**Display Flexbox**

    display: flex;

**adiciona o display flex ao container (onde vai tá os outros itens), por padrão é row, mesmo sem colocar o flex-direction.**

    flex-direction: row;

**row é o padrão do display flex, mesmo sem colocar.**

**Column – coloca os itens um abaixo do outro**

**Column reverse – um abaixo do outro com a ordem reversa, ao fim do container.**

**Row-reverse – um ao lado do outro, mas ao fim do container e com a ordem reversa.**

**CROSS AXIS**

**Row: horizontal**

**Column: vertical**

    flex-wrap: wrap;

**define se o item vai ou não ter uma quebra de linha. Por padrão é nowrap.**

**Wrap-reverse: os itens são renderizados de forma reversa.**

**Ex:**

    width: 40%;

**quando coloca essa propriedade nos itens que estão dentro do container, os itens vão ocupar 40% da tela. E quando não for possível que todos os itens ocupem 40%, automaticamente o item vai para baixo.**

    flex-flow: column wrap;

**agrupa o flex-direction + o flex-wrap, nessa ordem.**

    justify-content: flex-start;

**permite alinhar o container na posição horizontal, quando a direção for row, e alinhar verticalmente quando a a direção for column.**

**Faz apenas o alinhamento e não altera a ordem dos itens.**

**Flex-start é o valor padrão.**

**Center: alinha os itens ao centro (no topo) da página, quando for row.**

align-items: center;

**alinhamento baseado no conteúdo**

**alinhamento vertical quando o flex-direction do container é row**

**alinhamento horizontal quando o flex-direction do container é column.**

**Align-items: baseline (o alinhamento do container é baseado em seu conteúdo)**

**Ex: se o 1 item tem o margin-top de 40px, mesmo assim os outros itens estarão alinhados ao texto dele, e não na posição inicial (contando com o margin-top)**

 align-items: flex-start;

**alinha os itens pelo início do item**

 align-items: flex-end;

**vai usar o final do item pra alinhar**

**Valor padrão do align-items é stretch: permite que os itens cresçam igualmente dentro do container (independente do conteúdo)**

**Ex: se dentro do container o heigth for de 500px, com essa propriedade descrita, automaticamente os itens que estão dentro passam a ter 500px de altura !! quando o item não tiver o heigth declarado na sua propriedade. Para o stretch funcionar, o heigth deve estar declarado apenas no container!!**

    align-content: center;

**faz um alinhamento vertical (na posição row) apenas quando os itens tem uma quebra de linha- ou seja- apenas quando tiver o wrap, e tiver mais de uma linha.**

**Ex: mesmo quando houver a quebra de linha os itens sempre estarão no centro do container**

**\*\*UTILIZAR PARA RESPONSIVIDADE\*\***

**Ordenando itens**

*.flex-container* div*:nth-child*(3) {

            background-color: red;

            order: 1;

        }

**Todos os itens tem ordem inicial 0, ao pegar o terceiro filho da div (que tem quatro filhos) e colocar ordem 1, ele foi pra última posição.**

order: 0;

**Ao voltar a ordem desse elemento pra 0, ele vai ocupar a mesma posição que foi imposta no html.**

order: -1;

**ele passa a ocupar a primeira posição (nesse caso), pois os outros itens tem ordem 0**

align-self: center;

**alinha o item – por padrão o align-self é stretch**

**Flex-Grow**

* **Define os tamanhos dos elementos dentro do container. Ex: foi definido um width de 100px a 4 elementos que estão dentro do container, mas o container tem 600px. Sem o flex-grow vai sobrar um espaço de 200px.**
* **Então, utilizando o flex-grow, quando definimos um flex-grow 1 para todos os elementos, todos serão divididos igualmente.**
* **No caso de colocar o flex-grow 2 em apenas um elemento, ele vai receber o dobro do valor dos demais itens**
* **\*\*se o container for menor que o tamanho definido de cada item, não se aplica o flex-grow e sim o flex-shrink\*\*. Ex: 4 itens têm width de 100px, totalizando 400px. E o container 300px.**

*.flex-container* div {

            width: 100px;

          flex-grow: 0;

**continua exatamente igual**

flex-grow: 1;

**todos vão crescer de forma igual**

*.flex-container* div*:nth-child*(1) {

            flex-grow: 2;

        }

**Neste caso, vai ser alterado o primeiro elemento dito na div do html, e não o elemento 3 que passou a ter a primeira posição (no exemplo do order)**

**Flex-Shrink**

* **Define como os elementos do container vão encolher quando o tamanho do container for menor que o tamanho dos itens**
* **Ex: um contêiner com 5 elementos de 100px cada e um contêiner de 400px**

flex-shrink: 1;

**define que todos os itens vão encolher igualmente**

*.flex-container* div*:nth-child*(1) {

            flex-shrink: 2;

}

**Nesse caso, o primeiro elemento vai diminuir 2 vezes mais que os outros**

**Flex-basis**

* **Define a dimensão do elemento dentro do container**
* **Tomar cuidado ao definir um width ou heigth, pois quem vai prevalecer é o flex-basis**
* **Aplicado no main axis**
* **Em column muda a altura e em row muda a largura**

flex-basis: 100px;

*.flex-container* div {

            flex-basis: 100px;

            width: 300px;

        }

**nesse caso prevalece o flex-basis. E se não tivesse o flex-basis o width ia definir sua altura e não sua dimensão no main axis**