Guia CSS Grid Layout

Grid Container

Grid Item

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.

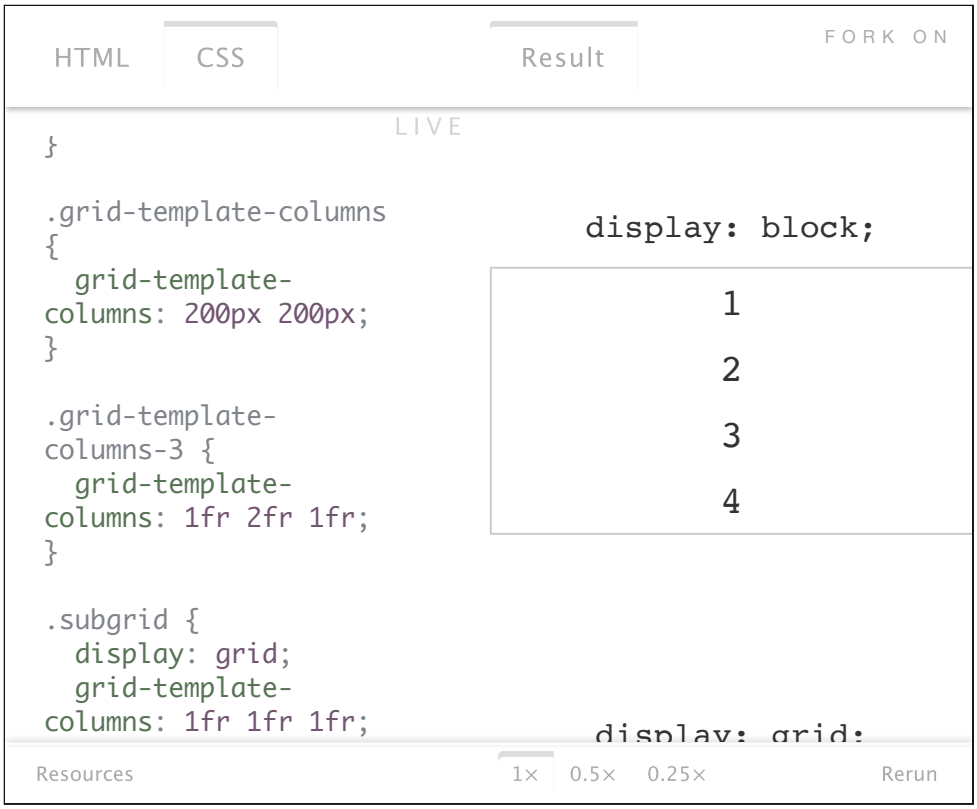
1 • display

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).
```



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.
```

HTML CSS Result EDIT ON

LIVE

```
/* Grid */
.grid {
  display: grid;
}

/* 100px é o valor total, ignora conteúdo, margem e etc. Respeita apenas o min-width do item.*/
.grid-template-columns-1 {
  grid-template-columns: 100px 100px 100px 100px;
}

/* fr é uma unidade fracional */
.grid-template-columns-2 {
  grid-template-columns: 1fr 2fr;
}
```

grid-template-columns: 100px 100px 100px 100px;

1 2 3

grid-template-columns: 100px 100px 100px 100px;

PalavraTe 2 3 Pal

Resources 1× 0.5× 0.25× Rerun

3 • grid-template-rows

```
grid-template-rows: 50px 100px
```

Define a quantidade de linhas no grid.

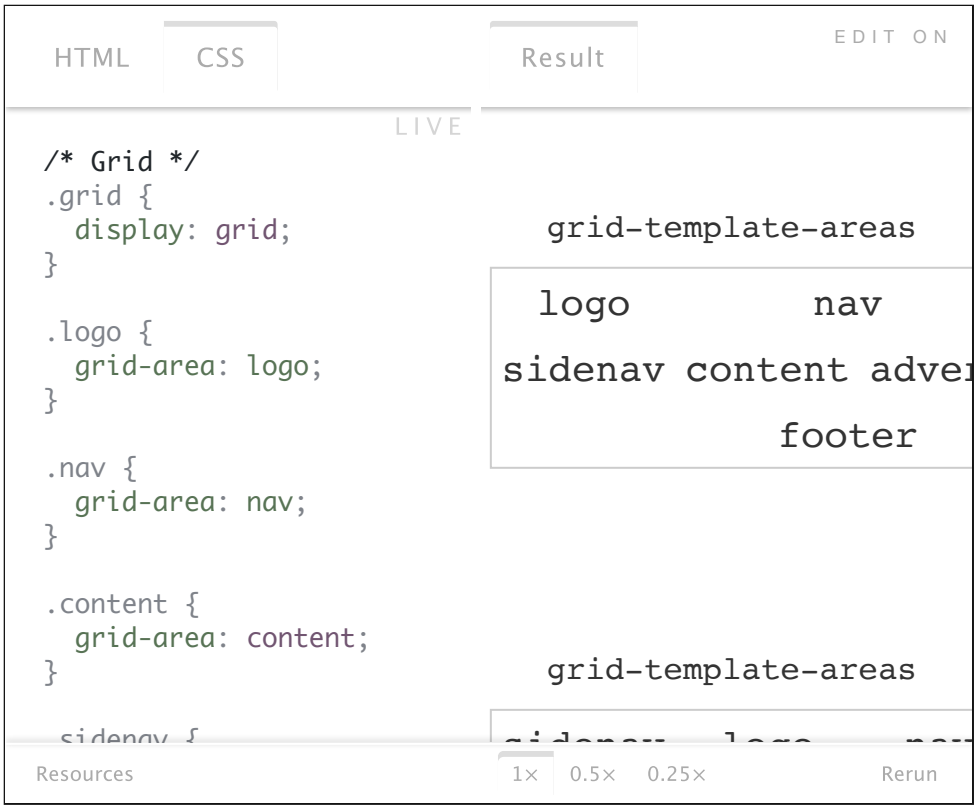
```
50px 150px;
// Cria 4 linhas no grid, sendo a primeira com 50px, segunda 100px, terceira 50px e quarta 150px. Caso o grid necessite de mais linhas, elas terão o tamanho de acordo com o conteúdo.

grid-template-rows: 1fr 2fr;
// Cria 2 linhas no grid, sendo a segunda com cerca de duas vezes o tamanho da primeira.
```

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
```



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.
```

HTML CSS Result EDIT ON LIVE

```
/* Grid */
.grid {
  display: grid;
}

.grid-template-1 {
  grid-template: 50px 1fr / 100px 1fr 100px;
}

.grid-template-2 {
  grid-template:
    "logo nav nav" 50px
    "sidenav content advert" 150px
    "sidenav footer footer" 100px
    / 100px 1fr 50px;
}

 logo {
```

Resources

6 • gap

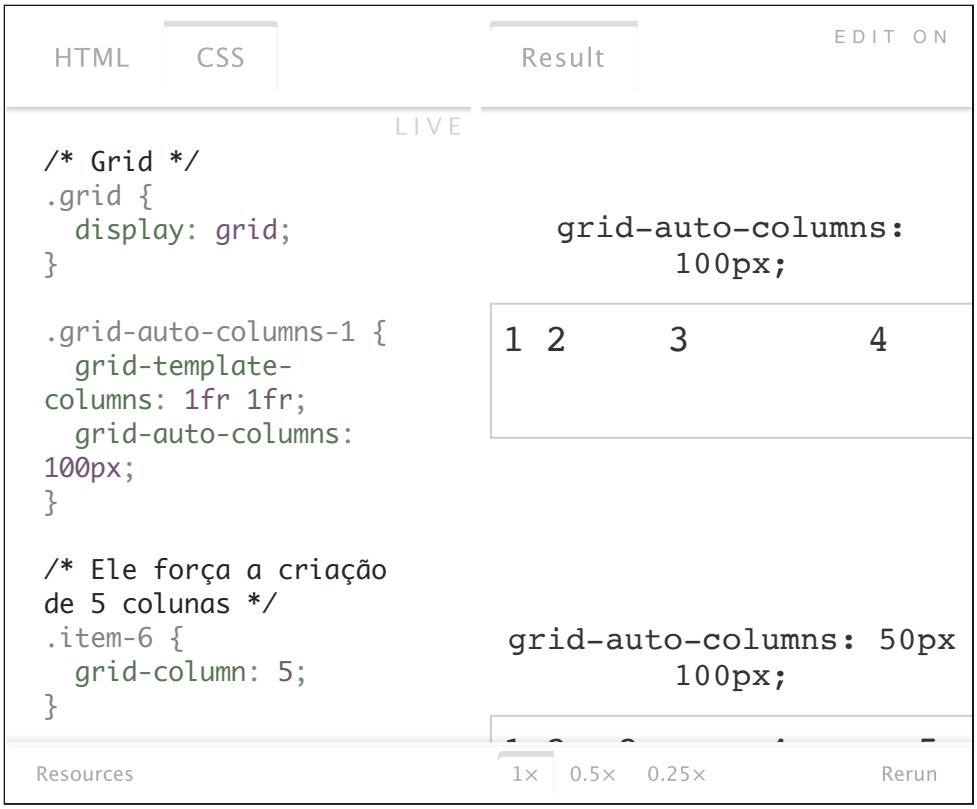
Define o gap (gutter) entre os elementos do grid.

```
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.
```

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.
```



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.
```

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.
```

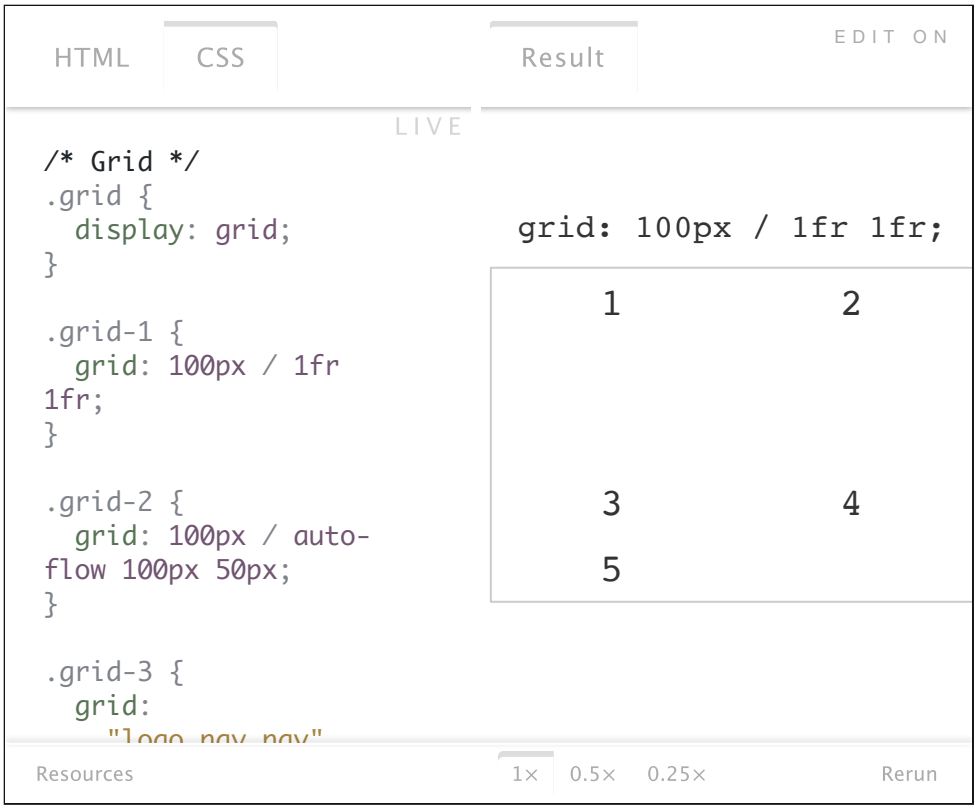
```
grid-auto-flow: dense
// Tenta posicionar o máximo dos
elementos que existirem nas primeiras
partes do grid (pode desorganizar o
conteúdo).
```

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
```



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.
```

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.
```

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.
```

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.
```

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.

1 • 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.
```

2 • 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.
```

3 • 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.
```

4 • 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.
```

5 • 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.
```

Principais referências utilizadas para a criação deste guia: CSS Tricks e Grid By Example