📢 GUIA COMPLETO DE FLEXBOX - CSS3

O Flexbox (Flexible Box Layout) é um modelo de layout que permite organizar, alinhar e distribuir elementos dentro de um container de forma eficiente. Ele resolve os problemas clássicos de alinhamento no CSS, como usar `float`, `inline-block` e `position` para centralizar elementos.

Com Flexbox, podemos:
✅ Criar layouts dinâmicos e responsivos facilmente
✅ Alinhar itens horizontalmente e verticalmente sem dor de cabeça
✅ Distribuir espaço entre os elementos de forma automática

1️⃣ Como ativar o Flexbox?

Apenas adicionar `display: flex;` no elemento pai (container):

css

CopiarEditar

```
.container {
    display: flex;
}
```

Isso transforma os elementos filhos em itens flexíveis.

2️⃣ Conceitos Essenciais

Antes de tudo, entenda que o Flexbox trabalha com dois eixos principais:

📌 Eixo Principal (Main Axis) → Controlado por `flex-direction`
📌 Eixo Secundário (Cross Axis) → Controlado por `align-items`

📍 Por padrão, o eixo principal é horizontal (da esquerda para a direita), e o eixo secundário é vertical (de cima para baixo).

- **Container (o PAI) → Controla o layout geral**
- **Items (os FILHOS) → São os elementos dentro do container**

Direções e Eixos no Flexbox

Quando usamos `display: flex;`, estamos basicamente ativando um "sistema de coordenadas" para os elementos filhos dentro do contêiner. Esse sistema tem dois eixos principais:

1. **Eixo Principal (Main Axis)** – É o eixo principal de organização dos itens. Depende da direção definida por `flex-direction`.
2. **Eixo Secundário (Cross Axis)** – É o eixo perpendicular ao principal.

Agora vamos entender o `flex-direction`, que define para onde os elementos filhos vão se alinhar dentro do contêiner.

---

flex-direction: Definindo a Direção dos Itens

O `flex-direction` pode ter quatro valores, e cada um altera a direção dos itens no eixo principal:

Valor	Eixo Principal	Sentido
`row`	Horizontal	Esquerda → Direita (padrão)
`row-reverse`	Horizontal	Direita → Esquerda
`column`	Vertical	Topo → Baixo
`column-reverse`	Vertical	Baixo → Topo

Exemplo:

css

CopiarEditar

```
.container {
    display: flex;
    flex-direction: row; /* Itens ficam lado a lado na horizontal */
}
```

Resumindo os Eixos

- **Eixo Principal:** Onde os itens são organizados (`flex-direction`).
- **Eixo Secundário:** Sempre perpendicular ao eixo principal.

Exemplo prático: Se `flex-direction: row;`, então:

- **Eixo Principal** → **Horizontal** (da esquerda para a direita).
- **Eixo Secundário** → **Vertical** (de cima para baixo).

Se `flex-direction: column;`, então:

- **Eixo Principal** → **Vertical** (de cima para baixo).
- **Eixo Secundário** → **Horizontal** (da esquerda para a direita).

3️⃣ Propriedades do Container (PAI)

Propriedade	O que faz
`display: flex;`	**Ativa o Flexbox**
`flex-direction`	**Define a direção dos itens (linha, coluna...)**
`justify-content`	**Alinha os itens no eixo principal**

`align-items`	Alinha os itens no eixo secundário
`align-content`	Alinha múltiplas linhas
`flex-wrap`	Define se os itens podem quebrar linha
`gap`	Adiciona espaçamento entre os itens

🎯 **Exemplo de Container**

css

CopiarEditar

```
.container {
    display: flex;
    flex-direction: row; /* Itens em linha */
    justify-content: center; /* Centraliza no eixo principal */
    align-items: center; /* Centraliza no eixo secundário */
    gap: 10px; /* Espaçamento entre os itens */
}
```

4️⃣ `flex-direction`: Direção dos itens no eixo principal

Define se os itens ficarão em linha ou coluna.

css

CopiarEditar

```
.container {
    display: flex;
    flex-direction: row; /* Padrão: da esquerda para a direita */
}
```

📌 **Outras opções:**

- `row-reverse` → **Inverte a ordem na horizontal**
- `column` → **Organiza os itens na vertical (de cima para baixo)**
- `column-reverse` → **Inverte a ordem na vertical**

5️⃣ `justify-content`: Alinhamento no eixo principal

Controla como os elementos são distribuídos horizontalmente (por padrão).

css

CopiarEditar

```
.container {
    display: flex;
    justify-content: center;
```

```
}
```

📌 **Opções:**

- `flex-start` → Alinha no início
- `flex-end` → Alinha no final
- `center` → Centraliza
- `space-between` → Espaço máximo entre os itens
- `space-around` → Espaço ao redor dos itens
- `space-evenly` → Espaçamento igual entre tudo

6️⃣ `align-items`: Alinhamento no eixo secundário

Controla como os itens se alinham verticalmente.

css

CopiarEditar

```
.container {
    display: flex;
    height: 300px;
    align-items: center;
}
```

📌 **Opções:**

- `stretch` → Estica os itens (padrão)
- `flex-start` → Alinha no topo
- `flex-end` → Alinha na base
- `center` → Centraliza

7️⃣ `flex-wrap`: Itens podem quebrar linha?

Se os itens devem ou não quebrar para a linha de baixo quando não houver espaço suficiente.

css

CopiarEditar

```
.container {
    display: flex;
    flex-wrap: wrap;
}
```

📌 **Opções:**

- `nowrap` → Itens ficam em uma única linha (padrão)
- `wrap` → Itens quebram linha quando necessário
- `wrap-reverse` → Quebra a linha, mas invertendo a ordem

8️⃣ Propriedades dos Itens (FILHOS)

Propriedade	O que faz
`flex-grow`	Faz o item crescer se tiver espaço

`flex-shrink`	Faz o item diminuir quando faltar espaço
`flex-basis`	Define o tamanho inicial do item
`align-self`	Alinha um item individualmente

🎯 `flex-grow`: Crescimento dos itens

Faz um item crescer para ocupar o espaço disponível.

css

CopiarEditar

```
.item {
    flex-grow: 1;
}
```

Se tivermos três itens com `flex-grow: 1`, todos crescem igualmente.

Se um item tiver `flex-grow: 2`, ele crescerá o dobro dos outros.

9️⃣ `align-self`: Alinhar um único item

Dá um alinhamento individual para um item dentro do Flexbox.

css

CopiarEditar

```
.item:nth-child(2) {

    align-self: flex-end;

}
```

📌 **Opções:**

- `auto` → Usa o `align-items` do pai
- `flex-start` → Alinha no topo
- `flex-end` → Alinha na base
- `center` → Centraliza
- `stretch` → Estica para ocupar espaço disponível

🔥 Exemplo Prático - Código Completo

html

CopiarEditar

```
<!DOCTYPE html>

<html lang="pt">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Flexbox</title>

    <style>

        .container {
```

```
            display: flex;
            flex-direction: row;
            justify-content: space-between;
            align-items: center;
            background-color: #eee;
            padding: 20px;
            gap: 10px;
        }

        .item {
            background: tomato;
            color: white;
            padding: 20px;
            text-align: center;
            width: 100px;
            font-size: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
```

```
        <div class="item">1</div>
        <div class="item">2</div>
        <div class="item">3</div>
    </div>
</body>
</html>
```

✨ **Resultado: Itens alinhados lado a lado, espaçados e centralizados.**

🎯 Conclusão

Agora você domina o Flexbox! 🚀
Ele é essencial para layouts modernos e responsivos.

📌 **Dicas finais:**
✅ **Pratique com `display: flex;` em seus projetos**
✅ **Experimente `flex-grow`, `flex-shrink` e `align-self`**
✅ **Use `flex-wrap` para layouts responsivos**