

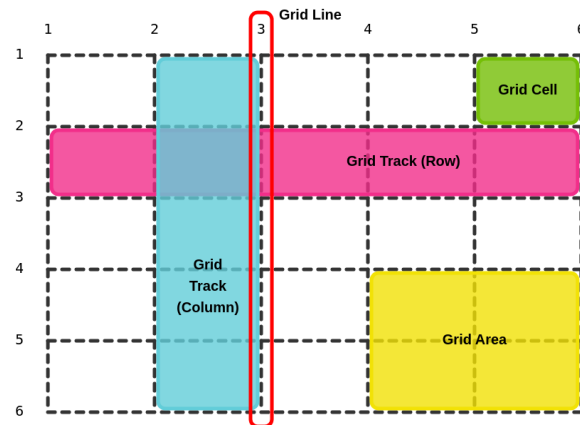
Grid Layout

Grid Layout x Flexbox

Qual a principal diferença entre os dois ?

O grid layout não substituiu o flexbox, ele veio para complementar. Enquanto o flexbox posiciona os elementos em uma única linha o grid layout veio para estruturar layouts mais complexos.

Estrutura



Grid Templates

Definindo linhas e colunas:

```
.grid-container{
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-template-rows: 150px 100px;
}
```

Tornando o código mais enxuto:

```
.grid-container{
  display: grid;
  grid-template-columns: repeat(3, 100px);
  grid-template-rows: 150px 100px;
}
```

Adicionando espaços entre a célula:

```
.grid-container{
  /*Espaço entre as linhas*/
  grid-row-gap: 40px;
  /*Adicionando espaço entre as colunas*/
  grid-column-gap: 20px;
  /*Atalho: linha e coluna*/
  grid-gap: 20px 5px;
}
```

Nova unidade de medida (fr):

Essa nova unidade veio junto como o grid layout.

Ela serve para dividir o espaço excedente de forma fracionada.

Exemplo:

```
grid-template-columns: repeat(3, 1fr);
```

Posicionando grid-itens

Para posicionar um grid-item utiliza-se as grid lines como referências, informando em qual grid line o elemento começa e termina.

```
.grid-itens:nth-child(1){  
  background-color: red;  
  grid-column-start: 3;  
  grid-column-end: 4;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}
```

Pode-se também utilizar o -1 para indicar que o elemento vai até a ultima grid-line:

```
.grid-itens:nth-child(1){  
  background-color: red;  
  grid-column-start: 1;  
  grid-column-end: -1;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}
```

Atalho para definir o inicio e final de linha e coluna:

```
.grid-itens:nth-child(1){  
  background-color: red;  
  grid-column: 1 / -1;  
  grid-row: 1 / 2;  
}
```

Mesclando

Forma tradicional:

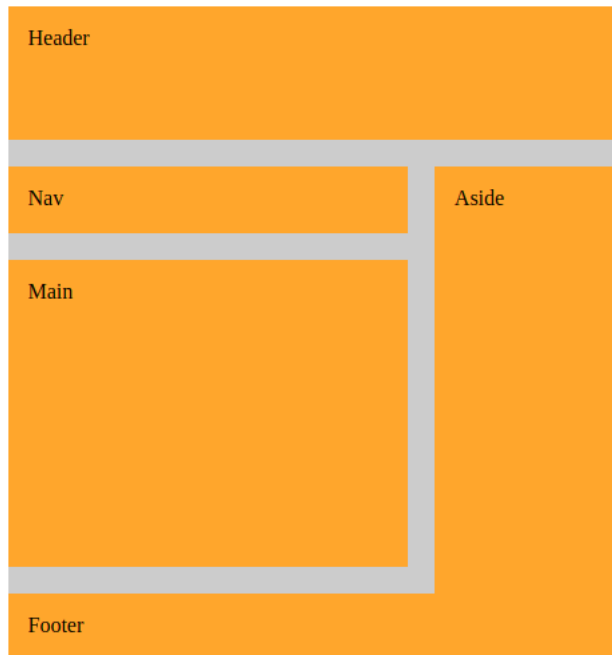
```
/*Mesclar: Forma tradicional*/  
.grid-itens:nth-child(2){  
  grid-row: 1 / 3;  
}
```

Forma usando o span (o span indica a quantidade de posições que o elemento ira ocupar)

```
.grid-itens:nth-child(2){  
  grid-row: 1 / span 2;  
}
```

Estrutura de Layout

Resultado desejado:



Utilizando os códigos acima para chegar a esse resultado:

```
/*Grid*/
.grid-conteiner{
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 100px 50px 1fr 50px;
  grid-gap: 20px;
}

.header{
  grid-column: 1 / -1;
}

.nav{
  grid-column: 1 / 3;
}

.main{
  grid-column: 1 / 3;
  min-height: 200px;
}

.aside{
  grid-row: 2 / -1;
  grid-column: 3 / -1;
}

.footer{
  grid-column: 1 / -1;
  grid-row: 4 / -1;
}
```

Obtendo um resultado semelhante utilizando grid área:

```
/*Grid*/
.grid-conteiner{
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 100px 50px 1fr 50px;
  grid-gap: 20px;
  grid-template-areas: "header header header"
}
```

```

        "nav    nav    aside"
        "main   main   aside"
        "footer footer aside";

    }

    .header{
        grid-area: header;
    }

    .nav{
        grid-area: nav;
    }

    .main{
        grid-area: main;
        min-height: 200px;
    }

    .aside{
        grid-area: aside;
    }

    .footer{
        grid-area: footer;
    }

```

Observações sobre o **templente-áreas**

- Não é possível mesclar colunas.
- Para definir um espaço em branco por conversão utiliza-se um “.”.

Linhas nomeadas

Construindo o layout proposto com linhas nomeadas.

```

/*Grid*/
.grid-container{
    grid-template-rows: [header-start] 100px [header-end nav-start] 50px [nav-end main-start] 1fr [main-end footer-start] 50px [foo
    grid-template-columns: [col1-start] 1fr [col1-end col2-star] 1fr [col2-end col3-start] 1fr [col3-end];
    grid-gap: 20px;
}

.header{
    grid-column: col1-start / col3-end;
}

.nav{
    grid-column: col1-start / col2-end;
}

.main{
    grid-column: col1-start / col2-end;
    min-height: 200px;
}

.aside{
    grid-row: nav-start / footer-end;
    grid-column: col3-start / col3-end;
}

.footer{
    grid-column: col1-start / col3-end;
    grid-row: footer-start / footer-end
}

```

Observações:

- É possível mesclar
- Não é tão simples quanto o grid-area

Ao utilizar as linhas nomeadas podemos utilizar um atalho, ao definir uma linha com `[exemplo-start]` e `[exemplo-end]`, basta só definir `exemplo` que automaticamente será ocupada a área de start e end. Exemplo:

```
.footer{
  grid-column: col1-start / col3-end;
  grid-row: footer
}
```

Podemos observar que na definição das `grid-template-columns` ficou mais. É possível utilizar o `repeat` e simplificar esse código? Sim, conforme o exemplo a baixo.

```
/*Grid*/
.grid-container{
  grid-template-rows: [header-start] 100px [header-end nav-start] 50px [nav-end main-start] 1fr [main-end footer-start] 50px [footer-end];
  grid-template-columns: repeat(3, [col-start] 1fr [col-end]);
  grid-gap: 20px;
}

.header{
  grid-column: col-start 1 / col-end 3;
}

.nav{
  grid-column: col-start 1 / col-end 2;
}

.main{
  grid-column: col-start 1 / col-end 2;
  min-height: 200px;
}

.aside{
  grid-row: nav-start / footer-end;
  grid-column: col-start 3 / col-end 3;
}

.footer{
  grid-column: col-start / col-end 3;
  grid-row: footer
}
```