**GRID LAYOUT**

- Faz uma divisão com linhas e colunas (duas dimensões)

- É possível colocar elementos onde quiser nessa divisão

**🡪 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.

**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). \*/

**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 \*/

**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 \*/

**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

\*/

**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;

/\*

Primeira linha com 50px, segunda com 150px e terceira linha com 100px

Depois do “/” são definidas as larguras das colunas

Primeira coluna com 100px, segunda com 1fr e terceira com 50px

\*/

**grid-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. \*/

**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 \*/

**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 \*/

**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). Boa aplicação para galeria de imagens \*/

**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 \*/

**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 \*/

**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 \*/

**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 \*/

**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 \*/

**🡪 GRID ITEM**

Os Grid Itens são os filhos diretos do Grid Container. Um grid item pode ser explicito ou implícito. Explicito é quanto você define ele explicitamente no container e implícito é quanto ele é criado automaticamente para preencher o conteúdo no grid.

**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: 1 / -1

/\*

O item vai se expandir de ponta a ponta no grid

O -1 representa a última 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 \*/

**grid-row**

Define quais linhas serão ocupadas pelo grid item.

Atenção aqui, pois esse 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 \*/

**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 \*/

**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 \*/

**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 \*/

**### REPRESENTAÇÃO SIMPLES NO HTML**

<!-- Elemento grid -->

<div class=”container”>

<!-- Itens do grid -->

<header></header>

<main></main >

<aside></aside>

<footer ></footer >

</div>

**### COLUNAS** (aplicação no grid)

.container {

display: grid;

grid-template-columns: 1fr 1fr;

ou

grid-template-columns: repeat(2, 1fr);

}

/\*

1fr 1fr divide em duas colunas iguais

1fr = 1 unidade fracional

- grid-template-columns: 3fr 1fr;

Faria a primeira coluna ser 3x maior que a segunda

\*/

.container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

}

/\*

auto-fit 🡪 preenche automaticamente todo o espaço disponível

minmax 🡪 tamanho mínimo que o objeto pode ter e o tamanho máximo

\*/

**### LINHAS** (aplicação no grid)

.container {

display: grid;

grid-template-rows: 20vh 70vh 10vh;

}

/\*

Três linhas com alturas diferentes

vh = view height (altura da tela)

20vh = 20% da altura da tela

\*/

**### ESPAÇAMENTOS** (aplicação no grid)

.container {

grid-row-gap: 20px;

grid-column-gap: 50px;

ou

grid-gap: 20px 50px;

}

/\* Espaçamento entre as linhas e colunas \*/

**### POSICIONAMENTO** (aplicação no item)

.container header {

grid-column-start: 1;

grid-column-end: 3;

ou

grid-column: 1/3

}

/\*

Tendo o grid (div.container) três colunas o header, que é um elemento dentro do grid, irá compor todas as três, iniciando na primeira e terminando na terceira

\*/

.container footer {

grid-row-start: 3;

grid-row-end: 4;

ou

grid-row: 3/4

}

/\* Mesmo esquema das colunas \*/

**### POSICIONAMENTO COM TEMPLATE AREAS**

PRIMEIRAMENTE, APLICANDO NO GRID

.container {

display: grid;

grid-template-areas: “header header”

“main aside”

“footer footer”;

/\*

Funciona como uma matriz para o posicionamento:

Neste caso seriam 2 colunas e 3 linhas.

Header ocupando a primeira linha e as duas colunas;

Main ocupando a segunda linha na primeira coluna;

Aside ocupando a segunda linha e a segunda coluna;

Footer ocupando a terceira linha e as duas colunas

\*/

ou

grid-template-areas: “h h”

“m a”

“f f”;

/\* Pode ser colocado qualquer nome \*/

ou ainda

grid-template-areas: “h .”

“m a”

“. f”;

/\*

O . (ponto) caracteriza um lugar vazio

Neste exemplo, na posição da primeira linha com a segunda coluna teria um espaço vazio, assim como na terceira linha com a primeira coluna

\*/

}

AGORA, APLICANDO NOS ITENS DO GRID

.container header {

grid-area: header;

ou

grid-areas: h;

}

.container main {

grid-area: footer;

ou

grid-areas: f;

/\*

Pode ser colocado em qualquer posição

O nome não justifica a posição, até porque pode ser qualquer nome

\*/

**### ALINHAMENTO**

PROPRIEDADES

justify-content

align-content

justify-items

align-items

justify-self

align-self

- O grid é bidimensional, portanto, temos os eixos x e y

- Eixo x: posicionamento horizontal, da esquerda para a direita (justify)

- Eixo y: posicionamento vertical, de cima para baixo (align)

CONTENT (aplicação no grid)

justify-content e align-content nos permite alinhar o grid relativo ao espaço fora do próprio grid

O uso dessas propriedades é raro, pois só é aplicado caso o grid seja menor que a área definida, por exemplo: quando usamos px para o tamanho do grid e não fr (que é flexível), o grid pode ficar pequeno em relação ao tamanho real disponível do próprio grid (sua área)

Valores para **justify-content** (horizontal) e **align-content** (vertical)

start *posição no início*

end *posição no final*

center *posição ao centro*

stretch *ocupação do espaço disponível (padrão)*

space-between *espaçamento entro os itens*

space-around *espaçamento ao redor*

space-evenly *espaçamento contínuo (igual)*

<div class=”container”>

<div class=”item”></div>

<div class=”item”></div>

<div class=”item”></div>

</div>

.container {

justify-content: center;

align-content: center;

}

/\*

Alinha o grid ao centro

Para isso é necessário que o grid seja menor que o seu tamanho disponível (não esteja com a propriedade stretch)

\*/

ITEMS (aplicação no grid)

justify-items e align-items nos permite alinhar os itens do grid dentro do espaço disponível da célula que eles habitam

Para isso é necessário que o item tenha um tamanho menor que a própria célula, ou seja, que não preencha todo o espaço disponível dentro da célula (não esteja com a propriedade strectch)

Valores para **justify-items** (horizontal) e **align-items** (vertical)

start *posição no início*

end *posição no final*

center *posição ao centro*

stretch *ocupação do espaço disponível*

.container {

justify-items: center;

align-items: center;

ou

place-items: center;

}

/\*

Alinha os itens ao centro da sua célula, ou seja, ao centro do seu próprio espaço disponível dentro do grid

place-items centraliza nas duas direções ao mesmo tempo

\*/

SELF (aplicação no item)

justify-self e align-self nos permite alinhar o item em si

Faz a mesma coisa que o justify-items e o align-items, porém é aplicado diretamente no item do grid

Valores para **justify-self** (horizontal) e **align-self** (vertical)

start *posição no início*

end *posição no final*

center *posição ao centro*

stretch *ocupação do espaço disponível*

.container .item {

justify-self: center;

align-self: center;

}

/\*

Também alinha os itens ao centro da sua célula, ou seja, ao centro do seu próprio espaço disponível dentro do grid

\*/