Conteúdo

Módulo 9:

CSS Grid

CSS Grid Layout

O CSS Grid Layout, ou simplesmente CSS Grid, é um sistema de layout bidimensional que permite aos desenvolvedores criar interfaces de usuário complexas e responsivas com maior facilidade e consistência. Ele é especialmente útil para organizar elementos em linhas e colunas, permitindo um controle preciso sobre o alinhamento, o espaçamento e a distribuição dos elementos dentro de um contêiner.

Características principais do CSS Grid:

Estrutura baseada em linhas e colunas: Você pode definir quantas linhas e colunas desejar, além de seus tamanhos.

Flexibilidade de tamanho: As dimensões das linhas e colunas podem ser definidas em uma variedade de unidades, incluindo pixels, porcentagens, ou como uma ***fração*** *do espaço disponível (****fr****).*

Áreas de grid: É possível nomear áreas do grid para posicionar os elementos de forma mais intuitiva.

Alinhamento e justificação: Oferece controle completo sobre o alinhamento e justificação dos itens dentro do grid, tanto horizontal quanto verticalmente.

CSS Grid *torna o design de layouts complexos mais simples e direto, diminuindo a necessidade de usar hacks e técnicas de posicionamento menos eficientes. É amplamente suportado pelos navegadores modernos e é uma ferramenta essencial para desenvolvedores front-end que buscam criar designs responsivos e dinâmicos.*

Grid Container
O Grid Container é a tag que envolve os itens do grid, ao indicar display: grid, essa tag passa a ser um Grid Container.

1 • display
Define o elemento como um grid container.

```
display: grid;
// Torna o elemento um grid container.
display: inline-grid;
// Torna o elemento um grid container porém com comportamento
inline.
display: subgrid;
```

```
// Para grids dentro de grids (ainda não é suportado, porém você pode normalmente colocar display: grid; no grid dentro do grid que funciona).
```

https://codepen.io/uchoamaster/pen/KKLKPvg

2 • grid-template-columns

Define o número total de colunas que serão criadas no grid.

```
grid-template-columns: 100px 100px 100px 100px;
// Quatro colunas de 100px de largura são criadas
grid-template-columns: 1fr 2fr;

// Duas colunas são criadas, sendo a segunda com o dobro do tamanho da primeira. fr é uma unidade fracional. O tamanho do conteúdo é respeitado, ou seja, se o conteúdo na primeira coluna for maior que o da segunda, a primeira será maior.
grid-template-columns: minmax(200px, 1fr) 1fr 1fr;
// Três colunas são criadas, a primeira terá no mínimo 200px de largura e no máximo 1fr(isso significa que após 200px ela se expande da mesma forma que as outras colunas). As outras duas colunas vão ter 1fr.
grid-template-columns: repeat(3, 1fr);
// Cria 3 colunas com 1fr de tamanho. O repeat seria a mesma coisa que escrever 1fr 1fr 1fr.
grid-template-columns: repeat(auto-fit, minmax(100px, auto));
// Cria automaticamente um total de colunas que acomode itens com no mínimo 100px de largura.
```

https://codepen.io/uchoamaster/pen/OJYJLZP

3 • grid-template-rows

Define a quantidade de linhas no grid.

```
grid-template-rows: 50px 100px 50px 150px;
// Cria 4 linhas no grid, sendo a primeira com 50px, segunda 100px, terceira 50px e quarta 150px. Caso o grid necessite de mais linhas, elas terão o tamanho de acordo com o conteúdo.
grid-template-rows: 1fr 2fr;
```

```
// Cria 2 linhas no grid, sendo a segunda com cerca de duas vezes o tamanho da primeira.
```

https://codepen.io/uchoamaster/pen/ExzxYRE

4 • grid-template-areas

Define áreas específicas no grid. O ponto (.) pode ser utilizado para criar áreas vazias.

```
grid-template-areas:
"logo nav nav"
"sidenav content advert"
"sidenav footer footer";
// Cria 3 colunas e 3 linhas. [logo] ocupa a coluna 1, linha 1. [nav] ocupa da coluna 2 a 3, linha 1. [sidenav] ocupa a coluna 1, da linha 2 a 3. [content] ocupa a coluna 2, linha 2. [advert] ocupa a coluna 3, linha 2. [footer] ocupa da coluna 2 a 3, linha 3
```

https://codepen.io/uchoamaster/pen/vYwYBaw

5 • grid-template

Atalho para definir o grid-template-columns, grid-template-rows e grid-template-areas.

```
grid-template:

"logo nav nav" 50px

"sidenav content advert" 150px

"sidenav footer footer" 100px

/ 100px 1fr 50px;

// A primeira linha com 50px, segunda com 150px e terceira com 100px. A primeira coluna com 100px, a segunda 1fr e a terceira com 50px.
```

https://codepen.io/uchoamaster/pen/mdYddpr

6 • gap

Define o gap (gutter) entre os elementos do grid.

```
gap: 20px

// Define 20px entre os elementos do grid (linha e coluna).

column-gap: 20px

// Define 20px de distância entre as colunas.

row-gap: 20px

// Define 20px de distância entre as linhas.
```

https://codepen.io/uchoamaster/pen/gOJOOeg

7 • grid-auto-columns

Define o tamanho das colunas do grid implícito (gerado automaticamente, quando algum elemento é posicionado em uma coluna que não foi definida).

```
grid-auto-columns: 100px

// As colunas implícitas, geradas automaticamente, terão 100px de largura.
```

https://codepen.io/uchoamaster/pen/xxNxLzR

8 • grid-auto-rows

Define o tamanho das linhas do grid implícito (gerado automaticamente, quando algum elemento é posicionado em uma linha que não foi definida).

```
grid-auto-rows: 100px
```

```
// As linhas implícitas, geradas automaticamente, terão 100px
de altura.
```

https://codepen.io/uchoamaster/pen/eYaYerP

9 • grid-auto-flow

Define o fluxo dos itens no grid. Se eles vão automaticamente gerar novas linhas ou colunas.

```
grid-auto-flow: row

// Automaticamente gera novas linhas.

grid-auto-flow: column

// Automaticamente gera novas colunas.

grid-auto-flow: dense

// Tenta posicionar o máximo dos elementos que existirem nas
primeiras partes do grid (pode desorganizar o conteúdo).
```

https://codepen.io/uchoamaster/pen/PovoOaW

10 • grid

Atalho geral para definir o grid: grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns e grid-auto-flow

```
grid: 100px / 1fr 1fr
// Gera uma linha com 100px de altura e 2 colunas com 1fr.
grid: 100px / auto-flow 100px 50px
// Gera uma linha com 100px de altura. O grid-auto-flow é definido
como column (pois está logo antes da definição das colunas). Ele
também define o grid-auto-columns com 100px 50px
```

https://codepen.io/uchoamaster/pen/zYQYPab

11 • justify-content

Justifica os itens do grid em relação ao eixo x (horizontal).

```
justify-content: start

// Justifica os itens ao início.

justify-content: end

// Justifica os itens ao final.

justify-content: stretch

// Estica os itens.

justify-content: space-around

// Distribui espaço entre os elementos. (O início e final são
menores que os espaços internos).

justify-content: space-between

// Cria um espaço entre os elementos, ignorando o início e final.

justify-content: space-evenly

// Cria um espaço igual entre as colunas (no início e final
também).

justify-content: center

// Centraliza o conteúdo.
```

https://codepen.io/uchoamaster/pen/jOoOapz

12 • align-content

Alinha os itens do grid em relação ao eixo y (vertical).

```
align-content: start
// Alinha os itens ao início.
align-content: end
// Alinha os itens ao final.
```

```
align-content: stretch
// Estica os itens.
align-content: space-around
// Distribui espaço entre os elementos. (O início e final são menores que os espaços internos).
align-content: space-between
// Cria um espaço entre os elementos, ignorando o início e final.
align-content: space-evenly
// Cria um espaço igual entre as colunas (no início e final também).
align-content: center
// Centraliza o conteúdo.
```

https://codepen.io/uchoamaster/pen/zYQYPLb

13 • justify-items

Justifica o conteúdo dos itens do grid em relação ao eixo x (horizontal). Justifica em relação a célula.

```
justify-items: start
// Justifica os itens ao início.
justify-items: end
// Justifica os itens ao final.
justify-items: center
// Centraliza o conteúdo.
justify-items: stretch
// Estica os itens.
```

https://codepen.io/uchoamaster/pen/mdYdqzr

14 • align-items

Alinha o conteúdo dos itens do grid em relação ao eixo y (vertical). Alinha em relação a célula.

```
align-items: start
// Alinha os itens ao início.
align-items: end
// Alinha os itens ao final.
align-items: center
// Centraliza o conteúdo.
align-items: stretch
// Estica os itens.
```

https://codepen.io/uchoamaster/pen/rNgNYqP

15 • grid-column

Define quais colunas serão ocupadas pelo grid item. É possível definir uma linha de início e final, assim o item irá ocupar múltiplas colunas.

```
grid-column: 1

// O item ocupará a coluna 1.

grid-column: 1 / 3

// O item ocupará a coluna 1 e 2 (Sim, isso mesmo, 1 e 2, pois os valores 1 / 3 são referentes as linhas da coluna. Isso significa que começa na linha 1 (início do grid) e vai até a linha 3, que é o começo da terceira coluna).

grid-column-start: 2

// O item vai começar na linha 2.

grid-column-end: 4

// O item vai terminar na linha 4.

grid-column: span 2

// O item irá ocupar duas colunas a partir de onde ele estiver.
```

https://codepen.io/uchoamaster/pen/GRaROwr

16 • grid-row

Define quais linhas serão ocupadas pelo grid item.

Atenção aqui, pois essa linha é referente a row. Porém as chamadas grid lines que por tradução também significam linhas do grid, são diferentes. Uma row (linha), possui sempre 2 grid lines (linhas do grid), uma no início dela e uma no final dela.

```
grid-row: 1
// O item ocupará a linha 1.
grid-row: 1 / 3
```

```
// O item ocupará a linha 1 e 2 (Sim, isso mesmo, 1 e 2, pois os valores 1 / 3 são referentes as linhas do grid. Isso significa que começa na linha 1 (início do grid) e vai até a linha 3 do grid, que é o começo da terceira linha).
grid-row-start: 2
// O item vai começar na linha do grid 2.
grid-row-end: 4
// O item vai terminar na linha do grid 4.
grid-row: span 2
// O item irá ocupar duas linhas a partir de onde ele estiver.
```

https://codepen.io/uchoamaster/pen/JjqjOxP

17 • grid-area

Define a área do item do grid. É um atalho para grid-row-start, grid-column-start, grid-row-end, grid-column-end.

O z-index pode ser utilizado para manipular a posição no eixo Z do item. Ou seja, se um item for posicionado em cima de outro, o z-index controla qual vêm na frente.

```
grid-area: 1 / 2 / 4 / 3;
// Este é um atalho para:
grid-row-start: 1;
grid-column-start: 2;
grid-row-end: 4;
grid-column-end: 3;
grid-area: header;
// Vai posicionar o item na área definida como header.
```

https://codepen.io/uchoamaster/pen/yLWLPZx

18 • justify-self

Justifica o item do grid em relação ao eixo x (horizontal). Justifica em relação a célula.

```
justify-self: start
// Justifica o item ao início.
justify-self: end
// Justifica o item ao final.
justify-self: center
// Centraliza o conteúdo.
justify-self: stretch
// Estica o item.
```

https://codepen.io/uchoamaster/pen/gOJOXEW

19 • align-self

Justifica o item do grid em relação ao eixo y (vertical). Alinha em relação a célula.

```
align-self: start
// Alinha o item ao início.
align-self: end
// Alinha o item ao final.
align-self: center
// Centraliza o conteúdo.
align-self: stretch
// Estica o item.
```

https://codepen.io/uchoamaster/pen/eYaYeXx

Vamos praticar?

Com base nos layouts iniciais, vamos dar um show de criatividade e transformá-los em 3 sites incríveis usando CSS Grid! Prepare-se para se inspirar e levar suas habilidades com CSS Grid para o próximo nível!

1. Loja Virtual Responsiva com Grid e Animações:

https://codepen.io/uchoamaster/pen/wvbvPLp

Aprimoramentos:

- **Layout responsivo:** Adapta-se perfeitamente a qualquer tela.
- **Cartões de produtos com estilo:** Bordas arredondadas, sombra e transição suave no hover.
- **Cabeçalho e navegação fixos:** Sempre visíveis enquanto rola a página.
- **Rodapé com fundo escuro e texto claro:** Melhora a legibilidade.
- **Animação de hover nos produtos:** Torna a interação mais dinâmica.

2. Blog com Grid, Imagens em Destaque e Navegação Lateral:

https://codepen.io/uchoamaster/pen/YzbzEmG

Aprimoramentos:

- **Layout responsivo: Adapta-se perfeitamente a qualquer tela.**
- **Navegação lateral: Organize suas categorias e tags.**
- **Imagens em destaque: Chame a atenção dos leitores.**
- **Estilos para posts: Torne seus posts mais atraentes.**
- **Layout de uma única coluna em telas menores: Prioriza o conteúdo principal.**

Dicas Extras:

- **Personalize as cores e fontes para combinar com o estilo do seu blog.**
- **Adicione mais elementos à navegação lateral, como links para redes sociais.**
- **Crie diferentes layouts para diferentes tipos de posts.**
- **Use animações CSS para tornar seu blog ainda mais interativo.**

3. Dashboard Moderno e Responsivo

https://codepen.io/uchoamaster/pen/jOoOYNp

Explicação:

- **HTML:**
    - **Estrutura os elementos do dashboard: header, cards (Visão Geral, Novos Produtos, Atividades Recentes) e seus conteúdos.**
    - **Espaços para gráficos (`.grafico-torta`) e listas (`.produtos`, `.atividades`).**
- **CSS:**
    - **Layout responsivo com `grid-template-columns: repeat**

5 Atividades para Aprimorar suas Habilidades em CSS Grid

Desenvolva seu domínio em CSS Grid com estas 5 atividades práticas:

1. Layout de Galeria de Imagens:

- **Crie um layout em grid para exibir fotos em uma galeria.**
- **Use diferentes proporções de imagens e ajuste o espaçamento entre elas.**
- **Implemente recursos como lightbox ou hover effects para aprimorar a experiência do usuário.**

2. Layout de Blog com Sidebar:

- **Crie um layout de blog com conteúdo principal e sidebar lateral.**
- **Posicione os elementos de forma responsiva para diferentes tamanhos de tela.**
- **Utilize recursos como sticky sidebar ou menu de navegação responsivo para melhorar a usabilidade.**

3. Layout de Lojas Virtuais com Produtos:

- **Crie um layout de loja virtual com produtos em grid.**
- **Exiba informações como nome, imagem, preço e botão de compra para cada produto.**
- **Implemente filtros e ordenação de produtos para facilitar a navegação do usuário.**

4. Layout de Painel de Controle:

- **Crie um layout de painel de controle com diferentes seções para exibir dados e gráficos.**
- **Utilize cards, grids e outros elementos para organizar as informações de forma clara e intuitiva.**

- **Implemente recursos como menus suspensos, botões e links para interação do usuário.**

5. Layout de App de Música:

- **Crie um layout de app de música com seções para artistas, álbuns, músicas e playlists.**
- **Utilize grids, cards e outros elementos para organizar as informações de forma amigável.**
- **Implemente recursos como player de música, controles de volume e busca para aprimorar a experiência do usuário.**

Dicas Extras:

- **Comece com layouts simples e vá aumentando a complexidade gradativamente.**
- **Utilize ferramentas online como o CSS Grid Playground para testar e visualizar seus layouts em tempo real.**
- **Inspire-se em sites e apps existentes para encontrar ideias e soluções criativas.**
- **Não tenha medo de experimentar diferentes técnicas e estilos de layout.**

Terminando, basta salvar no trello o link dos projetos de css grid que salvou no seu github.