Professores:

Celso Giusti

Daniel Manoel Filho

Marlon Palata Fanger Rodrigues

Flexbox e CSS Grid

SENAI

Por Que Flexbox e Grid?

Na última aula, vimos como criar layouts usando **float** (antigo) e **position** (rígido). Embora funcionem, eles exigem "truques" (hacks) para coisas simples, como centralizar verticalmente ou criar colunas de alturas iguais.

Para resolver isso, o CSS moderno nos deu duas ferramentas poderosas:

1. **Flexbox (Flexible Box Layout):**
   - Modelo **unidimensional** (1D): Controla o layout em uma linha **OU** uma coluna por vez.
   - **Perfeito para:** Componentes, menus de navegação, alinhar itens dentro de um card.
2. **CSS Grid (Grid Layout):**
   - Modelo **bidimensional** (2D): Controla linhas **E** colunas simultaneamente.
   - **Perfeito para:** O layout principal da página (Cabeçalho + Menu + Conteúdo + Rodapé).

Setup para a Aula Prática

Vamos criar um arquivo HTML específico para testar essas propriedades.

Estrutura:

```
/aula16.1-flex-grid/
|-- index.html
|-- style.css
```

Acesse as bases html e css:

https://github.com/marlon-palata/meu_site/tree/main/aula16.1-flex-grid/exemplos

Professores:
Celso Giusti
Daniel Manoel Filho
Marlon Palata Fanger Rodrigues

FLEXBOX

SENAI

O Conceito Principal

Para ativar o Flexbox, aplicamos **display: flex;** no **elemento pai** (o Container). Automaticamente, os **filhos diretos** (Itens) se tornam flexíveis e tentam ficar lado a lado.

O Flexbox trabalha com dois eixos invisíveis:

- **Eixo Principal (Main Axis):** A direção do fluxo (linha horizontal por padrão).
- **Eixo Cruzado (Cross Axis):** A direção perpendicular (coluna vertical por padrão).

CSS (style.css):

```css
/* 1.1. Básico */
#flex-basico {
  display: flex; /* Ativa o Flexbox */
}
```

Propriedades do PAI: flex-direction

Define a direção do **Eixo Principal**.

- **row:** (Padrão) Esquerda para a direita.
- **row-reverse:** Direita para a esquerda.
- **column:** Cima para baixo (empilha os itens).
- **column-reverse:** Baixo para cima.

CSS (style.css):

```
/* 1.2. Direção (Column) */
#flex-direction {
  display: flex;
  flex-direction: column; /* Muda eixo principal para vertical */

  /* CORREÇÃO: Remove a altura fixa de 150px para que o container
     cresça e abrace os itens empilhados, sem cortar a borda. */
  height: auto;
}
```

Propriedades do PAI: justify-content

Alinha os itens ao longo do **Eixo Principal** (horizontal, se for row).

- **flex-start:** (Padrão) Início.
- **flex-end:** Fim.
- **center:** Centro.
- **space-between:** Primeiro no início, último no fim, espaço igual no meio.
- **space-around:** Espaço igual ao redor de cada item.
- **space-evenly:** Espaço visualmente idêntico entre todos os itens.

CSS (style.css):

```css
/* 1.3. Justify Content (Eixo Principal - Horizontal aqui) */
#flex-justify {
  display: flex;
  justify-content: space-between; /* Espalha os itens */
}
```

Propriedades do PAI: align-items

Alinha os itens ao longo do **Eixo Cruzado** (vertical, se for row).

- **stretch:** (Padrão) Estica os itens para preencher a altura do container.
- **flex-start:** Topo.
- **flex-end:** Base.
- **center:** Meio.

CSS (style.css):

```css
/* 1.4. Align Items (Eixo Cruzado - Vertical aqui) */
#flex-align {
  display: flex;
  align-items: center; /* Centraliza na altura do container */
}
```

Propriedades do PAI: align-items

Alinha os itens ao longo do **Eixo Cruzado** (vertical, se for row).

- **stretch:** (Padrão) Estica os itens para preencher a altura do container.
- **flex-start:** Topo.
- **flex-end:** Base.
- **center:** Meio.

CSS (style.css):

```css
/* 1.5. Centralização Total (O Santo Graal) */
#flex-align {
  display: flex;
  justify-content: center; /* Centro Horizontal */
  align-items: center; /* Centro Vertical */
}
```

Propriedades do PAI: flex-wrap

Define se os itens devem "quebrar" para a linha de baixo se não houver espaço.

- **nowrap:** (Padrão) Força tudo na mesma linha (os itens encolhem).
- **wrap:** Quebra para a próxima linha.

CSS (style.css):

```
/* 1.6. Wrap (Quebra de Linha) */
#flex-wrap {
  display: flex;
  flex-wrap: wrap; /* Permite quebrar linha */
  height: auto; /* Altura automática para caber as linhas */
}

/* Ajuste nos itens do wrap para caberem e testar a quebra */
#flex-wrap .item {
  width: 150px;
  margin: 5px;
}
```

Propriedades do FILHO: flex-grow

Define se um item específico pode **crescer** para ocupar o espaço vazio que sobrou.

- **0:** (Padrão) Não cresce.
- **1:** Cresce para ocupar o espaço disponível.

CSS (style.css):

```
/* 1.7. Grow (Crescimento) */
#flex-grow {
  display: flex;
}
#flex-grow .item {
  flex-grow: 1; /* Todos crescem igualmente para preencher o espaço */
}
```

Professores:
Celso Giusti
Daniel Manoel Filho
Marlon Palata Fanger Rodrigues

GRID

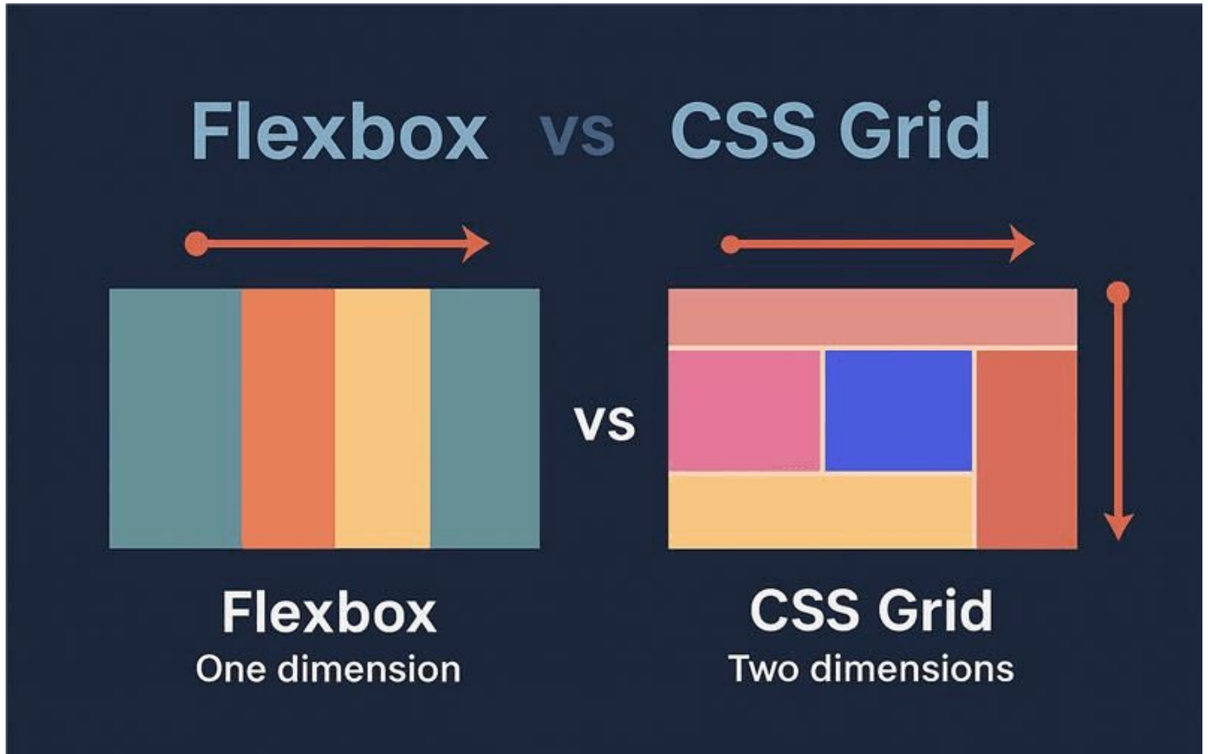
SENAI

Transição para o Grid

O Flexbox é ótimo para uma **linha de ícones** ou **menu.**

Mas para o layout geral da página (Cabeçalho + Lateral + Conteúdo), o **CSS Grid** é superior.

Ele **divide o pai** em uma **grade de linhas e colunas.**



Definindo Colunas e Linhas

Ativamos com **display: grid;** no pai. Depois, definimos o tamanho das trilhas (tracks).

A unidade **fr** (fraction) é exclusiva do Grid e significa "uma fração do espaço disponível".

CSS (style.css):

```css
#grid-basico {
    display: grid;
    /* Cria 3 colunas de tamanhos iguais (1 fração cada) */
    grid-template-columns: 1fr 1fr 1fr;

    /* Cria 2 colunas: a 1ª é o dobro da 2ª */
    /* grid-template-columns: 2fr 1fr; */
}
```

Espaçamento: gap

Em vez de usar margens nos filhos, usamos **gap** no pai para criar os canais entre as células.

CSS (style.css):

```css
.grid-container {
  border: 3px solid #2ecc71;
  background-color: #fff;
  padding: 10px;
  margin-bottom: 15px;
  display: grid; /* Grid já ativo para todos os exemplos de grid */
  gap: 10px;
}
```

Posicionamento por Linhas (grid-column)

Podemos dizer onde um item começa e onde termina na grade.

- A contagem é pelas **linhas da grade**, não pelas células.
- Numa grade de 3 colunas, temos as linhas 1, 2, 3 e 4 (a borda final).

CSS (style.css):

```
#item-header {
  /* Começa na linha 1, vai até a linha 4 (ocupa tudo) */
  grid-column: 1 / 4;
  background-color: #27ae60; /* Cor diferente para destacar */
}
```

Aprofundando nas Unidades de Medida

Não precisamos usar apenas **fr.** O Grid permite misturar unidades para criar layouts precisos.

- **px, rem, %**: Tamanhos fixos ou relativos ao pai.
- **auto**: O tamanho é definido pelo **conteúdo** do item (ou pelo espaço que sobrar).
- **fr**: Distribui o espaço **disponível** (o que sobrou depois de calcular os **px** e **auto**).

CSS (style.css - Teste no #grid-ex-1):

```
#grid-misto {
    display: grid;
    /* Coluna 1: 200px fixos (ex: Sidebar fixa)
       Coluna 2: auto (Tamanho do conteúdo mais largo)
       Coluna 3: 1fr (Ocupa todo o resto do espaço)
    */
    grid-template-columns: 200px auto 1fr;
    gap: 10px;
}
```

A Função repeat()

E se quisermos 12 colunas iguais? Escrever **1fr** doze vezes é ruim. Usamos a função **repeat().**

Sintaxe: repeat(número_de_vezes, tamanho)

CSS (style.css):

```css
#grid-repeat {
    display: grid;
    /* Cria 4 colunas iguais de 1fr */
    grid-template-columns: repeat(4, 1fr);

    /* Podemos misturar! */
    /* 1 coluna de 100px, depois 3 colunas de 1fr */
    /* grid-template-columns: 100px repeat(3, 1fr); */
}
```

Controle Mínimo e Máximo: minmax()

Uma das funções mais úteis para responsividade. Define que uma coluna deve ter *pelo menos* um tamanho, mas pode crescer até outro.

Sintaxe: minmax(minimo, maximo)

CSS (style.css):

```css
#grid-minmax {
  display: grid;
  /* As colunas terão no mínimo 150px.
    Se houver espaço sobrando, elas crescem (1fr).
    Se não houver, elas param em 150px (gera scroll horizontal ou quebra).
  */
  grid-template-columns: repeat(3, minmax(150px, 1fr));
}
```

Posicionamento Avançado: A palavra-chave span

No Slide 15, vimos como posicionar usando linhas de início e fim (**1 / 4**). Mas as vezes não queremos saber a linha final, queremos apenas dizer: "Ocupe 2 colunas".

Usamos a palavra-chave **span** (estender).

CSS (style.css):

```
/* No container #grid-posicao */
#item-header {
    /* Começa na linha 1 e estende por 3 colunas */
    grid-column: 1 / span 3;
    /* É o mesmo que grid-column: 1 / 4; */
}

#-item-sidebar{
    /* Começa onde estiver (auto) e estende por 2 linhas verticais */
    grid-row: span 2;
}
```

Linhas Negativas (Contando do Fim)

O Grid também permite contar as linhas de trás para frente usando números negativos.

- **-1** é sempre a **última linha** da grade.
- **-2** é a penúltima, etc.

Isso é perfeito para fazer um rodapé ou cabeçalho ocupar a largura total sem saber quantas colunas existem exatamente.

CSS (style.css):

```css
#item-footer {
    /* Começa na primeira linha e vai até a ÚLTIMA (-1) */
    grid-column: 1 / -1;
    background-color: #e74c3c; /* Destaque */
}
```

A Técnica "Mágica": grid-template-areas

Esta é a forma mais visual e intuitiva de criar layouts. Você "desenha" seu site no CSS usando nomes.

1. No PAI, defina o template (**grid-template-areas**).
2. Nos FILHOS, atribua o nome da área (**grid-area**).

CSS (style.css): →

```
#grid-areas {
  grid-template-columns: 1fr 3fr;
  grid-template-areas:
    /* Mapa do layout */
    grid-template-areas:
      "header header"
     "aside  main"
     "footer footer";
    gap: 10px;
}

/* Conectando os itens aos nomes do mapa */
#header { grid-area: cabeçalho; }
#aside  { grid-area: lateral; }
#main   { grid-area: principal; }
#footer { grid-area: rodape; }
```

HTML E CSS

Resumo:

- **FLEXBOX:** Use para **componentes** e alinhamentos em 1 direção.
  - *Exemplos*: Menu de navegação, alinhar botão e texto, galeria de cards simples.
- **GRID:** Use para **layout macro** da página em 2 direções.
  - *Exemplos*: Definir onde fica o Header, Sidebar, Main e Footer.

Jogo para aprender Flexbox: https://flexboxfroggy.com/

Dica de Ouro: Geralmente usamos **Grid** para a **estrutura da página** e **Flexbox** para **organizar o conteúdo** dentro dessas **áreas do Grid.**

ATIVIDADE

Layout "Santo Graal" Moderno

1

Objetivo: Refazer o layout da aula passada (Header, Nav, Main+Aside, Footer) usando Grid para a estrutura e Flexbox para o menu.

Instruções:

1. Crie a pasta **exercicio-santo-graal-moderno.**
2. Use o mesmo HTML da aula passada **removendo**:
   - a tag de abertura ** **.
   - a tag de fechamento ** ** correspondente (que fica antes do footer).
   - **MANTER** todo o resto **(header, nav, main, aside, footer**).
3. No **CSS**:
   - Use **display: grid** no **body**.
   - Crie áreas para **header, nav, main, aside, footer.**
   - Use **display: flex** na **ul** do menu para alinhar os links horizontalmente.

Referências

CSS TRICKS. **A Complete Guide to Flexbox**. Disponível em: https://css-tricks.com/snippets/css/a-guide-to-flexbox/. Acesso em: 17 nov. 2025.

CSS TRICKS. **A Complete Guide to Grid**. Disponível em: https://css-tricks.com/snippets/css/complete-guide-grid/. Acesso em: 17 nov. 2025.

FLEXBOX FROGGY. **Jogo para aprender Flexbox**. Disponível em: https://flexboxfroggy.com/. Acesso em: 17 nov. 2025.