Css Flexible Box Module

Flex-direction:

Existem 4 direções do flex-direction

Flex-direction: row;

Flex-direction: row-reverse;

Flex-direction: column;

Flex-direction: column-reverser;

Quando a **flex-direction: row;** está configurado no nosso idioma o **main-axis** sempre vai da esquerda para a direita e o **cross-axis** sempre de cima para baixo;

Porém quando eu colocar o **flex-direction: row-reverse; ele se inverte.** O **main-axis será da direita da esquerda** o **cross-axis NÃO TEM INVERSÃO;**

Para a configuração do **flex-direction: column;** **o main-axis vai de cima para baixo** e o **cross-axis vai da esquerda para a direita;**

Quando temos **o flex-direction: column-reverse; o main-axis** (eixo principal) **vai na vertical**, porém de baixo para cima e o **cross-axis** **vai ser deitado da esquerda para a direita**

Flex-wrap (empacotamento ou encapsulamento) valor padrão é **flex-wrap: nowarap;**

A quebra do wrap é sempre no sentido do cross-axis.

**Flex-wrap: wrap-reverse;**

A quebra do wrap é sempre no sentido contrário do cross-axis;

**ALINHAMENTO MAIN-AXIS > Justify-content:**

vai fazer o alinhamento em relação ao main-axis, se você alinhar o conteúdo com **o flex-flow: column;** ele vai alinhar verticalmente o conteúdo

**Justify-contend: flex-start**; vai alinhar o conteúdo no **main-start** do container;

**Justify-contend: flex-end;** vai condensar o conteúdo no **main-end** do container;

**Justify-contend: center**; vai alinhar no centro do main-axis **(dependendo do flow que for configurado)** distribuindo os espaços em branco igualmente entre o main-start e o main-end.

**Justify-contend: space-between;** vai alinhar o 1° item no main-start e o último item no main-end, deixando os demais centralizado no main-axis comespaçamento igual entre eles;

**Justify-contend: space-evely;** vai colocar os elementos dispostos dentro do container de forma que antes e depois dos itens tenhamos espaços iguais entre eles, podemos falar que esse é o mais **simétrico entre todos eles;**

**Justify-contend: space-around;** vai dividir o espaço e deixar espaço entre o ITEM e alinhar no centro, deixando uma certa forma um “espaçamento duplo” entre os itens, e no começo e final fica apenas um espaço.

**Align-items: segue o alinhamento do cross-axis;**

**O valor padrão É align-items: strech;**

**Align-items: strech;** vai esticar os itens no mesmo tamanho do cross-axis;

**Align-items: flex-start;** vai grudar sempre no cross-axis;

**Align-items: flex-end;** vai grudar sempre no final do cross-axis;

**Align-items: flex-center;** vai grudar sempre no centro do cross-axis.

**Propriedades aplicadas ao FLEX-CONTAINER**

**\*Cross-axis > align-items\***

**\*Mains-axis > jutify-content\***

Para deixar o alinhamento do item no centro do container, basta deixar as propriedades do justify-content:**center;** (container) e align-items:**center**;(item.

OBS: quando a gente cria um container a parte de fora é flex, não a parte de dentro, mas podemos pegar um ou mais elementos e torná-los flex.

**Flex item**

**Order: todo item em flex-box tem o pai(container), e todo elemento dentro do container “pai”, tem o valor de order 0.**

**Valores do Align-self:**

**Auto:** ele vai herdar a característica do alinhamento vertical do seu pai

**Flex-start:** é o alinhamento no cross-axis

**Flex-end:** é o alinhamento no cross-axis

**Center:** é o alinhamento no dentro cross-axis

**Strech:** vai esticar no tamanho do eixo transversal (cross-axis)

**Flex-basis**

**Valor padrão: auto**

**Você pode definir a largura de um elemento, porém ele vai variar conforme o tamanho do container (qual o formato da água? Depende do “container” que ela está)**

**É interessante colocar no flow o nowrap ( flex-flow: row nowrap )tanto no row quanto no collun.**

Valor Padrão: **Flex-shrink: 1; Flex-grow: 0;**

Shrink **é encolher** e Grow **é crescer.**

para deixar os itens mais flexíveis ao tamanho do container tem que deixar tanto o flex-shring quanto o flex-grow com o valor de 1.

O flex-grow ele funciona tanto nowrap quanto no wrap.

Flex-basis + flex-grow + flex-shrink

Você pode substituir:

Flex-basis: 150px;

Flex Grow: 0

Flex-shrink:1

Por: flex: **0**(Grow) **1**(Shrink) **150px**(basis), seguindo sempre essa sequencia.

Flex= flex-grow + flex shrink + flex basis.

Existem algumas configurações muito utilizadas no flex.

Para isso alguns valores já são pré-configurados

Ex: O valor **FLEX: INITIAL;** é a mesma coisa que dizer que o flex é

0(Grow) 1(shrink) auto (basis) (**flex: 0 1 auto**)

**Flex: none;** é a mesma coisa que dizer que o flex é

0(Grow) 0(shrink) auto (basis) (**flex: 0 0 auto**)

**Flex: auto;** é a mesma coisa que dizer que o flex é

1(Grow) 1(shrink) auto (basis) (**flex: 1 1 auto**)

**Essa é a mais flexível entre todos.**

Tem códigos que tem apenas um valor no FLEX.

Ex:

**FLEX: 3;**

O valor 3 é do GROW, por que isso?

Porque o flex tem sequência Grow > Shrink > Basis. Como foi definido valor apenas para o grou os demais seguem o valor de auto

Grid Layout

Oque é grid layout?

CSS Grid Layout é um sistema para criação de layout baseado em uma grade **bidimensional** e otimizado para design de **interface de usuário.** Nesse modelo de grade, os **elementos-filhos** do **contêiner** que defina a grade podem ser posicionados livremente em espaços criados na estrutura, que ela tenha sido definida com suas **dimensões flexíveis** ou **fixas.**

**Propriedades do contêiner do grid layout:**

**Display:**

**Grid-template-columns: número de colunas;**

**Grid-template-row: numeor de linhas;**

**Gap: espaço entre os itens.**

**Align-items** > o valor normal é **strech:** mas os valores são start, center e end.

**Justify-items** > o valor strech, mas os valores são: start, center e end.

**Align-content** > o valor padrão é start, mas os valores são: strach, center, end, space-between, space-evenly e space-around.

**Justify-content** > o valor padrão é start, mas os valores são: center, end, space-between, space-evenly e space-around.

Align – é vertical

Justify – é vertical