**Flexbox e Grid**

**Flexbox:**

Unidimensional: O Flexbox é projetado para layouts unidimensionais, ou seja, lida com o alinhamento e distribuição de elementos em uma única direção: linha (horizontal) ou coluna (vertical).

Alinhamento e Distribuição: Ideal para distribuir espaço e alinhar itens dentro de um contêiner, especialmente quando o tamanho dos itens é dinâmico ou desconhecido.

Exemplo de Uso: Criar menus de navegação horizontais ou verticais, barras de ferramentas ou grupos de botões.​

**Grid:**

Bidimensional: O Grid é projetado para layouts bidimensionais, permitindo o posicionamento de elementos tanto em linhas quanto em colunas simultaneamente.

Estrutura de Layout Completa: Ideal para criar layouts complexos, como grades de conteúdo, onde é necessário controlar o posicionamento de elementos em duas direções.​

Exemplo de Uso: Desenvolver a estrutura de uma página com cabeçalho, rodapé, barra lateral e áreas de conteúdo, organizadas em uma grade.​

**Quando Usar Cada Um:**

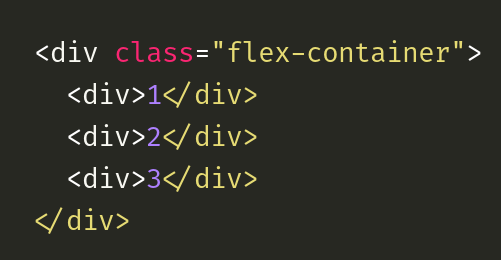
Flexbox: Utilize quando precisar alinhar e distribuir elementos em uma única direção, como em listas horizontais ou verticais, ou quando os itens devem se ajustar dinamicamente ao espaço disponível.​

Grid: Opte pelo Grid quando a estrutura do layout exigir o alinhamento de elementos em ambas as direções, como em layouts de página completos com múltiplas seções e áreas de conteúdo.

Obs: Flexbox e Grid podem ser usados em conjunto. Por exemplo, você pode usar o Grid para definir a estrutura principal da página e o Flexbox dentro de algumas seções para alinhar e distribuir elementos de forma eficiente.​

**Flexbox - Prática**

**Display flex** só é utilizado no elemento pai, que seria o container, os elementos internos desse container ou seja, os filhos, recebem outro tipo de configuração.



flex-container = pai

Div 1,2 e 3 = filhos

**Flex-direction ->** se você não definir um flex-direction em seu container, ele por padrão já preenche com “flex-direction: row;” (No caso seria em linha). Caso você queira definir que o seu container se comporte em colunas ao invés de linha, basta colocar como Flex-direction: column;

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>FlexBox</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="container">

<div class="num">1</div>

<div class="num">2</div>

<div class="num">3</div>

</div>

</body>

</html>

.container{

background-color: aqua;

display: flex;

}

.num {

background-color: brown;

border: 1px solid black;

padding: 20px;

text-align: center;

color: white;

}

Quando você coloca para que seu container ali definido fique em display: flex, podemos ver uma outra configuração interessante acontecer quando não inserimos mais modificações, no caso agora seria do **flex-wrap.** O que seria essa modificação, quando optamos por deixar o padrão que seria flex-wrap: nowrap; ao adicionarmos elementos em nossa página, quando diminuirmos o tamanho da tela exibida ele irá criar uma barra de rolagem, sendo assim não irá mostrar os itens de forma responsiva em nossa página. Para isso usamos da seguinte forma:

**Flex-wrap: wrap;** (Ele vai pegando os itens que criamos e vai jogando para baixo dentro do nosso container, reajustando os mesmos para caber na tela e não criar a barra de rolagem).  
**Flex-wrap: wrap-reverse;** (Ao invés de jogar os elementos para baixo, ele joga os elementos para cima, seria o inverso do que o wrap faz).

Obs: **Flex-wrap** você utiliza no elemento PAI.

Abaixo um exemplo de como seria o “padrão” :

.container{

background-color: aqua;

display: flex;

flex-direction: row;

flex-wrap: nowrap;

}

Uma forma interessante de se trabalhar com flex-direction e flex-wrap seria você resumir os dois em apenas uma única linha, mais conhecido como shorthand. Para facilitar você pode fazer da seguinte forma:  
  
.container{

background-color: aqua;

display: flex;

**flex-flow: row wrap;**

}

Com o flex-flow você cria uma shorthand do flex-direction e flex-wrap, no caso do exemplo acima você está alinhando eles em linha e não criando uma barra de rolagem.

Quando estamos utilizando o display:flex, podemos alinhar nossos elementos ao centro, ao fim da tela, e para que isso seja possível utilizamos o **justify-content**. Vale lembrar que essa propriedade alinha nossos elementos no eixo horizontal como mostrado na imagem acima.

Utilizando o justify-content, podemos alinhar de diferentes formas nossos elementos:

justify-content : flex-end; ( coloca no final)

justify-content : flex-start ( coloca no começo)

justify-content : center ( centraliza)

justify-content : space-between ( coloca o primeiro e o último elemento colado nas extremidades e deixa os demais com um espaço igual)

justify-content : space-evenly (deixa todos com o mesmo espaço de distância)

justify-content : space-around( cria uma espécie de caixa e centraliza os itens dentro dessa caixa imaginaria que ele criou)

Agora, caso você queira alinhar eles de forma vertical, podemos utilizar uma outra forma, que seria com o **align-items**.

Com o align-items, podemos fazer da seguinte forma:

align-items: center

align-items: flex-end

align-items: flex-start

Caso você utilize o comando align-items da forma abaixo, quando você utilizar junto o flex-flow: row wrap; ao quebrar os elementos para baixo, ele irá criar um espaço entre os elementos superiores e inferiores. Se isso não for o que você procura em seu design, basta utilizar a propriedade **align-content**.

Utilizando **align-items:**

.container{

background-color: aqua;

display: flex;

flex-flow: row wrap;

height: 500px;

align-items: center;

}

.num {

background-color: brown;

border: 1px solid black;

padding: 50px;

text-align: center;

color: white;

}

Utilizando **align-content:**

.container{

background-color: aqua;

display: flex;

flex-flow: row wrap;

align-content: flex-start; ( pode usar os mesmos de align-items)

height: 400px;

}

.num {

background-color: brown;

border: 1px solid black;

text-align: center;

padding: 50px;

color: white;

}

Pode ser que em alguns cenários você queira colocar seus itens alinhados ao centro, porém um ou outro você queira que fique com um outro posicionamento do container em geral, para isso você tem a opção de utilizar o **align-self.** Neste caso, essa propriedade é aplicada nos FILHOS, assim você pega o elemento que quer colocar em outra posição, e aplica essa propriedade, dessa forma você não afeta todo o design da sua página.

Flexbox é uma forma de organizar, alinhar e distribuir melhor os itens de um container , mesmo não sabendo o tamanho do item

Propriedade para colocar um espaço entre os elementos no flex, ao inves de padding, utiliza o **gap.**

**Flex-grow** funciona quase da mesma forma que o z-index, quanto maior, mais é a prioridade de espaço que será ganho.

**Flex-shrink** ele coloca a capacidade do item de diminuir, lembrando que no grow e aqui o padrão é 1.

**Grid - Prática**

Com o grid você consegue alinhar linhas e colunas em sua página, começando pelo

**grid-template-columns:** 200px 200px 200px; O que esse comando faz? Ele cria três colunas de 200px cada, supondo que em seu html você tenha criado 9 divs, ele irá criar 3 colunas de 200px e 3 linhas automaticamente. Se você no lugar de colocar o número exato de pixels, quiser colocar auto, ele irá se adaptar ao tamanho da tela, conforme a mesma for diminuindo.

Agora quando formos atualizar o tamanho de nossas linhas, utilizamos o seguinte conceito **grid-template-rows:** 200px 100px 50px; nesse caso em específico ele está dizendo que a primeira linha vai ter de altura 200px, a segunda 100px e a terceira 50px.

Ao invés de utilizarmos px, podemos utilizar o **fr**, que no caso ele ocupará uma fração do espaço disponível em tela, se comparado ao auto, é a mesma coisa, porém, você pode mudar o tamanho de outras colunas proporcionalmente, por exemplo, sua primeira coluna terá 1fr e a segunda 2fr, é o mesmo conceito de prioridade do z-index, a segunda coluna terá um tamanho maior se comparada com a primeira.

Lembrando que a fr se ajusta a tela, o px ele se mantém o mesmo tamanho.

Ao utilizar o grid-template-columns podemos otimizar ao digitar o tamanho das colunas usando o **repeat()**, como ele funciona, basta você digitar repeat(5, 200px); que ele irá criar 5 colunas de 200px

Podemos definir também tamanhos mínimos para nossas colunas utilizando o minmax, da seguinte forma abaixo grid-template-columns: repeat(5, minmax(1 fr, 2 fr)); No caso dessa linha ele irá criar 5 colunas com no minimo de 1fr e no máximo de 2fr ao redimensionar nossa tela utilizando o columns, podemos utilizar o justify-items, dessa forma os itens vão se alinhar dentro de sua célula criada, mas, de acordo com o que selecionarmos, seja

ao centro, início ou fim.

Se utilizarmos o grid-template-rows, podemos utilizar o **align-items** para alinhar nossos blocos verticalmente, da mesma forma que horizontalmente, utilizando as mesmas

propriedades, já o justify-content ele funciona da mesma forma que o display flex, alinhando os itens.(Place-items seria uma shorthand de align items e justify-items)

Existem formas de você criar espaços em sua página e colocá-los em diferentes tamanhos e posições, por exemplo, podemos usar o grid-columns supondo que você crie um html em que tenha header, section1, section2, section3 e um footer, se você utilizar o **grid-column** da seguintes forma

header

grid-column: 1

section1

grid-column: 2

section2

grid-column: 3

section3

grid-column: 1

footer

grid-column: 2

Isso quer dizer que header e section3 ocuparão a mesma coluna e section1 e footer ocuparão a coluna 2 e section2 criará e ocupará uma terceira coluna. Mas caso você queira

que o header por exemplo, ocupe 2 colunas das 3 para ter um espaço maior, você pode criar da seguinte forma usando o span

header

grid-column: span 2

section1

grid-column: 2

section2

grid-column: 3

section3

grid-column: 1

footer

grid-column: 2

Podemos alinhar nossas linhas em cada elemento que criamos, utilizando o justify-self para horizontal e align-self para vertical, lembrando que para o align funcionar você deve definir antes um tamanho para a linha no container pai com o grid-template-rows.

Se você não definir o espaço para as divs, elas irão ocupar o espaço total que foi ditado no início para elas align-items e justify-items alinham as caixas dentro do bloco em que elas foram criadas.  
  
**Quando usar grid-template-areas ou grid-row e grid-column**

Cenário Melhor escolha

Layout fixo com poucos elementos principais (header, main, aside, footer...) grid-template-areas

Layout dinâmico com muitos blocos repetidos (ex: cards, galeria, produtos) grid-column / grid-row com auto-fill / auto-fit

Layout responsivo que muda com o tamanho da tela   
grid-template-areas com media queries

Precisa de sobreposição ou layouts bem personalizados  
grid-row e grid-column