Grid Layout

Certified Developer
The Ultimate Tech Degree

DigitalHouse >
Coding School

Temas

1 Grid e conceitos

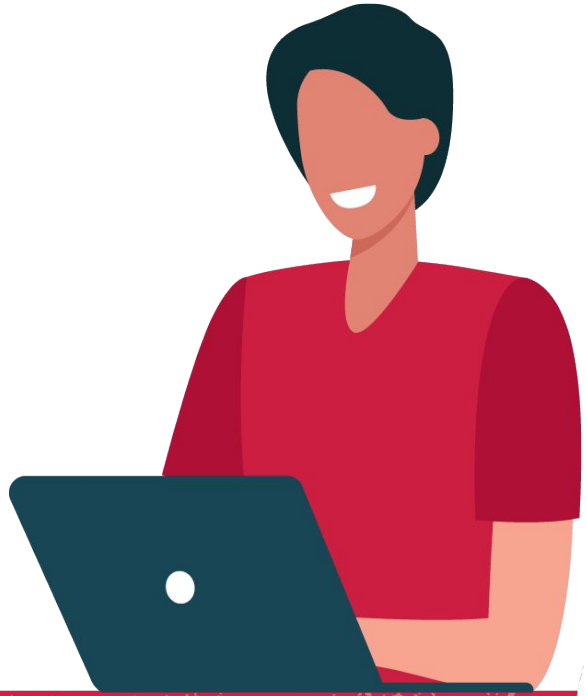
1. Objetivo
2. Criando um Grid
3. Conceitos de Coluna
4. Conceitos de Linha
5. Revisão e Conclusão

1 Objetivo

Objetivo

Ao contrário do Flexbox, que serve para construirmos layouts em uma única direção (vertical ou horizontal), o Grid nos permite construir layout bidimensionais (vertical e horizontal).

O conceito do Grid é literalmente a tradução da palavra, consiste em montar uma 'Grade' e irmos distribuindo elementos por ela, até atingirmos o resultado esperado.

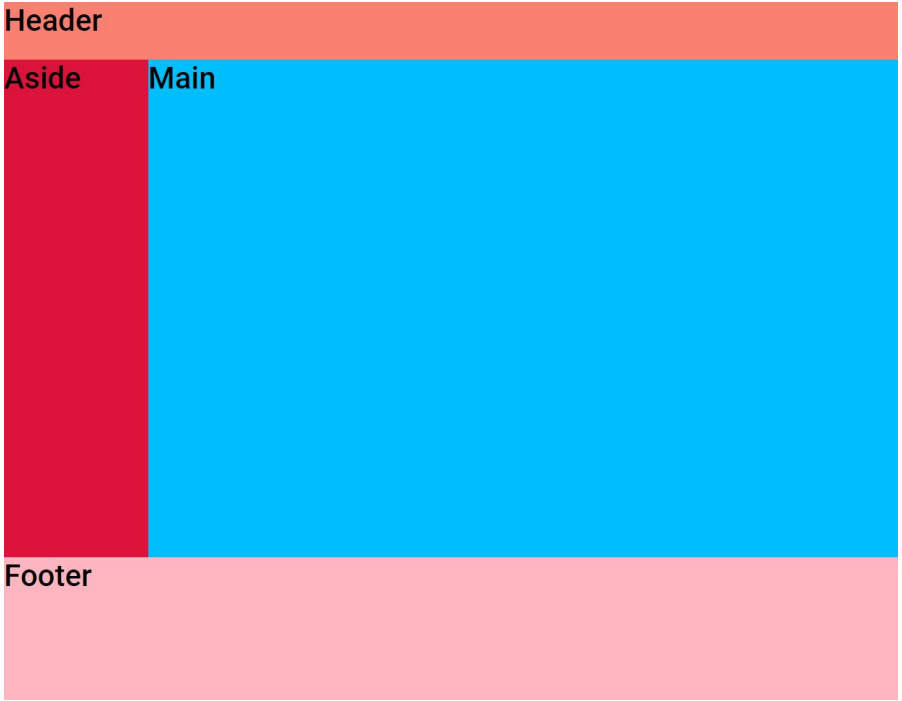


Objetivo

A criação do Grid se assemelha à criação do Flexbox, sempre iremos ter um elemento pai que irá dizer como os Elementos filhos devem se comportar dentro dele.

Objetivo

Nossa intenção é criar um layout de um site padrão, onde ele terá um Header para o menu Principal, um Aside para itens referentes à página na esquerda, um Main para exibirmos o conteúdo principal da nossa página, e por fim, um Footer para exibirmos outras informações do site.



2 | Criando um Grid

Criando um Grid

Como iremos falar do Layout Principal do nosso site, podemos usar o próprio Body como pai dos nossos elementos.

Agora que entendemos o conceito do Grid, vamos aplicá-lo. Para isso, crie um Header, Aside, Main e Footer dentro do seu Body.

```
<body>

    <header>Header</header>
    <aside>Aside</aside>
    <main>Main</main>
    <footer>Footer</footer>

</body>
```

Criando um Grid

Após isso, dê cores diferentes aos Elementos Filhos do nosso Body para conseguirmos separá-los. De preferência, escreva algo dentro de cada elemento, como o nome de sua própria tag para melhor visualização do que vai acontecer.

```css
body header {
    background-color: salmon;
}
body aside {
    background-color: crimson;
}
body main {
    background-color: deepskyblue;
}
body footer {
    background-color: lightpink;
}
```

Criando um Grid

No CSS, referencie o seu Body e atribua a propriedade display com o valor grid para ele.

De início, nada aconteceu porque justamente nós não definimos um formato de Grid para o nosso Elemento Pai, apenas dizemos que ele é e pode aceitar propriedades relacionadas à Grid.

```
body {
    display: grid;
}
```

Criando um Grid

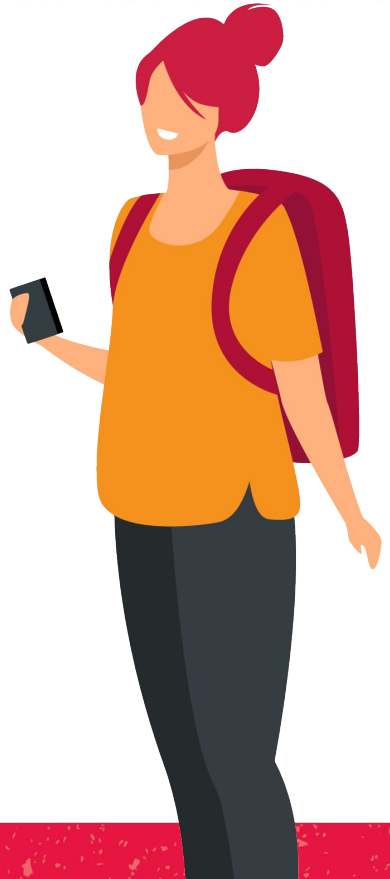
Agora que já criamos o grid, precisamos definí-lo. Mas antes disso, precisamos entender o conceito de Linhas e Colunas!

3 Conceitos de Coluna

Conceitos de Colunas

Em uma tabela normal, nós possuímos linhas e colunas para separar ou até mesmo agrupar informações... e no Grid não é diferente.

Com a propriedade 'grid-template-columns' nós podemos dizer a largura que as colunas da nossa Grid terão. Para visualizarmos isso perfeitamente precisamos desenvolver o olhar crítico e entender o que será e o que não será uma coluna...

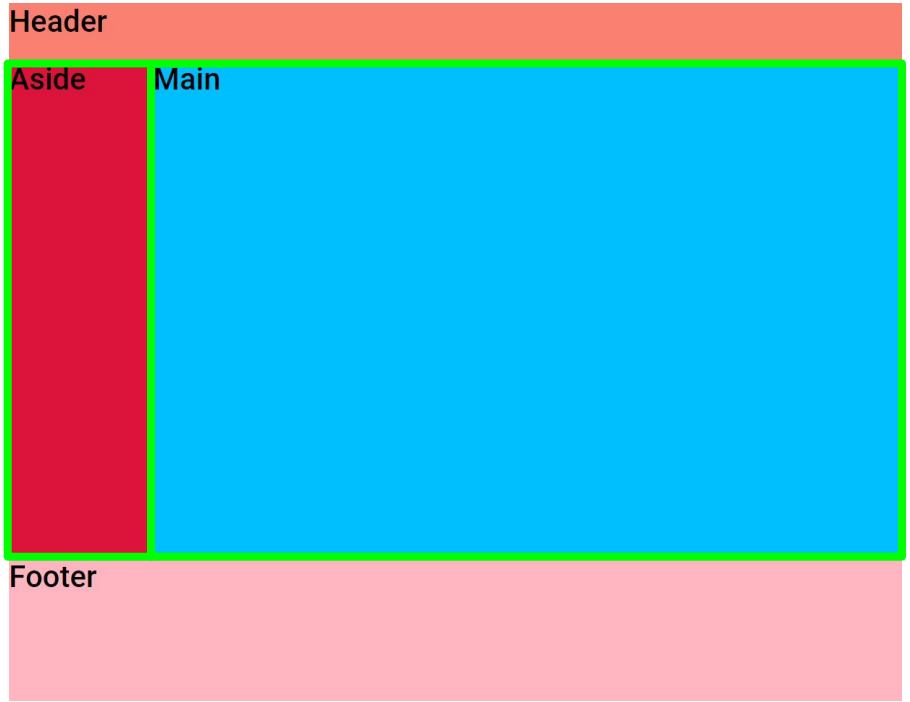


Conceitos de Colunas

Nesse caso, se formos analisar nosso layout, há duas colunas.

A primeira menor, que serve para o Aside e a segunda maior, que serve para o Main.

Apesar de enxergarmos essas duas colunas apenas no contraste, entre Aside e Main elas não estão somente nesses dois elementos...



Conceitos de Colunas

Sempre que definimos uma coluna em nosso Grid ela será Global, ou seja, ela não servirá apenas para o Aside mas sim para o nosso Grid no geral.

Isso pode parecer um pouco confuso agora, mas irá fazer mais sentido quando formos distribuir os nossos elementos nessa Grid criada.

Conceitos de Colunas

Agora que já entendemos o funcionamento das Colunas, nós podemos defini-las utilizando 'grid-template-columns', como observamos a nossa Grid irá precisar apenas de 2 colunas.

```css
body {
    grid-templates-columns: 200px 1fr;
}
```

Segunda coluna

Definimos a largura que nossa segunda coluna irá ter.

Primeira coluna

Definimos a largura que nossa primeira coluna irá ter.

Unidade de medida fr

Após feito isso, você irá notar que os nossos elementos se agruparam de jeitos diferentes em duas colunas, uma menor (200px) e outra maior (1**fr**).

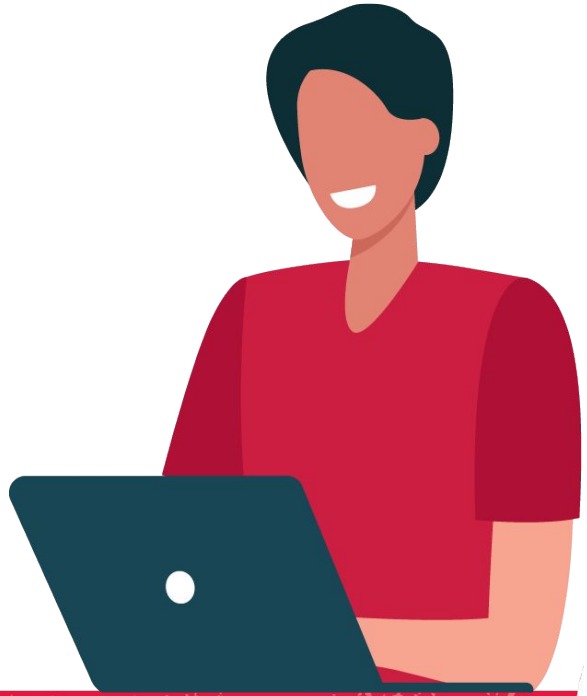
A unidade de medida **fr** é uma unidade que nos ajuda a calcular o espaço total para dividirmos, funciona quase como a porcentagem. Então, nesse caso, como só temos duas colunas e a segunda está sendo representada como 1fr, então ela pegará todo o espaço disponível para a segunda coluna.

4 Conceitos de Linha

Conceitos de Linha

Com as nossas Colunas já definidas e utilizando o 'grid-template-columns', agora podemos dizer para o nosso Elemento Pai como será a distribuição dos Elementos Filhos.

Mas antes, vamos nomear os nossos elementos para que o Grid possa entender qual elemento agrupar!



Conceitos de Linha

Para isso, você precisará definir um nome de referência para cada Elemento Filho, utilizando o '**grid-area**'.

O nome de referência pode ser qualquer coisa, mas é sempre bom nomear de maneira que você irá lembrar facilmente.

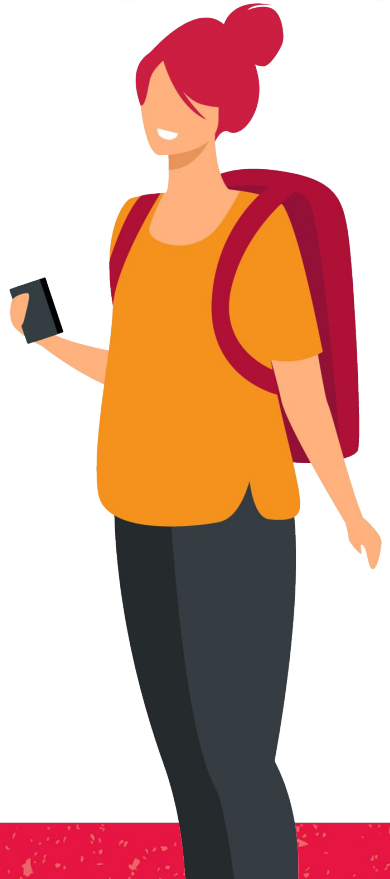
```
body header {

    background-color: salmon;
    grid-area: header;

}

body aside {

    background-color: cr
    grid-area: aside;

}
```

```
body main {

    background-color: deepskyblue;
    grid-area: main;

}

body footer {

    background-color: lightpink;
    grid-area: footer;

}
```

Conceitos de Linha

Após definir um nome de referência para cada Elemento Filho, nós podemos finalmente agrupá-los na Grid. Para isso, iremos utilizar o '**grid-template-areas**' diretamente em nosso Elemento Pai.

Com ele, é possível dizermos ao nosso Elemento Pai quantas colunas cada Elemento Filho irá ocupar, baseado no nome que definimos como referência utilizando '**grid-area**' em nossos Elementos Filhos.



Conceitos de Linha

No nosso caso, o nosso Header irá ocupar duas colunas, Aside apenas a primeira, Main apenas a Segunda e Footer as duas colunas.

A parte legal é que assim conseguimos visualizar com clareza quantas colunas cada elemento está ocupando.

```
body {

    display: grid;
    grid-template-columns: 200px 1fr;
    grid-template-areas:
    'header header'
    'aside main'
    'footer footer';

}
```

Aprofundando no Grid Templates Areas

Para entendermos melhor o grid-templates-areas é só imaginarmos que cada palavra representa um elemento e cada espaço uma coluna, ou seja:

```css
body {
    grid-templates-areas:
    'header header'
    'aside main'
    'footer footer';
}
```

Header ocupando as duas colunas

Aside ocupando a primeira coluna e Main a segunda

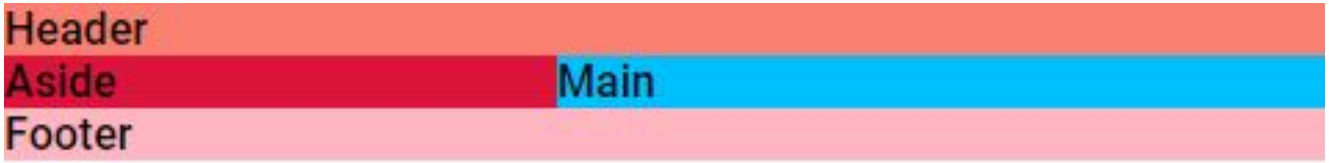
Footer ocupando as duas colunas

Primeira coluna

Segunda coluna

Conceitos de Linha

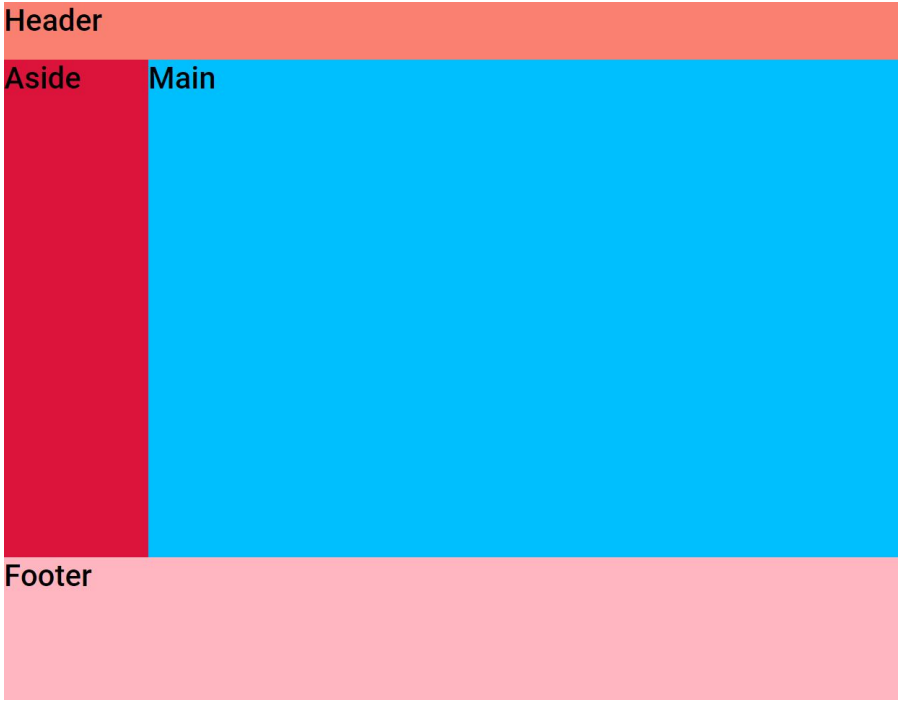
Após isso, teremos algo parecido com nosso Layout inicial:



Conceitos de Linha

Para finalizar, basta apenas darmos algumas alturas para nossos elementos Filhos e pronto, temos o resultado utilizando Grid.

Claramente esse exemplo está ‘feio’ mas representa muito bem o poder do Grid :)



Hora da revisão

Conclusão

- Vimos que é possível **construirmos layouts** mais complexos facilmente com Grid, que seriam bem trabalhosos utilizando Flexbox ou Position.

- Podemos **definir a largura de Colunas** e especificar quantas Colunas cada Elemento Filho poderá ter.


DigitalHouse>
Coding School