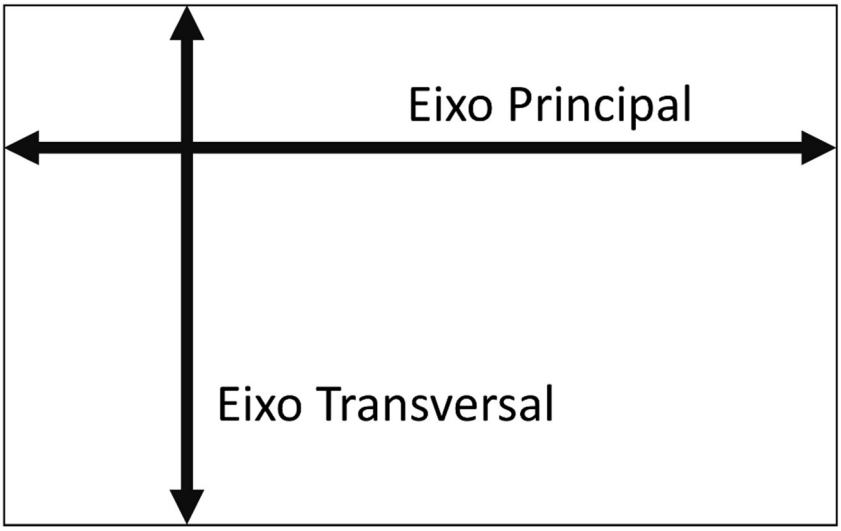

Figura 1

Ao declararmos um elemento com a propriedade “flex”, por padrão, o conteúdo é posicionado em linha, da esquerda para a direita. A propriedade utilizada para esse controle é a **“flex-direction”**, que pode assumir os seguintes valores:

```
/* Valor padrão. O eixo principal é horizontal, da esquerda para a direita */
flex-direction: row;
/* O eixo principal é horizontal, da direta para a esquerda */
flex-direction: row-reverse;
/* O eixo principal é vertical (coluna), de cima para baixo */
flex-direction: column;
/* O eixo principal é vertical (coluna), de baixo para cima */
flex-direction: column-reverse;
```

Para compreendermos esses valores, considere o código HTML abaixo:

```
<h1>Eu sou Flexbox</h1>
<section>
    <div id="div1">1</div>
    <div id="div2">2</div>
    <div id="div3">3</div>
    <div id="div4">4</div>
    <div id="div5">5</div>
</section>
```

A estilização adota uma cor de fundo para cada ID. O contêiner, nesse exemplo, é o elemento <section>. Portanto ele precisa ter a propriedade ***display*** com o valor ***“flex”.***

```
section{
    border: 3px solid black;
    width: 90%;
    height: 50vh;
    margin: 0 auto;
    display: flex;
}
```

Antes de atribuirmos o valor “flex” para a propriedade “display”, a página irá renderizar como mostrado na Figura 2

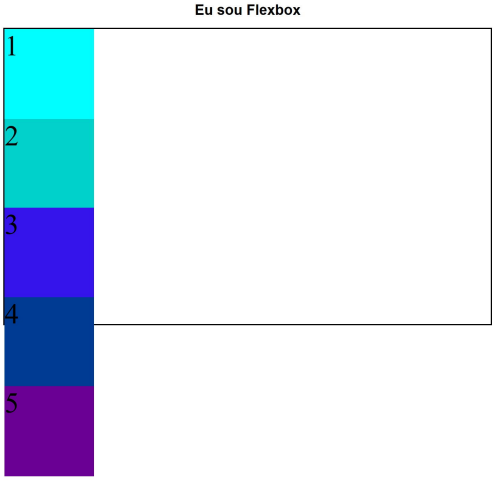

Figura 2

Após atribuirmos o valor “flex” para a propriedade “display”, a página irá renderizar as lado a lado, ocupando uma linha, com a primeira a esquerda e a última mais a direita. Esse é o valor padrão para o eixo principal, isto é, `flex-direction: row;`

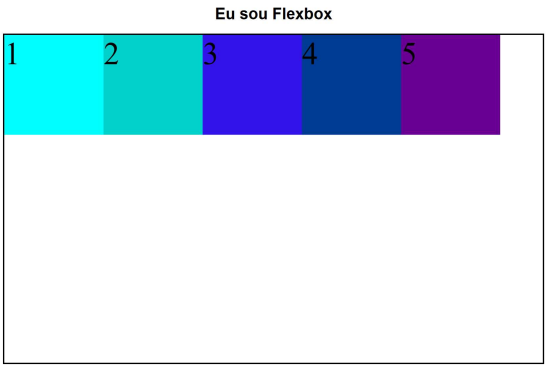

Figura 3

Se trocarmos para `flex-direction: row-reverse;` mantemos como “row” (linha) mas iremos inverter o sentido, como mostrando na Figura 4.

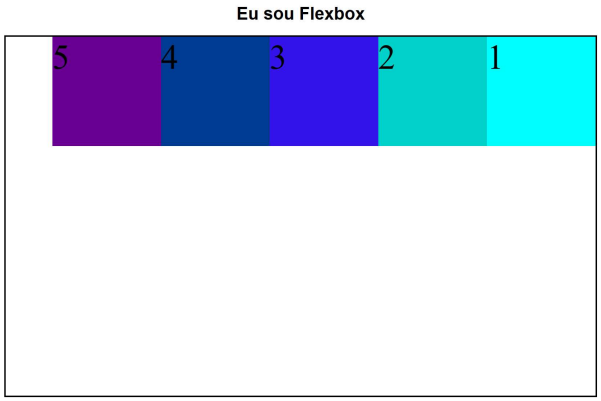

Figura 4

Se trocarmos para `flex-direction: column;` o eixo principal passa a ser a coluna, iniciando de cima para baixo, como mostrando na Figura 5.

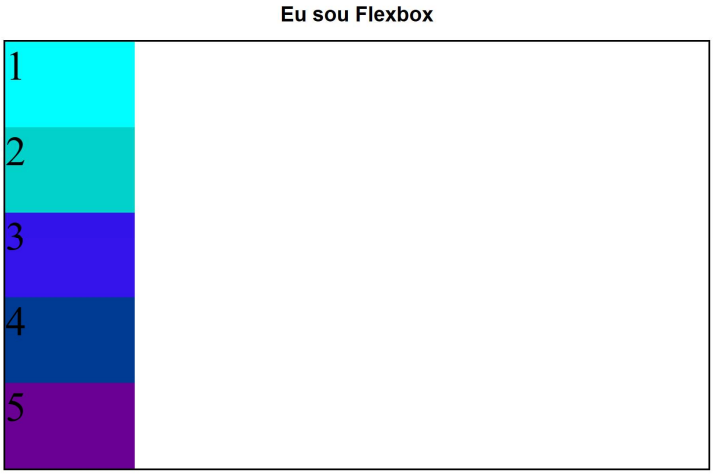

Figura 5

Se trocarmos para `flex-direction: column-reverse;` o conteúdo será invertido, de baixo para cima.
O valor da propriedade `flex-direction` irá influenciar também o eixo-transversal, pois este é perpendicular ao eixo-principal. Se for atribuído `row` para a propriedade `flex-direction`, o eixo transversal será uma coluna.

2.2 – Alinhando os elementos no eixo principal

Para alinhar e espaçar os elementos no eixo principal, usamos a propriedade `justify-content`. Seu comportamento irá depender de como a propriedade `flex-direction` foi definida. Para `justify-content`, podemos ter os seguintes valores.

```
/* Alinhamento padrão, com os elementos posicionados de acordo com o início do eixo
principal  */
justify-content: flex-start;
/* Elementos posicionados de acordo com o fim do eixo principal  */
justify-content: flex-end;
/* Elementos centralizados no eixo principal  */
justify-content: center;
/* Elementos com o espaço distribuído ao redor  */
justify-content: space-around;
/* Elementos com o espaço distribuído entre os elementos  */
justify-content: space-between;
/* Elementos com o espaço distribuído igualmente entre os elementos  */
justify-content: space-evenly;
```

As Figuras 6-8 considera o `flex-direction` como “row” e o `justify-content` é modificado de acordo com os valores:

```
justify-content: flex-end;
```

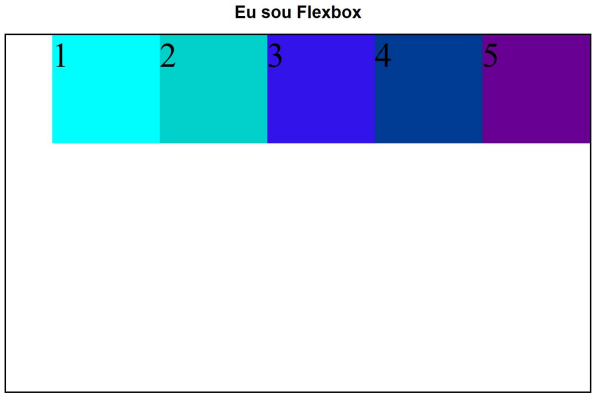

Figura 6

```
justify-content: center;
```

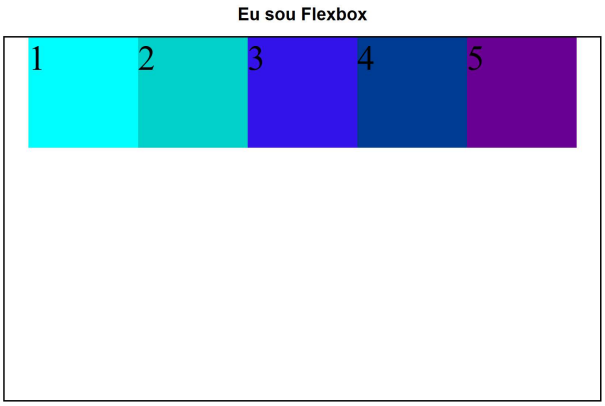

Figura 7

```
justify-content: space-around;
```

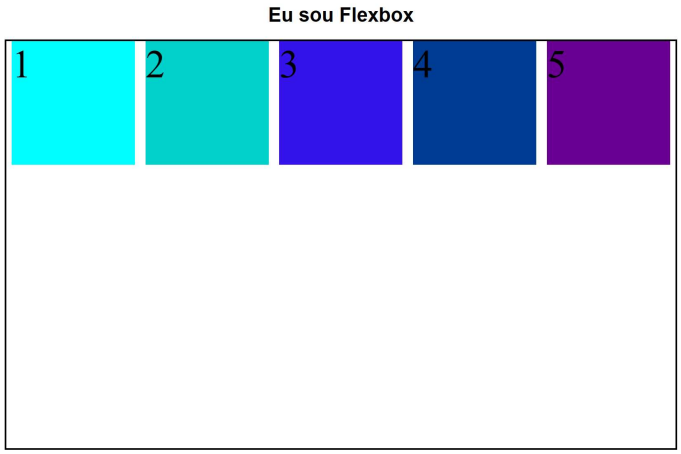

Figura 8

Trocando `flex-direction: row-reverse;` como `justify-content: center;` Temos o resultado na Figura 9

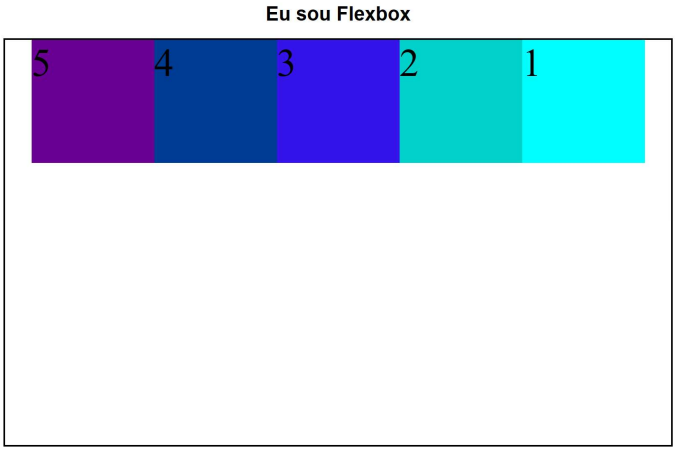

Figura 9

Experimente com os valores e veja os resultados. Lembre-se que o conteúdo ficará alinhado (justify-content) considerando o eixo-principal (flex-direction).

2.3 – Wrap dos elementos.

A propriedade “flex-wrap” define se os elementos são forçados a ficarem na mesma linha ou se podem ser quebradas em várias linhas. Importante observar que o sentido que os elementos serão “quebrados” segue o eixo transversal. Por isso a “linha” pode ser uma coluna dependendo do eixo principal. A propriedade pode assumir três valores, apresentados a seguir:

```
/* Valor padrão. Os elementos não passam para proxima linha/coluna */
flex-wrap:nowrap ;
/* Os elementos passam para proxima linha/coluna na mesma direção do eixo transversal */
flex-wrap: wrap;
/* Os elementos passam para proxima linha/coluna na direção contrária ao eixo transversal
flex-wrap: wrap-reverse;
```

2.4 – Alinhamento de elementos no eixo transversal

Para alinhar os elementos de um “flex” container, ao longo do eixo transversal, usamos a propriedade `align-items`. A propriedade `justify-content`, alinha os elementos ao longo do eixo principal. Ambas as propriedades possuem valores parecidos.

```
    /* Valor padrão. Alinha os elementos a partir do início do eixo transversal  */
    align-items: flex-start;
    /* Alinha os elementos a partir do fim do eixo transversal  */
    align-items: flex-end;
    /* Centraliza os elementos no eixo transversal  */
    align-items: center;
    /* Alinhas os elementos no eixo transversal de acordo com a base do conteúdo  */
    align-items: baseline;
```

Quando se tem múltiplas colunas e/ou linhas usamos a propriedades `align-content` para controlar o espaçamento. É necessário ter algum tipo de "wrap" ativado.

`align-self`

3. Media Queries.

Permite modificar a estilização dependendo de características como tamanho de tela e tipo de dispositivo.

Por exemplo, se desejarmos modificar a cor do elemento <h1> quando a tela estiver exatamente com 800px, podemos utilizar a seguinte sintaxe:

```
@media (width: 800px){
    h1{
        color: purple;
    }
}
```

Podemos ver o tamanho exato da tela utilizando a inspeção e acompanhando a largura do elemento <html>.
Esse exemplo é bastante limitado e só fará efeito com o valor exato da largura especificada.
Para especificarmos, por exemplo, que o <h1> deverá ser roxo quando a tela tiver 800px ou mais, podemos usar a sintaxe:

```
@media (min-width: 800px){
    h1{
        color: purple;
    }
}
```

Alternativamente, se quisermos mudar a propriedade somente enquanto a tela não atingir 800px, podemos usar a sintaxe:

```
@media (max-width: 800px){
    h1{
        color: purple;
    }
}
```

É possível combinarmos condições com operadores lógicos. Por exemplo, se desejarmos modificar uma propriedade com a tela entre 600px e 800px, podemos usar a seguinte sintaxe:

```
@media (min-width: 600px) and (max-width:800px){
    h1{
        color: purple;
    }
}
```

O “min-width” define o tamanho mínimo e o “max-width” define o tamanho máximo.

É possível também escrevermos as “queries” de forma separada, mas é necessário tomar precaução para não criar regras conflitantes, que sobrescrevem outras.

Exemplo: Para definirmos diferentes condições que irão modificar a cor do elemento <h1> dependendo do tamanho máximo da tela, podemos criar as regras iniciando pelas telas maiores até as menores utilizando “max-width” ou podemos começar pelas telas menores até maiores utilizando “min-width”.

O código a seguir, não terá o efeito desejado e o texto será verde até completar 600px

```
@media (max-width:400px){
    h1{
        color: purple;
    }
}

@media (max-width:600px){
    h1{
        color: green;
    }
}
```

Para corrigir, é necessário inverter:

```
@media (max-width:600px){
    h1{
        color: purple;
    }
}

@media (max-width:400px){
    h1{
        color: green;
    }
}
```

Alternativamente, podemos iniciar do menor, mas utilizando “min-width”.