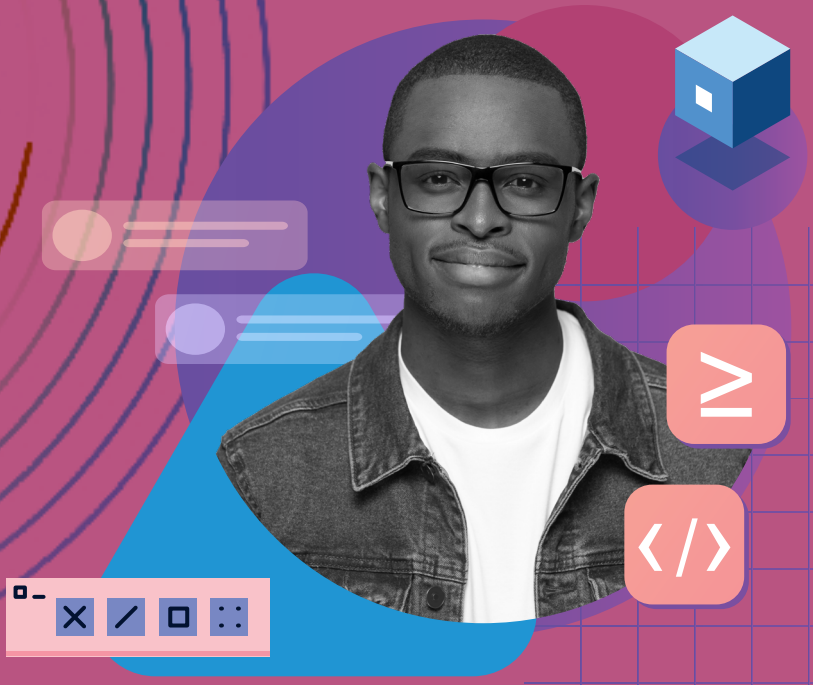
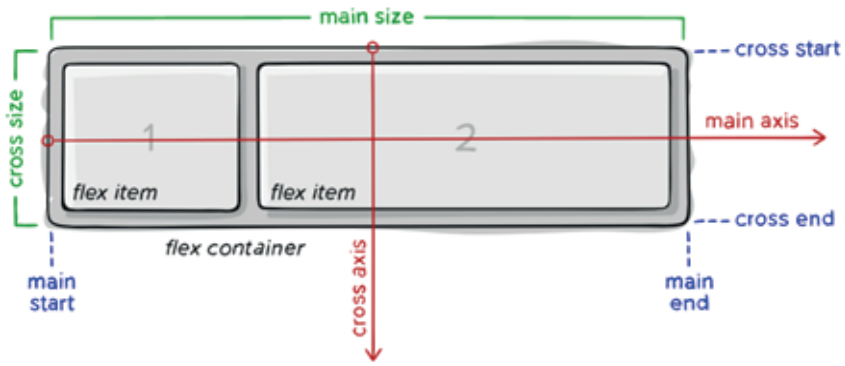
Google olabi

<Displays flexegrid>

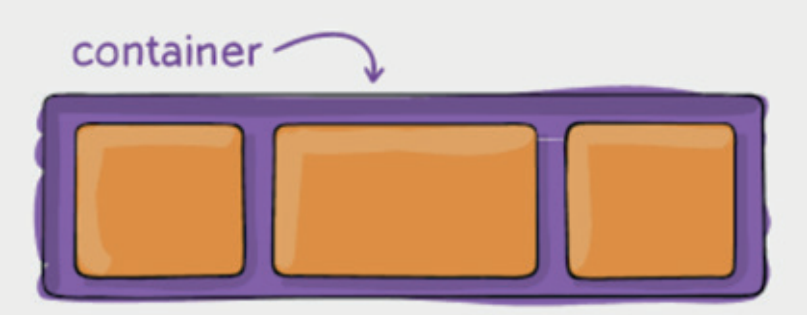


{ Flexbox

O *Flexbox* é mais eficiente para criação de *layouts*, alinhamento e distribuição de espaços entre itens em um container, mesmo quando as dimensões destes itens forem desconhecidas e/ou dinâmicas. Com o *flexbox*, organizamos elementos HTML dentro de um container (uma caixa em volta de vários itens).

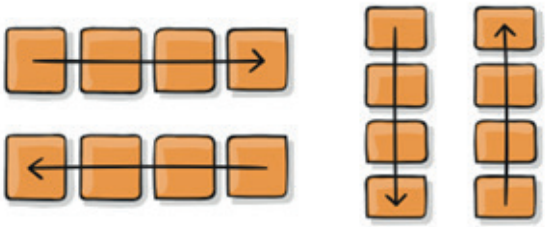


Pensando de forma lúdica, podemos imaginar o *flexbox* como uma caixa/ container na qual todos os itens precisam estar posicionados de acordo com o espaço dela. Com os atributos do *flexbox*, podemos ajustar os espaços onde cada item ficará posicionado.

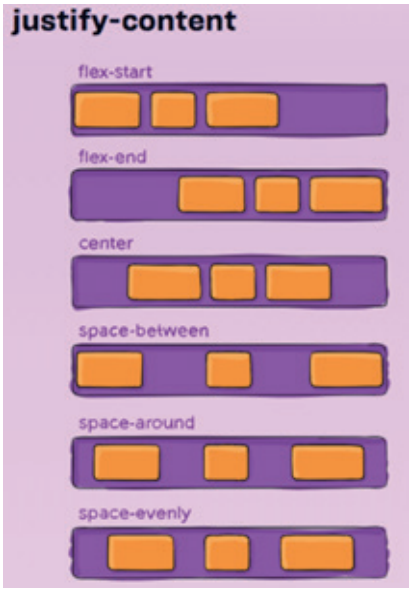


{ Disposição dos elementos

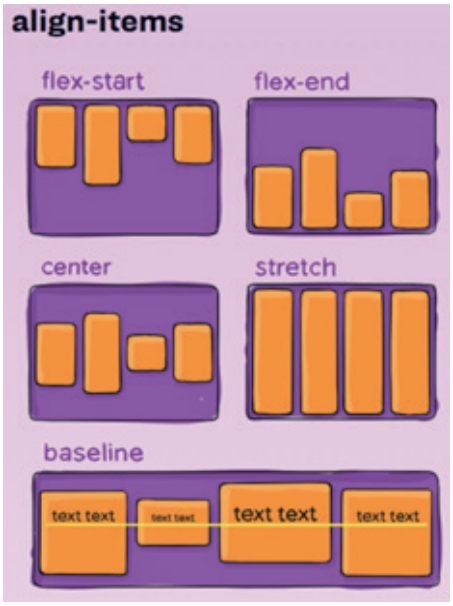
Existem algumas opções de definição de elementos dentro de um container:
1. flex-direction: define como os itens serão colocados no container flexível definindo o eixo principal e a direção (normal ou invertida);



2. justify-content: define o eixo principal e a direção na qual seus *flex-items* serão colocados, o que permite que você altere a ordem dos itens - que antes exigiam a alteração do HTML subjacente. Suas propriedades são *flex-start, flex-end, center, space-between, space-around, e space-evenly;*



3. align-items: define o valor de todos os filhos diretos, como um grupo dentro de um container. Suas propriedades são flex-start, flex-end, center, stretch e baseline;

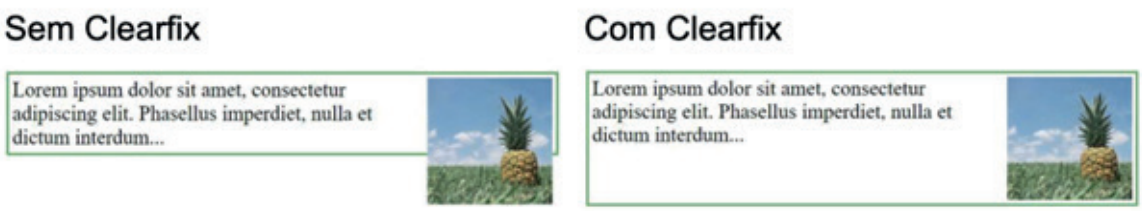


4. display: define a organização e o tipo de caixa a ser renderizada de forma organizada em uma página *web* - *inline* ou bloco, de acordo com o valor fornecido. O *display* fornece um contexto flexível para todos os seus filhos diretos ao trabalhar como uma tabela na qual o conteúdo pode ser colocado em linhas e colunas. Ao passarmos o *display*, o css entende que nos referimos ao *flexbox* e seus atributos;

5. float: determina a colocação de um item ao longo do lado direito ou esquerdo do container, onde textos e elementos em linha se posicionarem ao seu redor;

6. clear: Em conjunto com o *float*, usamos a propriedade *clear* quando queremos que o próximo elemento fique abaixo e não ao lado, como só com o *float*;

7. clearfix: Caso um dos elementos dentro do container seja mais alto que o outro, com a propriedade *clearfix*, é possível igualar o espaçamento entre eles.



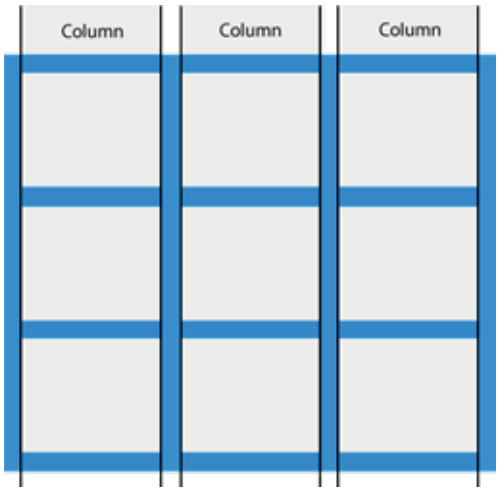
{ Grid

O *display grid* oferece um sistema de layout baseado em grade, com linhas e colunas que facilitam o design de páginas web sem necessidade de flutuações e posicionamento.

- display: grid;// Torna o elemento um *grid* container;
- display: inline-grid;// Torna o elemento um *grid* container, mas com comportamento *inline*;
- display: subgrid;// Para *grids* dentro de *grids*. Embora ainda não seja suportado, você pode colocar **display: grid**; no *grid* normalmente, dentro do que funciona.

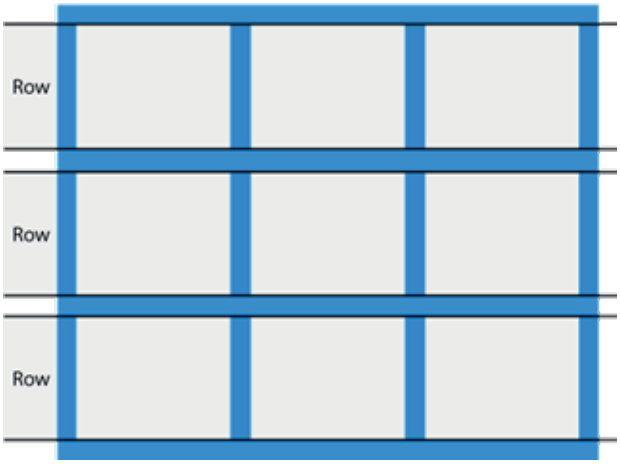
<Colunas de grade>

As linhas verticais dos itens de grade são chamadas de colunas.



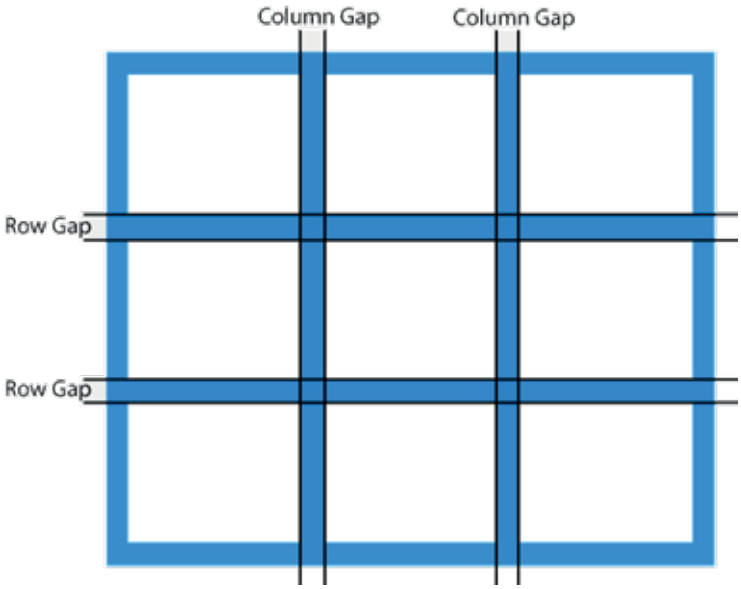
<Linhas de grade>

As linhas horizontais dos itens de grade são chamadas de linhas.



<Lacunas de grade>

Os espaços entre cada coluna/linha são chamados de lacunas.

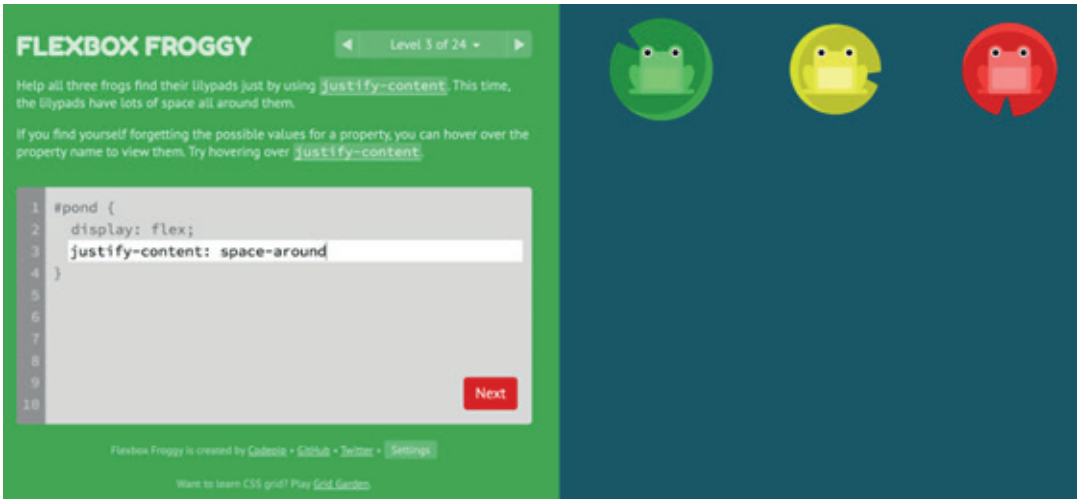


É possível ajustar o tamanho da lacuna usando uma das seguintes propriedades:
• *column-gap;*
• *row-gap;*
• *gap.*
Abaixo, as propriedades da grade CSS:

Propriedade	Descrição
column-gap	Especifica a lacuna entre as colunas
grid-area	Especifica um nome para o item da grade
grid-auto-columns	Especifica um tamanho de coluna por padrão
grid-auto-flow	Especifica como os elementos auto colocados serão inseridos na grade
grid-auto-rows	Especifica um tamanho de linha por padrão
grid-column-end	Especifica onde termina o item da grade
grid-column-gap	Especifica o tamanho da lacuna entre as colunas
grid-column-start	Especifica onde começa o item da grade
grid-row-end	Especifica onde termina o item da grade
grid-row-gap	Especifica o tamanho da lacuna entre as linhas
grid-row-start	Especifica onde começa o item da grade
row-gap	Especifica a lacuna entre as linhas da grade

<Exercícios>

1 O *flexbox froggy* é um site/jogo usado por desenvolvedores para colocarem em prática os atributos do *flexbox* permitindo que cada sapo encontre sua posição certa. Acesse o site e complete todos os desafios para concluir os 24 níveis.



2 Monte as estruturas abaixo usando HTML, CSS e largura máxima de 1100px.

a. **Primeira estrutura:**

b. **Segunda estrutura:**

c. **Terceira estrutura:**

<Referências>

Grade Flexbox (flexboxgrid.com)
CSS Grid Layout - Guia Completo - display: grid; (origamid.com)
Layout da grade CSS (w3schools.com)