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):

```
/* 1.1. Básico */
#flex-basico {
  display: flex; /* Ativa o Flexbox */
/* Os filhos agora são itens flexíveis */
}
```

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;
  height: 150px;
  align-items: center; /* Centraliza na altura do container verticalmente  */
}
```

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 */
  height: 150px;
}
```

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):

```css
/* 1.6. Wrap (Quebra de Linha) */
#flex-wrap {
  display: flex;
  flex-wrap: wrap; /* Permite quebrar linha */
}
```

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

}
```

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 */
}
```

Propriedades do FILHO: Order

Altera a ordem visual dos elementos sem precisar mudar o HTML. Útil para responsividade (mudar a ordem no celular).

- **Valor Padrão:** 0.
- Aceita números inteiros (..., -1, 0, 1, 2, ...).
- Menor número aparece primeiro.

```
/* 1.8. Order */
#flex-order {
    display: flex;
}
#flex-order .item-invertido {
    order: -1; /* Este item vai para o começo, mesmo sendo o último no HTML */
}
```

Propriedades do FILHO: align-self

Permite que um **único item** desobedeça a regra align-items definida no pai.

- **Valores**:
  - auto, flex-start, flex-end, center, baseline, stretch.

```
/* 1.9. Align-Self */
#flex-align-self {
    display: flex;
    align-items: center; /* Padrão: centro */
    height: 150px;
}
#flex-align-self .item-diferente {
    align-self: flex-end; /* Só este vai para o chão */
}
```

Professores:
Celso Giusti
Daniel Manoel Filho
Marlon Palata Fanger Rodrigues

GRID

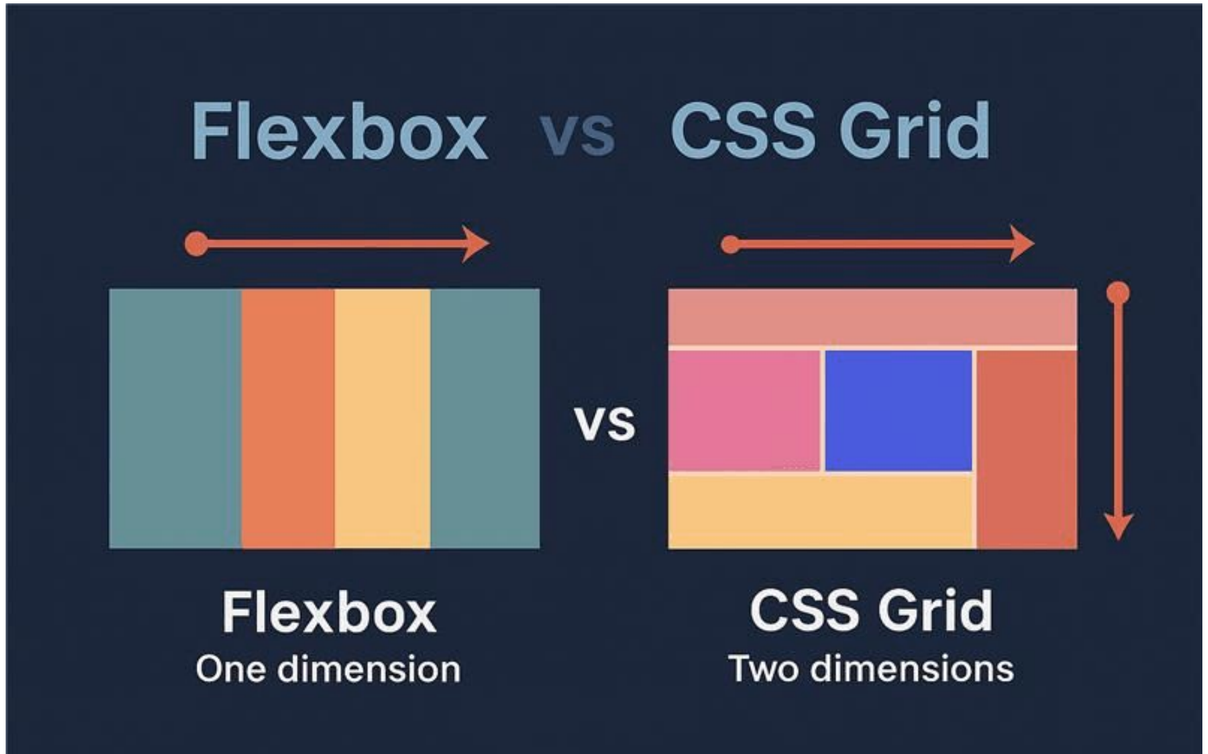
SENAI

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 a Grade

Definindo Colunas e Linhas Ativamos com display: grid.

- **grid-template-columns:** Largura das colunas.
- **grid-template-rows:** Altura das linhas.
- **Unidade fr:** Fração do espaço disponível.

```
/* 2.1. Básico */
#grid-basico {
display: grid;
grid-template-columns: 1fr 1fr 1fr;
}
```

Espaçamento: gap

Espaçamento: gap Cria canais (espaços) entre as células, sem adicionar margem externa.

- **row-gap:** Espaço entre linhas.
- **column-gap:** Espaço entre colunas.
- **gap:** Atalho (linha coluna).

```
/* 2.1. Básico */
#grid-basico {
/*...*/
  gap: 10px;
}
```

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):

```
/* 2.2. Misto */
#grid-misto {
display: grid;
grid-template-columns: 200px auto 1fr;
gap: 10px;
 }
/* Coluna 1: 200px fixos (ex: Sidebar fixa)
Coluna 2: auto (Tamanho do conteúdo mais largo)
Coluna 3: 1fr (Ocupa todo o resto do espaço)
*/
```

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
/* 2.3. Repeat */
#grid-repeat {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
}
   /* 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):

```
/* 2.4. Minmax */
#grid-minmax { display: grid;
grid-template-columns: repeat(3, minmax(150px, 1fr));
gap: 10px;
}
   /* 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).
   */
```

HTML E CSS - Grid

Grade Implícita e Fluxo

O Que Acontece Quando Faltam Espaços?

Imagine que você definiu um tabuleiro com 4 casas, mas tentou colocar 5 peças. O CSS Grid não joga a peça fora; ele **automaticamente cria uma nova casa** para acomodá-la. Essa é a **Grade Implícita**.

Podemos controlar como esses novos espaços são criados:

1. **grid-auto-flow (A Direção):**
   - row (**Padrão**): Cria uma nova **LINHA** abaixo da grade atual.
   - column: Cria uma nova **COLUNA** à direita da grade atual.
2. **grid-auto-rows / grid-auto-columns (O Tamanho):**
   - Define a altura (para linhas) ou largura (para colunas) desses novos espaços automáticos.

Grade Implícita e Fluxo

Exemplo:

```
.container {
    display: grid;
    /* Grade explícita de 2x2 (4 espaços) */
    grid-template-columns: 1fr 1fr;
    grid-template-rows: 100px 100px;

    /* Se tivermos um 5º item, crie uma nova LINHA de 50px para ele */
    grid-auto-flow: row; /* Padrão, nem precisaria escrever */
    grid-auto-rows: 50px;
}
```

Alinhamento de Itens na Grade

Propriedades do PAI: justify-items e align-items Controla o alinhamento dos itens **dentro de suas células** (se a célula for maior que o item).

1. **justify-items (Horizontal):** start, end, center, stretch (padrão).
2. **align-items (Vertical):** start, end, center, stretch (padrão).
3. **place-items:** Atalho (align justify).

```
/* 2.6. Alinhamento de Itens */
#grid-alignment-items {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 10px;
    height: 200px;
    justify-items: center; /* Horizontal na célula */
    align-items: center;   /* Vertical na célula */
}
```

HTML E CSS - Grid

Alinhamento da Grade

Propriedades do PAI: justify-content e align-content Se a grade total for menor que o container, como a grade inteira se posiciona?

1. **justify-content (Eixo X):** start, end, center, space-between, space-around, space-evenly.
2. **align-content (Eixo Y):** start, end, center, space-between...
3. **place-content:** Atalho.

```
/* 2.7. Alinhamento da Grade (NOVO) */
#grid-alignment-content {
    display: grid;
    grid-template-columns: 100px 100px;
    gap: 5px;
    height: 300px;
    justify-content: center; /* Grade no meio horizontal */
    align-content: center;   /* Grade no meio vertical */
}
```

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):

```css
/* 2.8. Posicionamento por Linhas (Start/End) */
#grid-linhas {
    display: grid; grid-template-columns: repeat(4, 1fr); gap: 10px;
}
.item-linha-especifica {
    grid-column: 1 / 3; /* Da linha 1 até a 3 */
    background-color: #e74c3c;
}
```

Posicionamento Avançado: A palavra-chave span

Muitas vezes não sabemos o número da linha final, só queremos que o item ocupe **"X espaços".** Usamos a palavra-chave **span**.

```
/* 2.9. Posicionamento com Span */
#grid-span {
    display: grid; grid-template-columns: repeat(4, 1fr); gap: 10px;
}
.item-span {
    grid-column: span 2; /* Ocupa 2 espaços */
    background-color: #9b59b6;
}
.item-span-row {
    grid-row: span 2; /* Ocupa 2 linhas verticais */
    background-color: #8e44ad;
}
```

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
/* 2.10. Linhas Negativas */
#grid-negativo {
    display: grid; grid-template-columns: repeat(4, 1fr);
    gap: 10px;}
.item-negativo {
    grid-column: 1 / -1;      /* Do começo até o fim absoluto */
    background-color: #34495e;
}
```

Alinhamento Individual: self

Permite alinhar um **único item** dentro de sua área de grade, diferente do padrão definido no pai. Um item específico ignora o alinhamento padrão da grade

- **justify-self (Horizontal):** start, end, center, stretch.
- **align-self (Vertical):** start, end, center, stretch.

```css
/* 2.11. Alinhamento Individual */
#grid-item-alignment {
    display: grid;
    grid-template-columns: 1fr 1fr;
    height: 200px;
    gap: 10px;
     justify-items: start;
}
```

```css
.item-diferente {
    justify-self: end;
    align-self: end;
    background-color: #d35400;
}
```

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): →

```css
/* 2.10. Template Areas */
#grid-areas {
    display: grid;
    grid-template-columns: 1fr 3fr;
    grid-template-areas:
        "header header"
        "aside  main"
        "footer footer";
    gap: 10px;
    height: 300px;
}
.area-header { grid-area: header; background-color: #d35400; }
.area-main { grid-area: main; background-color: #f39c12; color: #333; }
.area-aside { grid-area: aside; background-color: #e67e22; }
.area-footer { grid-area: footer; background-color: #d35400; }
```

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/

Gui para Grid: https://www.origamid.com/projetos/css-grid-layout-guia-completo/

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: Consolide seus conhecimentos criando um layout de site profissional. Você usará **CSS Grid** para definir a estrutura principal da página (cabeçalho, menu lateral, conteúdo, rodapé) e **Flexbox** para organizar os componentes internos (menu de navegação, rodapé). O grande desafio será aplicar uma técnica avançada do Grid (auto-fit e minmax) para criar uma galeria de cards que se adapta automaticamente a diferentes tamanhos de tela, quebrando linhas de forma inteligente sem precisar de configurações complexas.

Instruções:

1. **Preparação:**
   - Crie uma nova pasta para o projeto chamada exercicio-final-cards-responsivos.
   - Dentro dela, crie dois arquivos vazios: index.html e style.css.
2. **Estrutura HTML (index.html):**
   - Crie a estrutura básica de um documento HTML5.
   - Vincule o arquivo style.css no <head>.
   - No <body>, crie a seguinte estrutura semântica, utilizando as classes indicadas:
     - Um <header class="header"> contendo um título <h1> para o logo e um <nav> com uma lista <ul> (classe .menu) de 5 links. Adicione a classe .destaque-contato ao último item da lista (<li>).

ATIVIDADE

Layout "Santo Graal" Moderno

2

- Um <aside class="sidebar"> com um título <h3> e uma lista <ul> de links secundários.
- Um <main class="content"> com um título <h2> para a galeria e uma container (classe .grid-cards) que abrigará pelo menos 8 s filhas, cada uma com a classe .card.
- Um <footer class="footer"> contendo um parágrafo <p> para o copyright e um (classe .btn-topo) para um botão de "Voltar ao Topo".

2. **Estilização CSS (style.css):**
   - **Passo A: Reset e Estrutura Grid Principal**
     - Aplique um reset básico (margin: 0, padding: 0, box-sizing: border-box).
     - Defina o body como um Grid Container.
     - Crie um layout de 3 colunas: uma lateral fixa de 220px e duas colunas flexíveis (1fr).
     - Crie linhas que se adaptem ao conteúdo, garantindo que o rodapé fique sempre na parte inferior da tela (min-height: 100vh).
     - Use grid-template-areas para mapear o layout: Header no topo, Sidebar na esquerda, Content no centro/direita, e Footer na base.
     - Adicione um gap de 15px entre as áreas.

ATIVIDADE

Layout "Santo Graal" Moderno

3

- **Passo B: Atribuição de Áreas e Estilos Base**
  - Conecte cada elemento HTML principal (.header, .sidebar, .content, .footer) à sua respectiva área nomeada no Grid (grid-area).
  - Aplique cores de fundo e padding distintos para cada área para facilitar a visualização.
- **Passo C: Menu Flexbox (Cabeçalho)**
  - Transforme o .header em um container Flexbox para alinhar o logo e o menu nas extremidades (space-between), centralizados verticalmente. Permita quebra de linha (flex-wrap: wrap).
  - Transforme a lista .menu em Flexbox com wrap e gap.
  - Estilize os links (<a>) como botões (bloco, padding, fundo). Use white-space: nowrap para impedir que o texto quebre dentro do botão.
  - **Desafio order:** Use a propriedade order na classe .destaque-contato para fazer o link "Contato" aparecer visualmente como o primeiro item do menu, e dê a ele uma cor de destaque.

ATIVIDADE

Layout "Santo Graal" Moderno

4

- **Passo D: Sidebar e Footer (Alinhamentos Individuais)**
  - Na .sidebar, use align-self: start para que ela não estique até o rodapé, ficando alinhada ao topo.
  - No .footer, use Flexbox para separar o texto do botão, com wrap. Use align-self: start no botão .btn-topo para alterar seu alinhamento vertical padrão.
- **Passo E: Galeria de Cards Inteligente (O Desafio Principal)**
  - No container .grid-cards, ative o Grid.
  - **A Mágica:** Em vez de definir um número fixo de colunas, utilize a combinação repeat(auto-fit, minmax(200px, 1fr)). Isso fará com que o navegador crie tantas colunas quanto possível, desde que cada uma tenha pelo menos 200px. Se não houver espaço, os cards quebrarão para a próxima linha automaticamente.
  - Estilize os .card para que tenham uma aparência visual agradável (cor, padding, borda arredondada).

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.