Desenvolvimento WEB I

Fatec Zona Leste

Centro Paula Souza

PROFº DRº FRANCISCO DOUGLAS. L ABREU

MSC E PHD EM ENGENHARIA BIOMÉDICA
ESP. EM ANÁLISE DE DADOS

FRANCISCO.ABREU@FATEC.SP.GOV.BR

- **SELETORES**
  - TAG
  - CLASSE
  - IDENTIFICADOR
  - UNIVERSAL
- **PREENCHIMENTOS**
  - MARGIN E PADDING
- **DISPLAY**
  - BLOCK
  - INLINE
  - INLINE-BLOCK
  - FLEX-GRID
- **POSITION**
  - STATIC
  - RELATIVE
  - ABSOLUTE
  - FIXED
- **FLEXBOX**
  - JUSTIFY-CONTENT
  - ALIGN ITENS

AULA 10

CSS part II

SELETORES

CSS

#UTILIZANDO SELETORES

- Seletores no CSS são padrões **usados para selecionar elementos HTML** aos quais você deseja aplicar estilos.

- São fundamentais **para a estilização e formatação** de páginas da web. Com os seletores, você pode direcionar elementos específicos ou grupos de elementos para **aplicar regras de estilo, como cores, tamanhos de fonte, margens, espaçamento**
- **Categoria de Seletores:**
  - **TAG:** elementos com base em seu tipo HTML, por exemplo, **<p>**
  - **ID:** elemento único dentro de uma página e pode ser atribuido uma formatação especial apenas a esse id
  - **CLASS:** elemento ou grupo de elementos que participam da mesma classe podem receber uma formatação específica
  - **UNIVERSAL:** Seleciona todos os elementos de uma página através ***

#UTILIZANDO SELETORES - POR TAG

- Para fazermos referência a uma **TAG** a é apenas mencionada o nome da TAG
- No exemplo ao lado, todos **p** em uma página receberão tais propriedades

```css
p {
  font-size: 16px;
  color: #333;
}
```

#UTILIZANDO SELETORES - POR CLASSE

- Para fazermos refereência a uma **CLASS** é apenas mencionada nome da classe com um ponto (.) antes

- No exemplo ao lado, todos os elementos que receberem a class **souClass** vão receber tais propriedades

#UTILIZANDO SELETORES - POR IDENTIFICADOR

- Para fazermos referência a um **ID** é apenas mencionada nome do ID com um sustenido (#) antes

- No exemplo ao lado, o elemento que receber o id **cabecalho** vai receber tais propriedades

```css
#cabecalho {
  font-size: 24px;
  color: #007bff;
}
```

#UTILIZANDO SELETORES - GLOBAL

- Para fazermos referência **UNIVERSAL** é apenas mencionada um asteristico (*)

- No exemplo ao lado, todos os elemento vão receber tais propriedades

```
* {
  margin: 0;
  padding: 0;
}
```

#UTILIZANDO SELETORES

- Dependendo da estrutura da nossa aplicação, selecionar elementos acaba se tornando um trabalho complexo e, depois de algumas atualização do CSS foi-se criado um dinamismo para fazer a seleção:

- No exemplo 1, Seleciona todos os elementos **<p>** com a **class** text_color

- No exemplo 2, Seleciona todos os elementos **<p>** que estão dentro de uma ** **

EXEMPLO 1

```
p.text_color {
    color: #fff;
}
```

EXEMPLO 2

```
div > p {
   color: #0800
}
```

PREENCHIMENTO

CSS

#UTILIZANDO PRENCHIMENTOS

- As propriedades de preenchimento (padding properties) são usadas para definir o espaço entre o conteúdo de um elemento HTML e suas bordas

- Permitem ajustar o espaço interno de um elemento, criando margens entre o conteúdo e as bordas

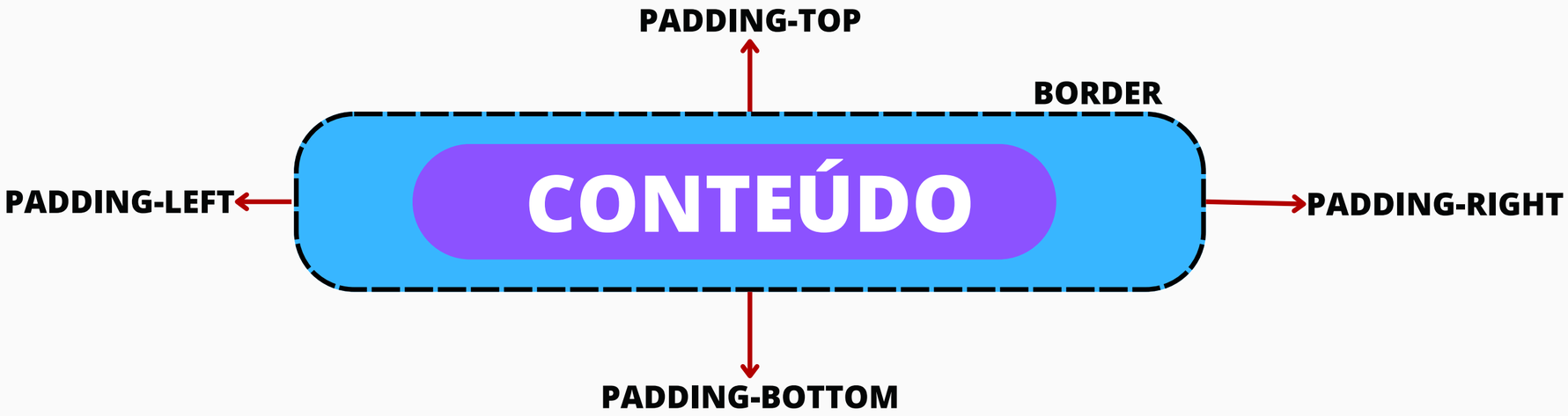
- As principais propriedades são:
  - **PADDING**: Define o preenchimento de todos os lados do elemento (top, right, bottom e left) em uma única linha de código
  - **MARGIN**: Define o tamanho do espaço fora da borda. É utilizado para gerar um espaço em torno do elemento

#UTILIZANDO PRENCHIMENTOS

- Em Resumo o Padding é o espaço entre a borda e o margin é o espaço fora da borda, veja uma ilustração abaixo:

- No **Margin** também tem o conjunto de propriedades manuais para determinar a **espaço** de um elemento na página, através:

    - **Margin-Top:** Define o tamanho do espaço fora da borda na vertical (Y), no lado superior do elemento
    - **Margin-Right**: Define o tamanho do espaço fora da borda na horizontal(X), no lado direito do elemento
    - **Margin-Bottom**: Define o tamanho do espaço fora da borda na vertical (Y), no lado inferior do elemento;
    - **Margin-Left**: Define o tamanho do espaço fora da borda horizontal (X), no lado esquerdo do elemento



- No **Padding** também tem o conjunto de propriedades manuais para determinar a **preenchimento** de um elemento na página, através:

    - **Padding-Top:** Define o tamanho do preenchimento antes da borda na vertical (Y), no lado superior do elemento
    - **Padding-Right**: Define o tamanho do preenchimento antes da borda na horizontal(X), no lado direito do elemento
    - **Padding-Bottom**: Define o tamanho do preenchimento antes da borda na vertical (Y), no lado inferior do elemento;
    - **Padding-Left**: Define o tamanho do espaço fora da borda horizontal (X), no lado esquerdo do elemento

DISPLAY

CSS

#UTILIZANDO DISPLAYS

- Basicamente todos os elementos do HTML há uma propriedade para display no qual determina como o tal elemento será renderizado no layout da página

- Alguns dos valores comuns para a propriedade "display" incluem:
  - **block**: Isso faz com que um elemento seja renderizado como um bloco retangular, ocupando toda a largura disponível. Elementos "block" geralmente começam em uma nova linha.
  - Exemplos típicos incluem ** , <p>, e <h1>.**

```css
.block-element {
  display: block;
  width: 300px;
  height: 100px;
  background-color: lightblue;
}
```

#UTILIZANDO DISPLAYS

- **inline**: Isso faz com que um elemento seja renderizado em linha, ocupando apenas o espaço necessário. Elementos "inline" não começam em uma nova linha.

- Exemplos incluem , <a>, e .

```css
.inline-element {
  display: inline;
  background-color: lightgreen;
  padding: 5px;
}
```

#UTILIZANDO DISPLAYS

- **inline-block**: Combinando características de elementos "block" e "inline", os elementos "inline-block" são renderizados como um bloco retangular, mas ficam na mesma linha, se houver espaço suficiente.

```css
.inline-block-element {
  display: inline-block;
  width: 100px;
  height: 100px;
  background-color: lightcoral;
  margin-right: 10px;
}
```

#UTILIZANDO DISPLAYS

- **grid**: A propriedade display: grid cria um container grid que permite o posicionamento preciso de elementos em linhas e colunas. É útil para layouts complexos

```css
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 10px;
}
.grid-item {
  background-color: lightblue;
  padding: 10px;
}
```

#UTILIZANDO DISPLAYS

- **flex e grid**: Esses valores permitem a criação de layouts flexíveis e baseados em grade, permitindo o controle preciso dos elementos filhos dentro de um contêiner.

```css
.flex-container {
  display: flex;
}
.flex-item {
  flex: 1;
  padding: 10px;
  background-color: lightyellow;
}
```

#UTILIZANDO DISPLAYS

- **none**: Isso faz com que um elemento fique invisível e não ocupe nenhum espaço no layout.

```css
.hidden-element {
  display: none;
}
```

POSITION

CSS

#UTILIZANDO POSITIONS

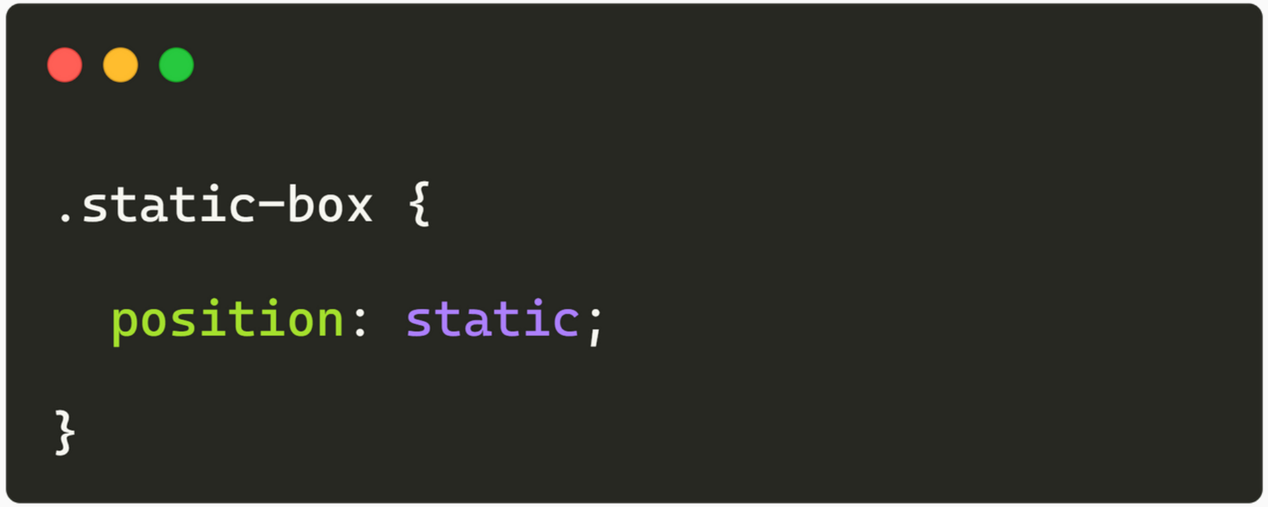
- É uma das propriedades fundamentais do CSS **que controla o posicionamento de elementos** HTML em uma página da web

- É frequentemente usada em conjunto com outras propriedades, como **top, right, bottom e left**, para determinar a posição exata de um elemento na página

    - **Top:** Desloca o elemento na vertical (Y), o valor é a distância do elemento com o topo
    - **Right**: Desloca o elemento na horizontal(X), o valor é a distância do elemento com a borda direita
    - **Bottom**: Desloca o elemento na vertical (Y), o valor é a distância do elemento com a borda inferior
    - **Left**: Deslocal o elemento na horizontal (X), o valor é a distância do elemento com a borda esquerda

#UTILIZANDO POSITIONS

- A propriedade **position** possui 4 valores possíveis:
  - **static (padrão)**:
    - Este é o valor padrão e o comportamento padrão de todos os elementos HTML.
    - Elementos com position: static são posicionados de acordo com o fluxo normal do documento.
    - As propriedades top, right, bottom e left não têm efeito em elementos com position: static.



#UTILIZANDO POSITIONS

- A propriedade **position** possui 4 valores possíveis:
  - **relative**
    - Elementos com position: relative são posicionados em relação à sua posição normal no fluxo do documento.
    - Pode usar as propriedades top, right, bottom e left para ajustar a posição do elemento em relação à sua posição origin**al.**

```css
.relative-box {
  position: relative;
  top: 20px;
  left: 30px;
}
```

#UTILIZANDO POSITIONS

- A propriedade **position** possui 4 valores possíveis:
  - **absolute**
    - Com position: absolute, o elemento é posicionado em relação ao elemento pai que tenha position diferente de static.
    - O elemento não ocupa espaço em seu local original no layout.

```css
.parent {
  position: relative;
}


.absolute-box {
  position: absolute;
  top: 10px;
  left: 10px;
}
```

#UTILIZANDO POSITIONS

- A propriedade **position** possui 4 valores possíveis:
  - **fixed**:
    - Elementos com position: fixed são posicionados em relação à janela do navegador, independentemente da posição de seus elementos pais.
    - Esses elementos permanecem fixos na tela, mesmo quando a página é rolada.

```css
.fixed-box {
  position: fixed;
  top: 20px;
  right: 20px;
}
```

FLEXBOX

CSS

#UTILIZANDO FLEXBOX

- É um dos principais conceitos CSS que permite **facilitar o posicionamento e o alinhamento de elementos, especialmente quando se trata de construir layouts responsivos e dinâmicos** em uma página da web

- É possível criar layouts complexos sem depender de estruturas de layout tradicionais, como floats e posicionamento absoluto. Em vez disso, você define a relação entre os elementos em relação ao contêiner pai e permite que o Flexbox cuide do dimensionamento e do posicionamento.

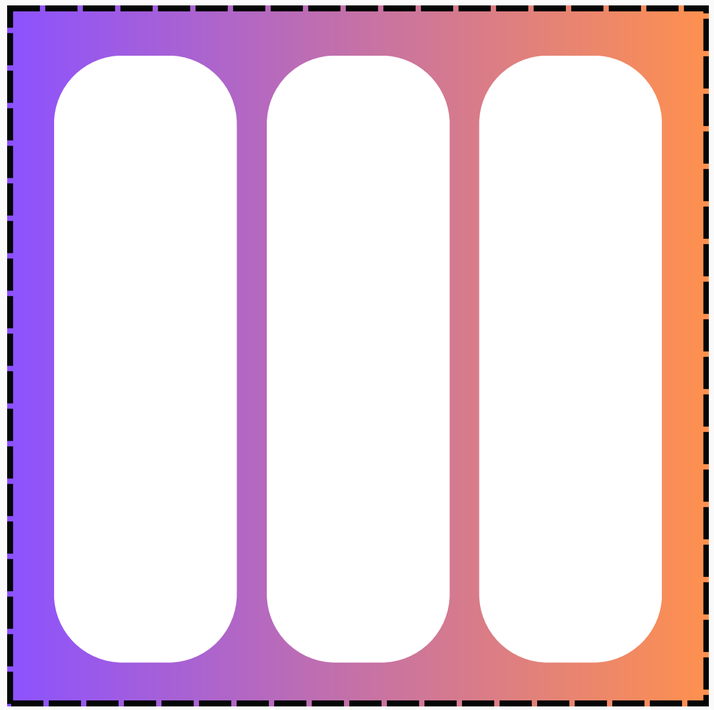
- Principais conceitos incluem:
  - **Contêiner Flex (flex container):** É o elemento que envolve os itens flexíveis e define o contexto do layout flexível. Para tornar um elemento um contêiner flex, você aplica a propriedade display: flex; ou display: inline-flex; ao elemento pai.
  - **Itens Flexíveis (flex items):** os elementos filhos diretos do contêiner flex. Esses elementos são organizados e alinhados dentro do contêiner flex de acordo com as regras do Flexbox

#UTILIZANDO FLEXBOX

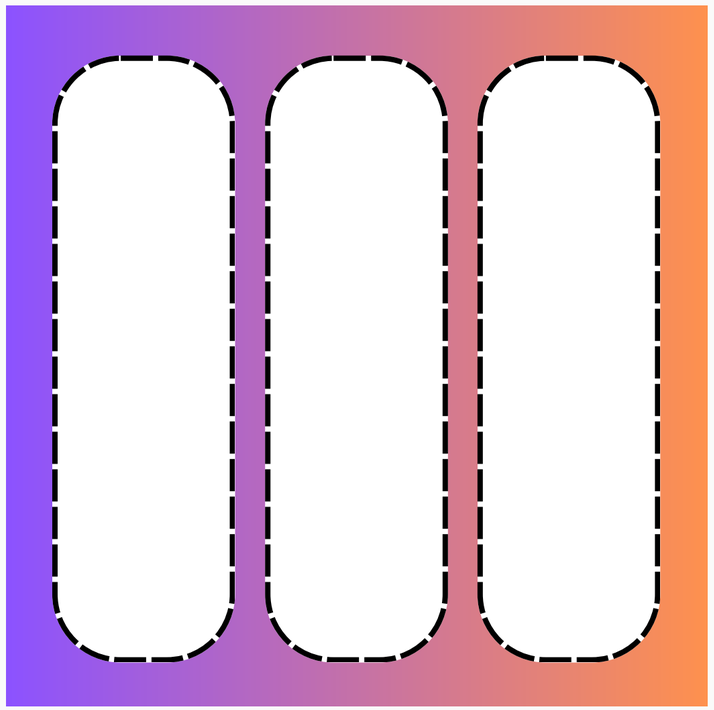
PERMITE AOS FILHOS:

- Direção
- Justificação
- Alinhamento

CONTAINER PAI



ELEMENTOS FILHOS

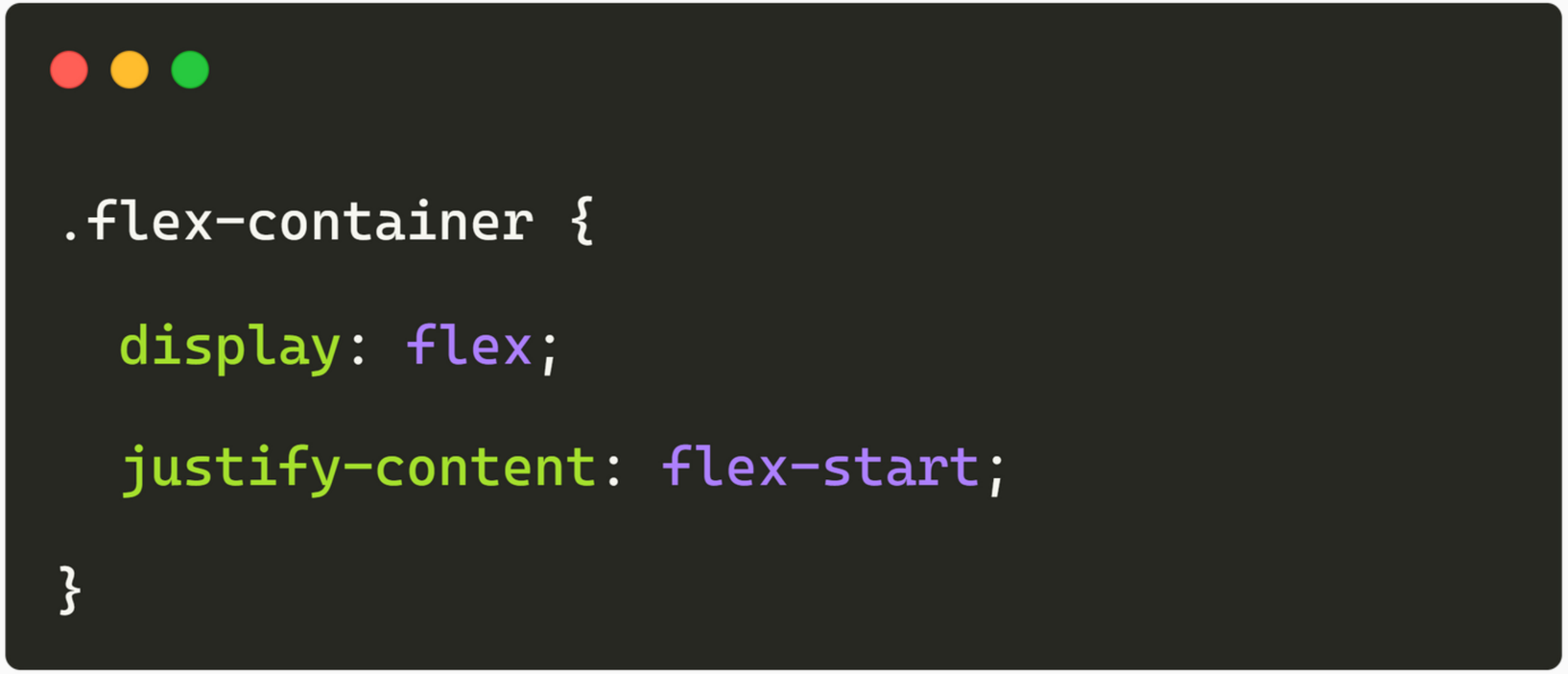


PODEM INDIVDUALMENTE:

- Ordenados
- Alinhados

#UTILIZANDO FLEXBOX - JUSTIFY CONTENT

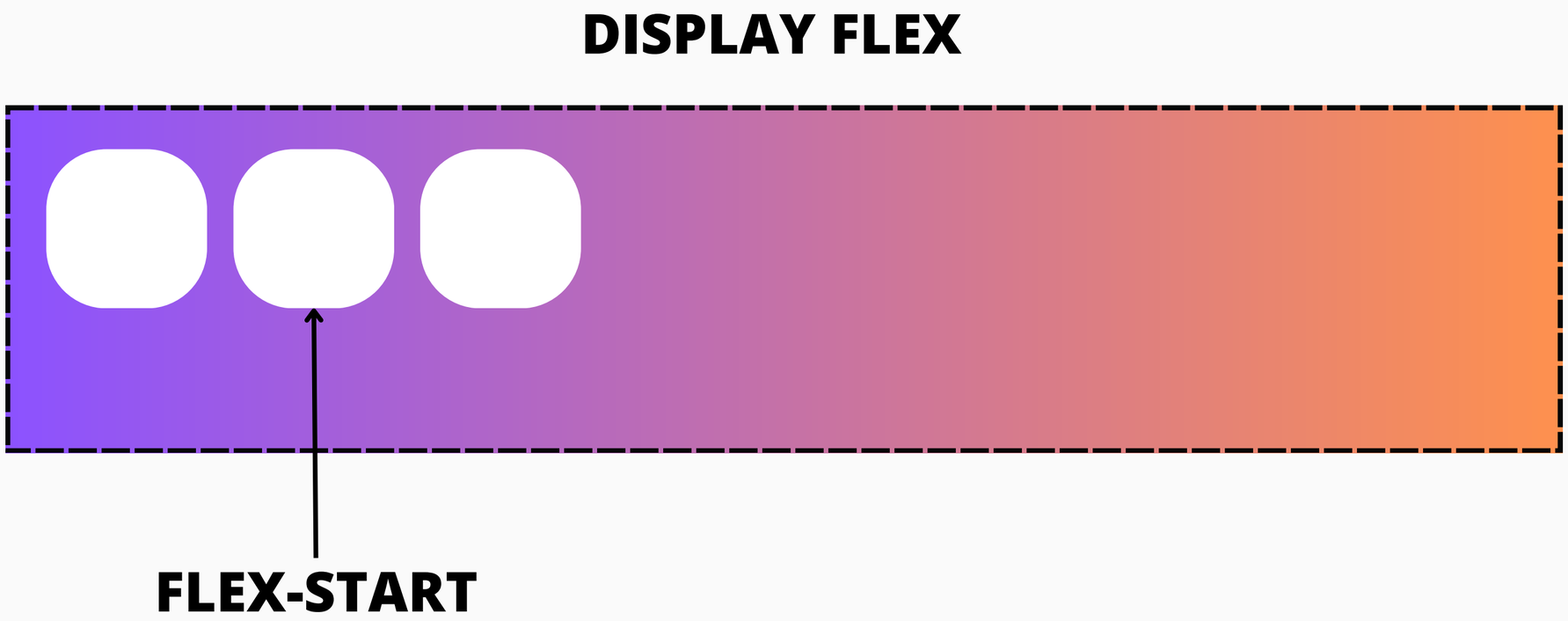
- A propriedade **justify-content** define como o navegador distribui o espaço entre e ao redor dos itens ao longo do eixo principal de um container flexível, **no caso o eixo horizontal**. Atenção! A propriedade terá efeito caso a propriedade display flex for mencionada.



```
.flex-container {
  display: flex;
  justify-content: flex-start;
}
```

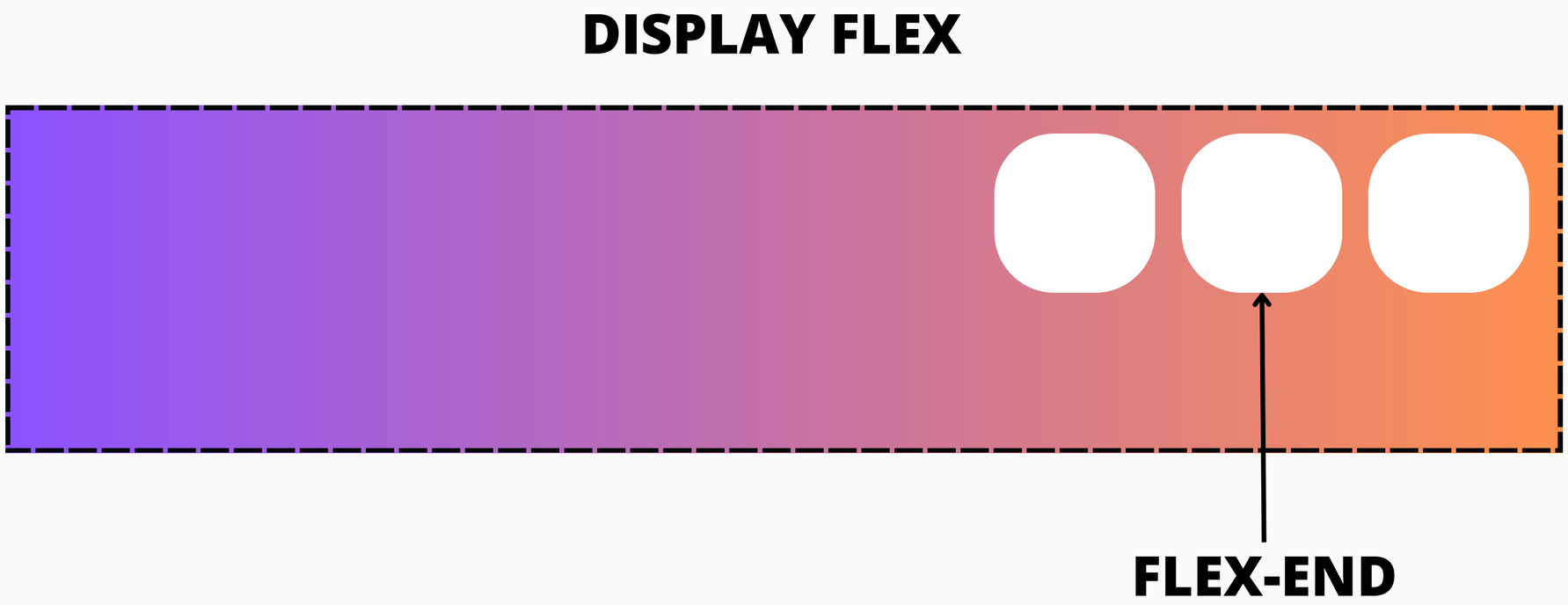
#UTILIZANDO FLEXBOX - JUSTIFY CONTENT

- A propriedade **flex-start** define que os itens serão alinhados no começo do eixo principal. No CSS essa propriedade é considerada *default* (ou padrão). Vejao o Exemplo abaixo



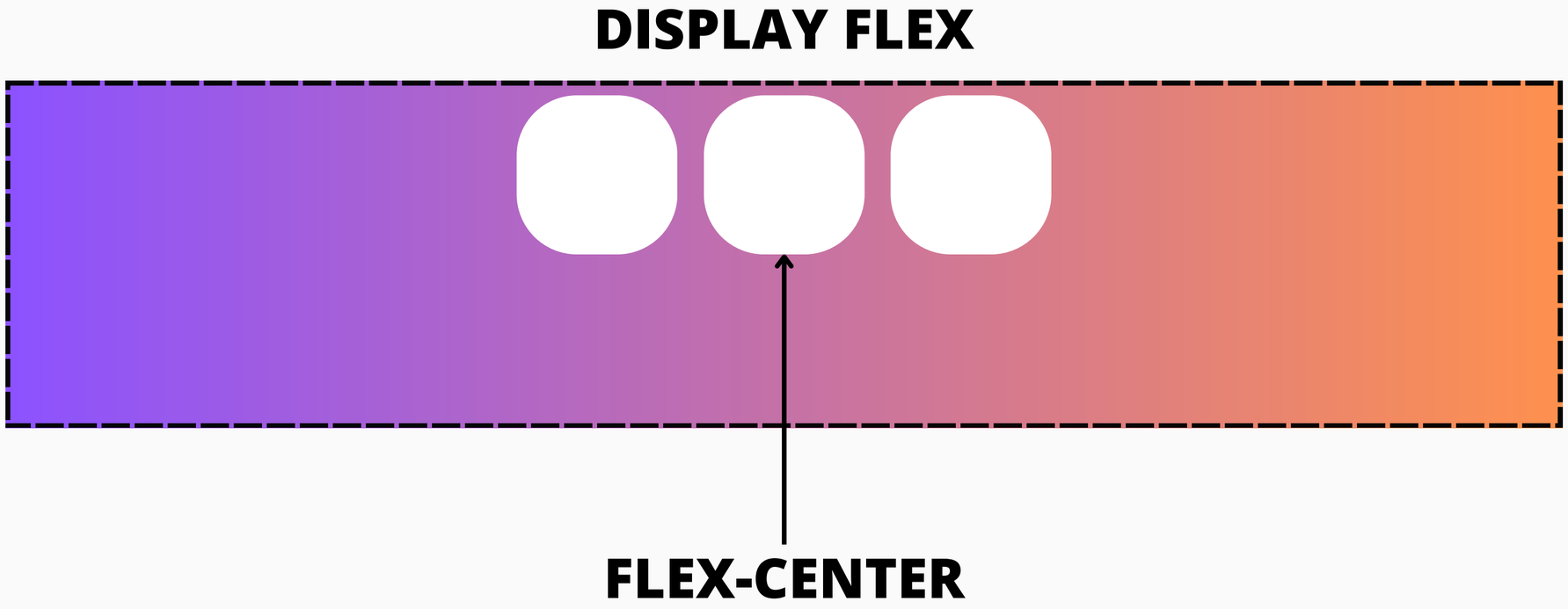
#UTILIZANDO FLEXBOX - JUSTIFY CONTENT

- A propriedade **flex-end** define que os itens serão alinhados no final do eixo principal. Esse é o inverso do flex-start



#UTILIZANDO FLEXBOX - JUSTIFY CONTENT

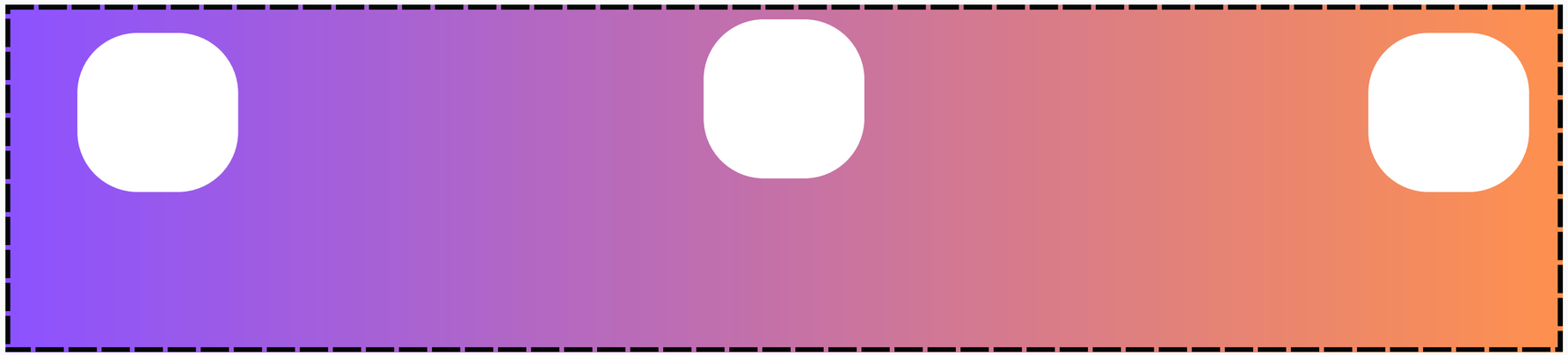
- A propriedade **center** define que os itens serão alinhados no meio do eixo principal.



#UTILIZANDO FLEXBOX - JUSTIFY CONTENT

- A propriedade **space-between** define que o serão alinhados dessa seguinte maneira:
  - O Primeiro item no inicio do eixo principal
  - OUltimo item no fim do eixo principal.
  - Os demais itens alinhas entre si e no meio do eixo principal;

DISPLAY FLEX

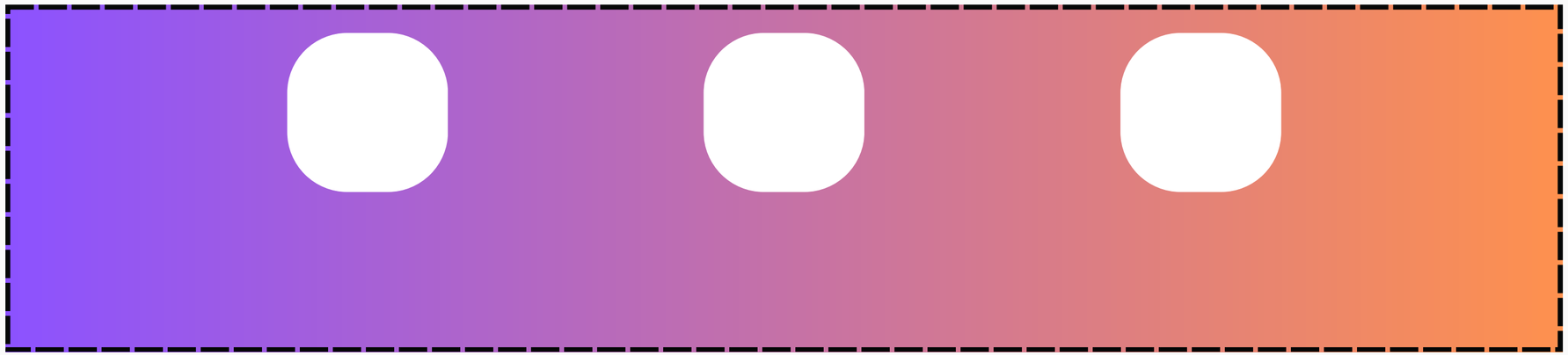


SPACE-BETWEEN

#UTILIZANDO FLEXBOX - JUSTIFY CONTENT

- A propriedade **space-around** define que o serão alinhados e distribuidos de forma igual ao seu redos, isto é, haverá um espaço (ou área) padrão entre eles

DISPLAY FLEX



SPACE-BETWEEN

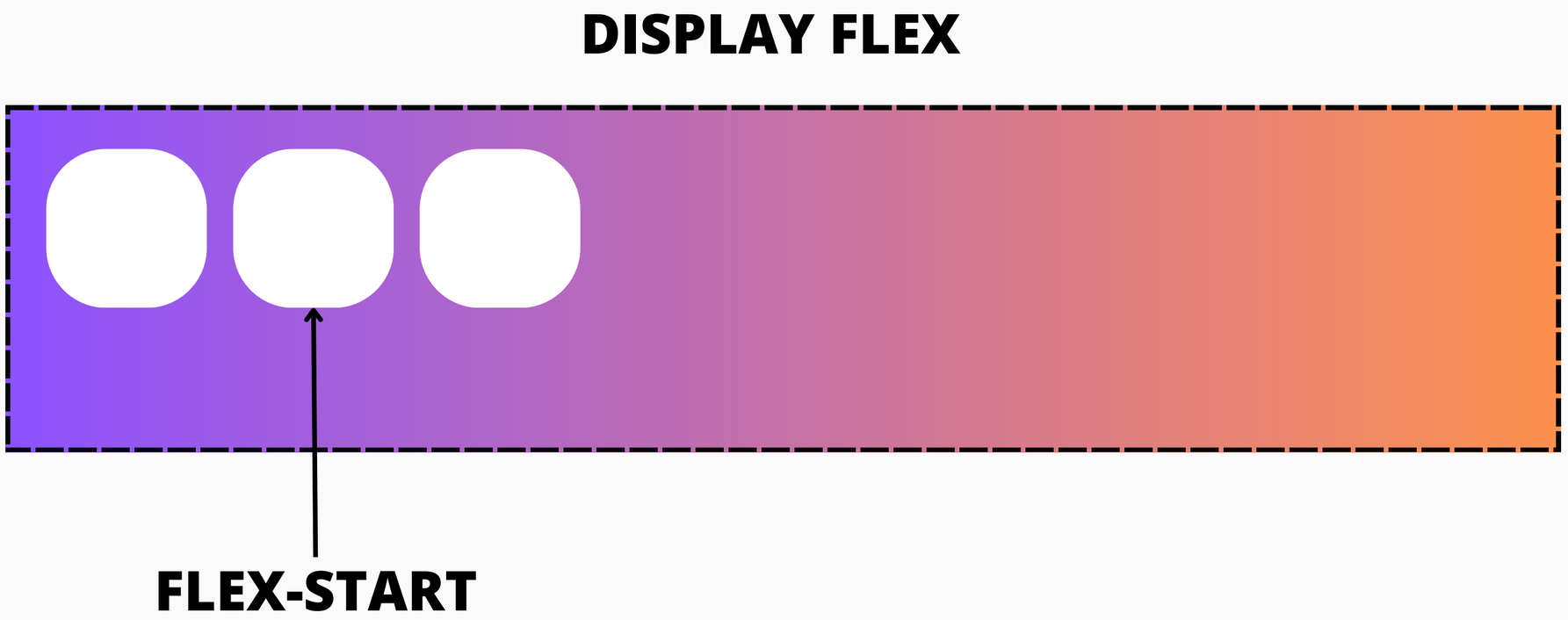
#UTILIZANDO FLEXBOX - ALIGN ITENS

- A propriedade **align-item** define como os itens no container serão alinhados, **no caso no eixo vertical**. Atenção! A propriedade terá efeito caso a propriedade display flex for mencionada.

```css
.flex-container {
  display: flex;
  align-items: center;
}
```

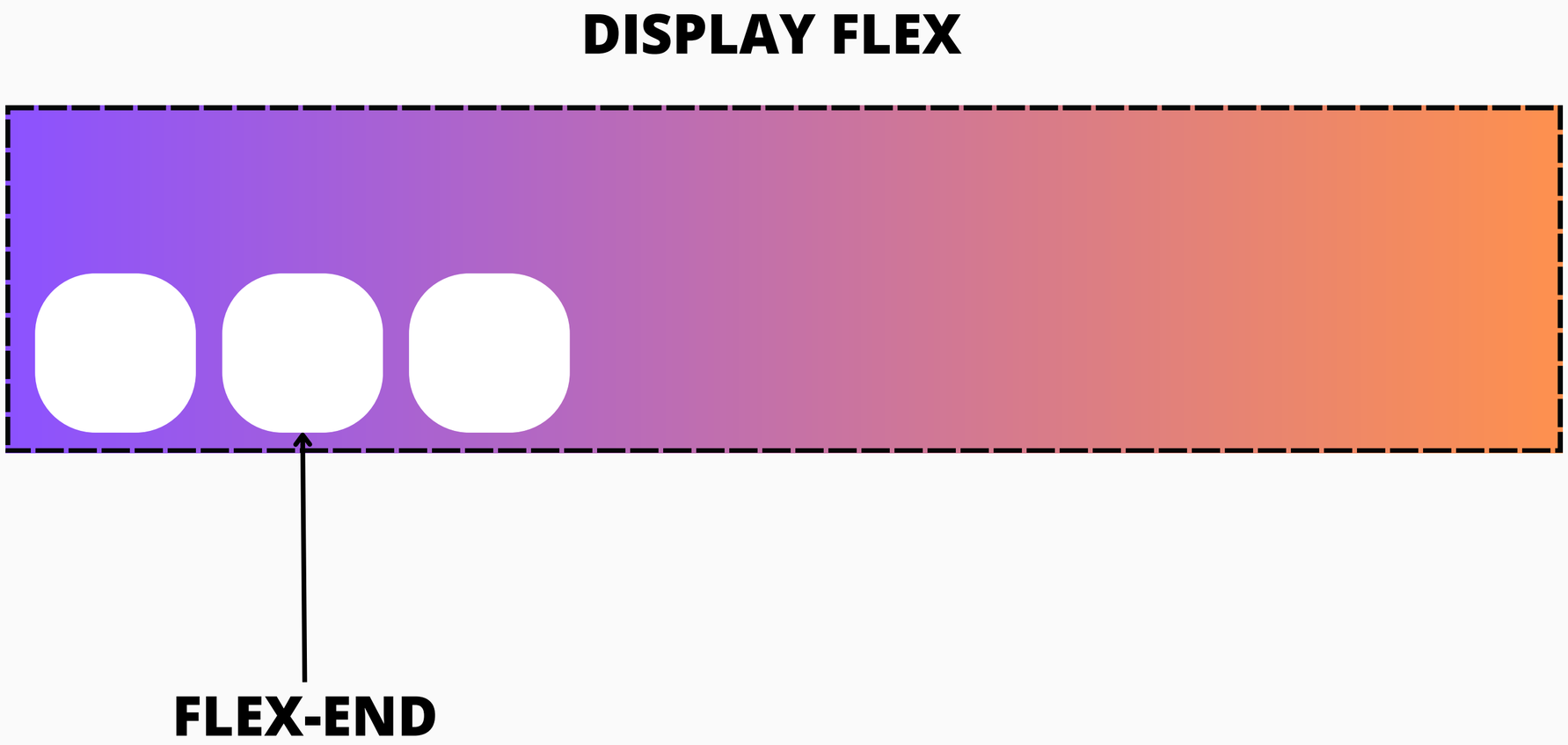
#UTILIZANDO FLEXBOX - ALIGN ITENS

- A propriedade **flex-start** define que os itens serão alinhados **no começo** do eixo transversal. No CSS essa propriedade é considerada *default* (ou padrão). Vejao o Exemplo abaixo



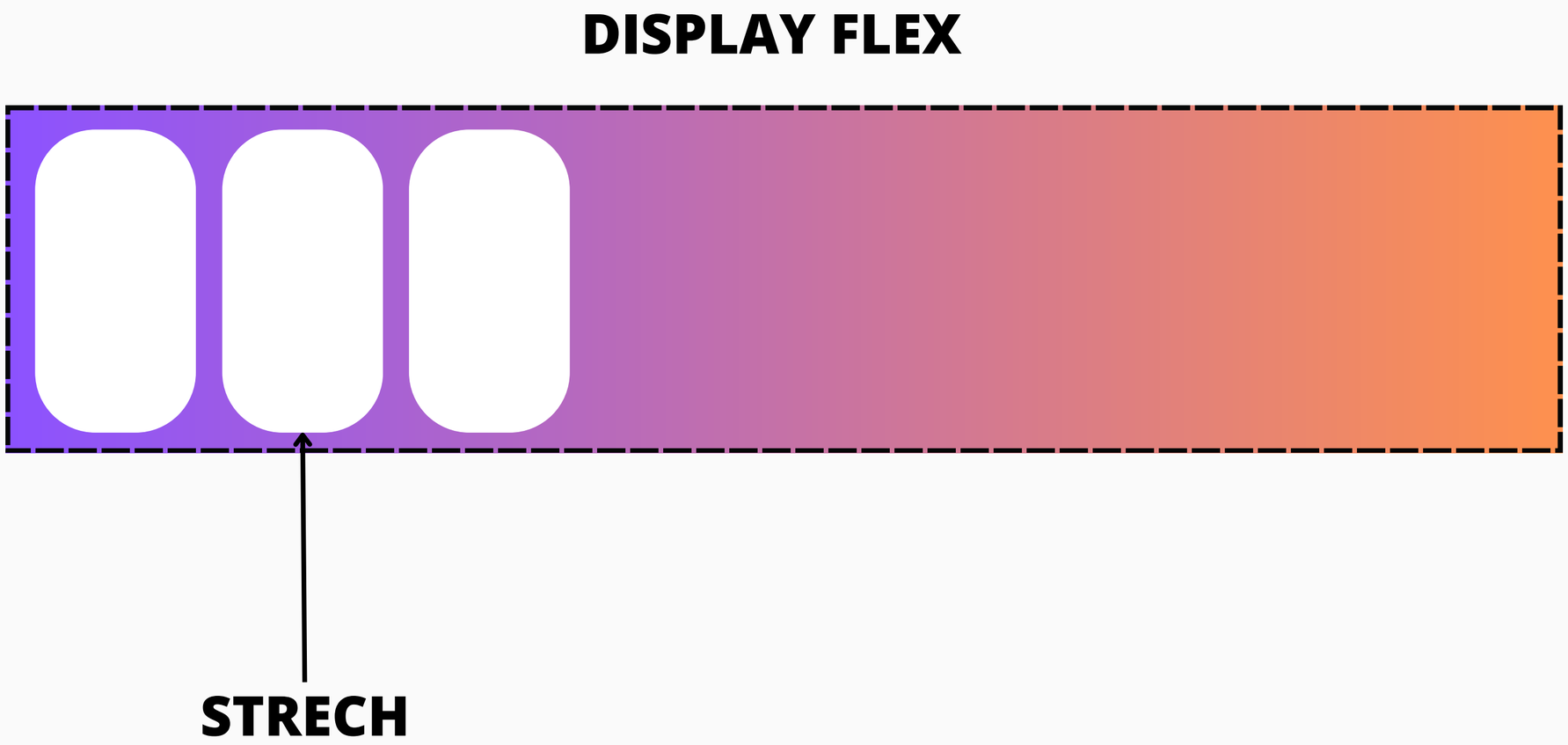
#UTILIZANDO FLEXBOX - ALIGN ITENS

- A propriedade **flex-end** define que os itens serão alinhados **no final** do eixo transversal. Esse é o inverso do flex-start



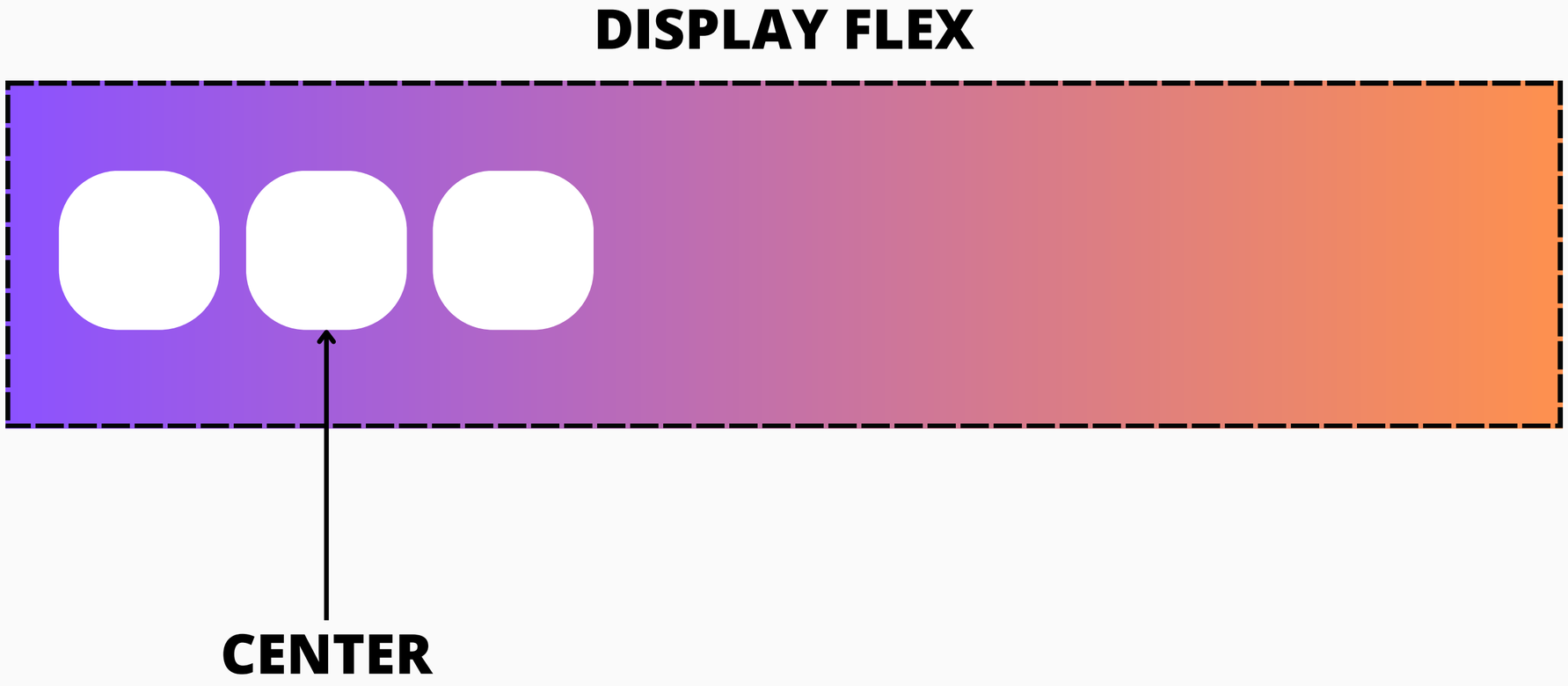
#UTILIZANDO FLEXBOX - ALIGN ITENS

- A propriedade **strech** define que os itens serão extendidos até **o final** do eixo transversal.



#UTILIZANDO FLEXBOX

- A propriedade **center** define que os itens serão alinhados no meio do eixo transversal.



Alguma Dúvida?