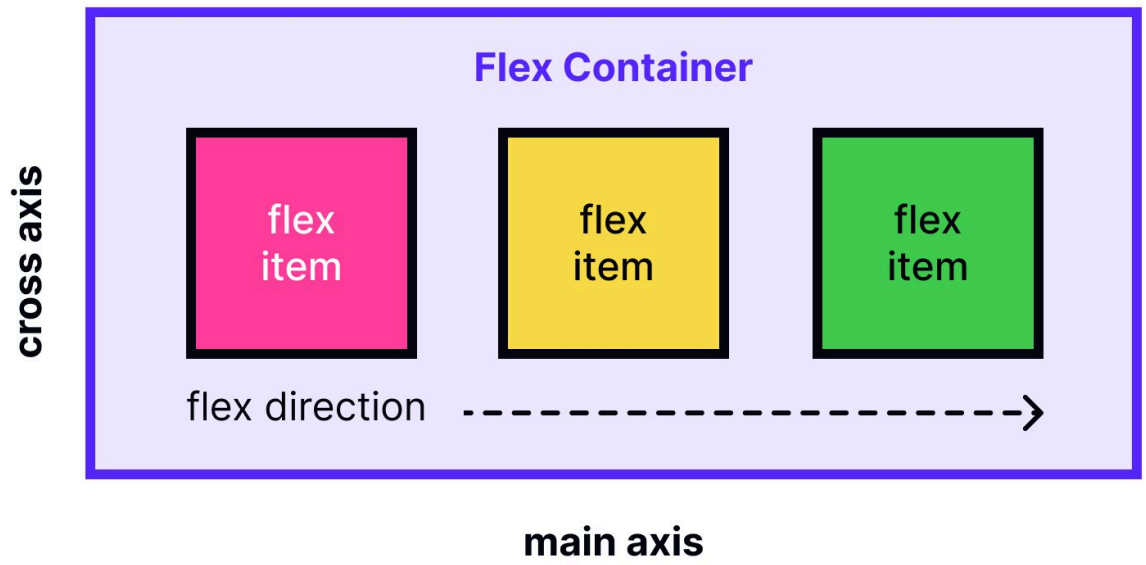
CSS Flexbox

Link Notion: https://cherry-client-b8f.notion.site/CSS-Flexbox-176911d84e0d8080830df4df84fd5ffd?pvs=73

Com a propriedade **display: flex**, nós conseguimos criar blocos flexíveis, ou seja, todos os elementos dentro deste bloco respeitarão as propriedades e regras do elemento pai. Dessa maneira conseguimos ajustar comportamentos de uma forma mais flexível e responsiva.

Por padrão, o flexbox vai alinhar os filhos do elemento lado a lado, na mesma linha e esticar na vertical, para que tenham a mesma altura.



Nós iremos dividir as propriedades entre o **Flex Container**(elemento pai) e o **Flex Item**(Elementos filho).

Flex Container

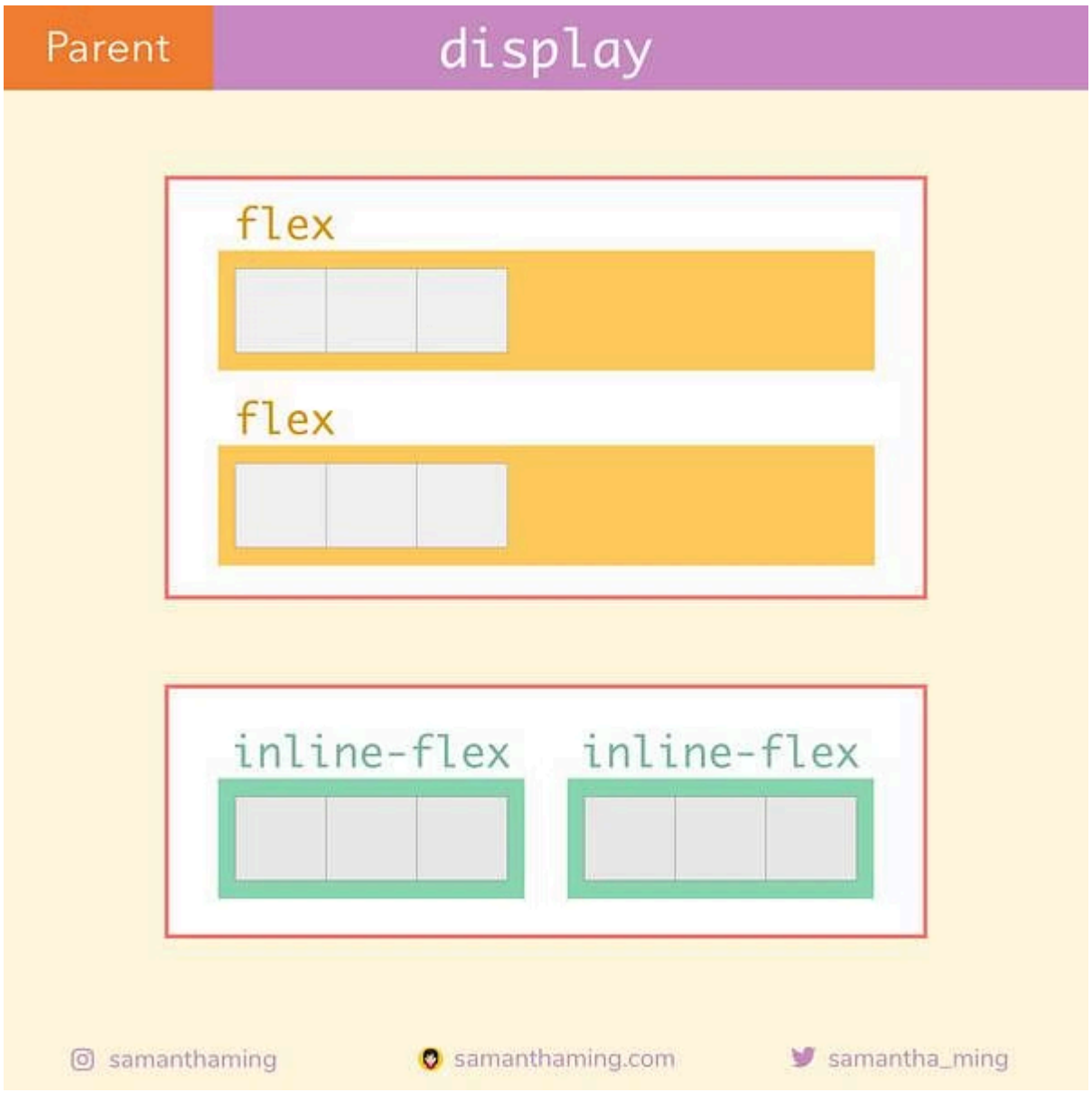
Abaixo estão as propriedades que você deverá definir no CSS do bloco que contém o display: flex.

▼ **display**

É a propriedade inicial, que define que um bloco irá se comportar como um bloco flexível. Sem ele, as outras propriedades não irão funcionar. Nele temos duas opções: **flex e inline-flex**.

```
.container{
      display: flex | inline-flex;
}
```

Enquanto com o flex, o bloco irá ocupar a **linha inteira**, assim como elementos com **display: block**, o inline-flex possibilitará que duas ou mais caixas com display: flex **ocupem a mesma linha**, muito parecido com o comportamento do **display: inline**.



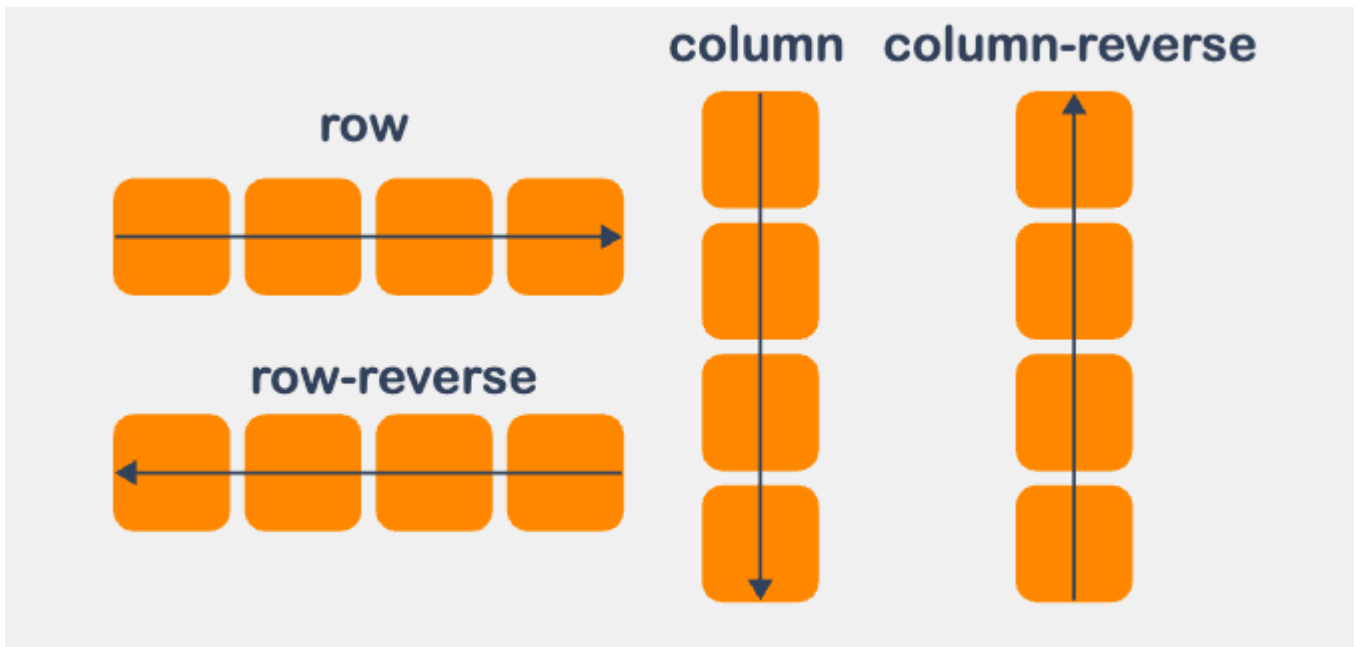
No vídeo abaixo você pode ver um pouco mais sobre a diferença de comportamento entre os dois.

https://www.youtube.com/watch?v=Xo3vyx2KSK8

▼ **flex-direction**

É a propriedade que define a direção dos elementos dentro de um bloco flexível. Normalmente usaremos para definir se queremos um ao lado do outro ou um embaixo do outro. Também temos a opção de definir se a ordem dos elementos será da esquerda para a direita ou da direita para a esquerda.

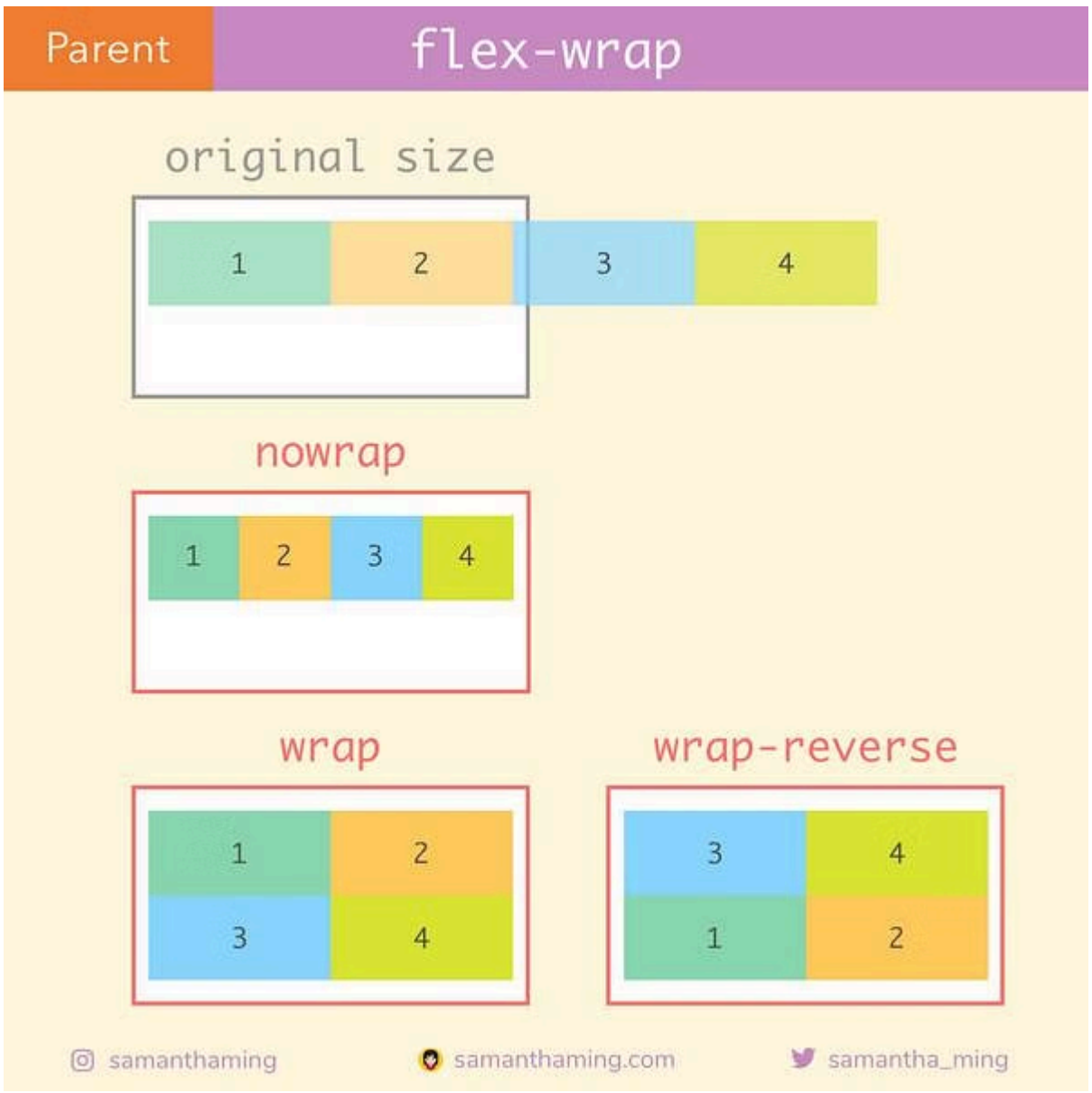
```
.container {
  flex-direction: row | row-reverse | column | column-reverse;
}
```



▼ flex-wrap

Por padrão, os elementos serão encaixados em uma única linha. Para que os itens mantenham seu tamanho e o bloco flexível quebre a linha, utilizamos a propriedade flex-wrap.

```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```



▼ flex-flow

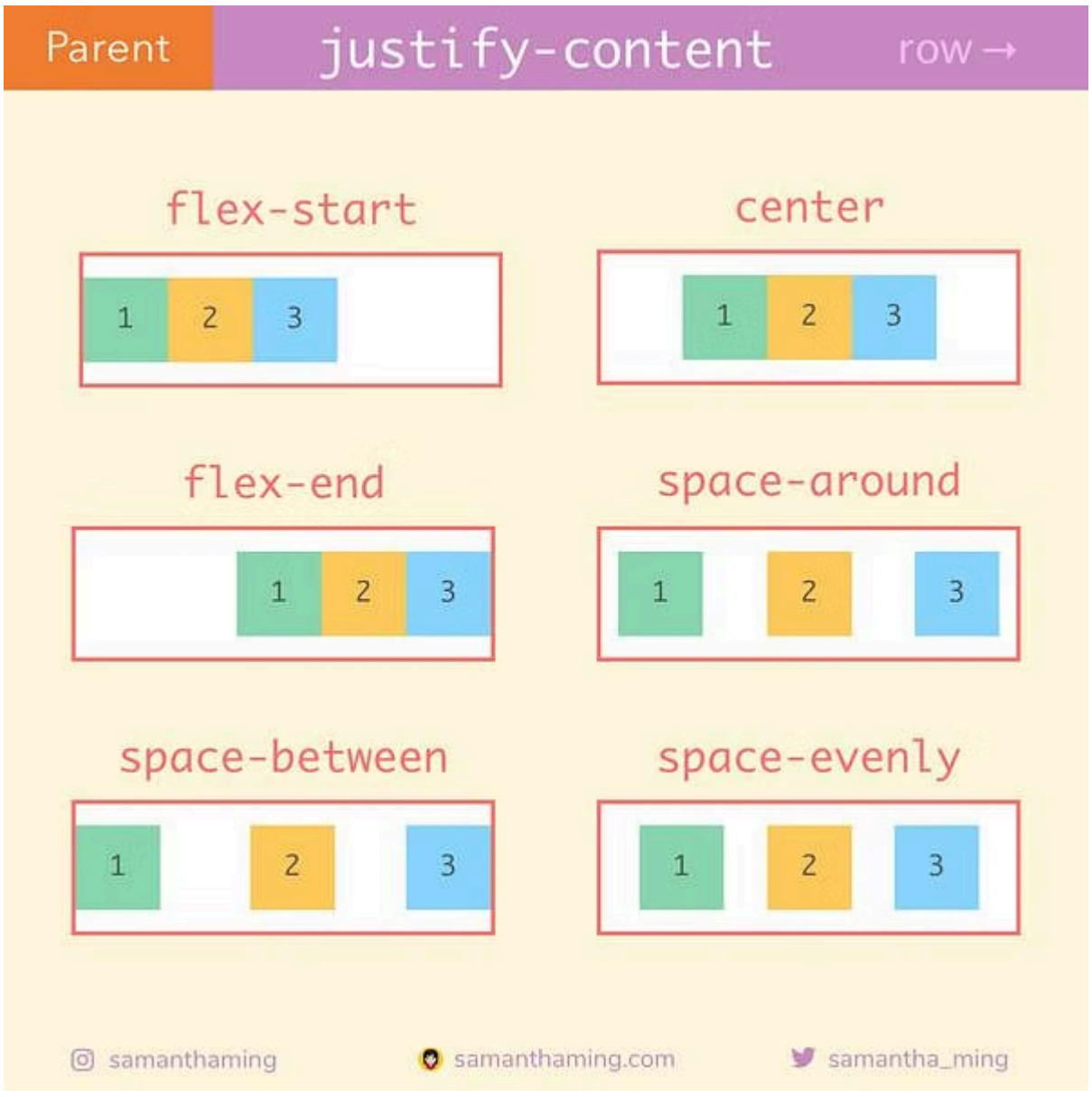
É uma forma reduzida de configurar a direção e a quebra de linha em uma única propriedade, onde a primeira propriedade configura a direção e a segunda a quebra de linha.

```
.container {
  flex-flow: column wrap;
}
```

▼ justify-content

Uma das propriedades que mais usaremos no dia-a-dia, ela determina como os elementos irão se alinhar na horizontal. **Atenção:** caso você altere a direção para column, você também está invertendo o eixo principal do bloco, portanto neste caso a propriedade justify-content irá controlar como os itens irão aparecer na vertical.

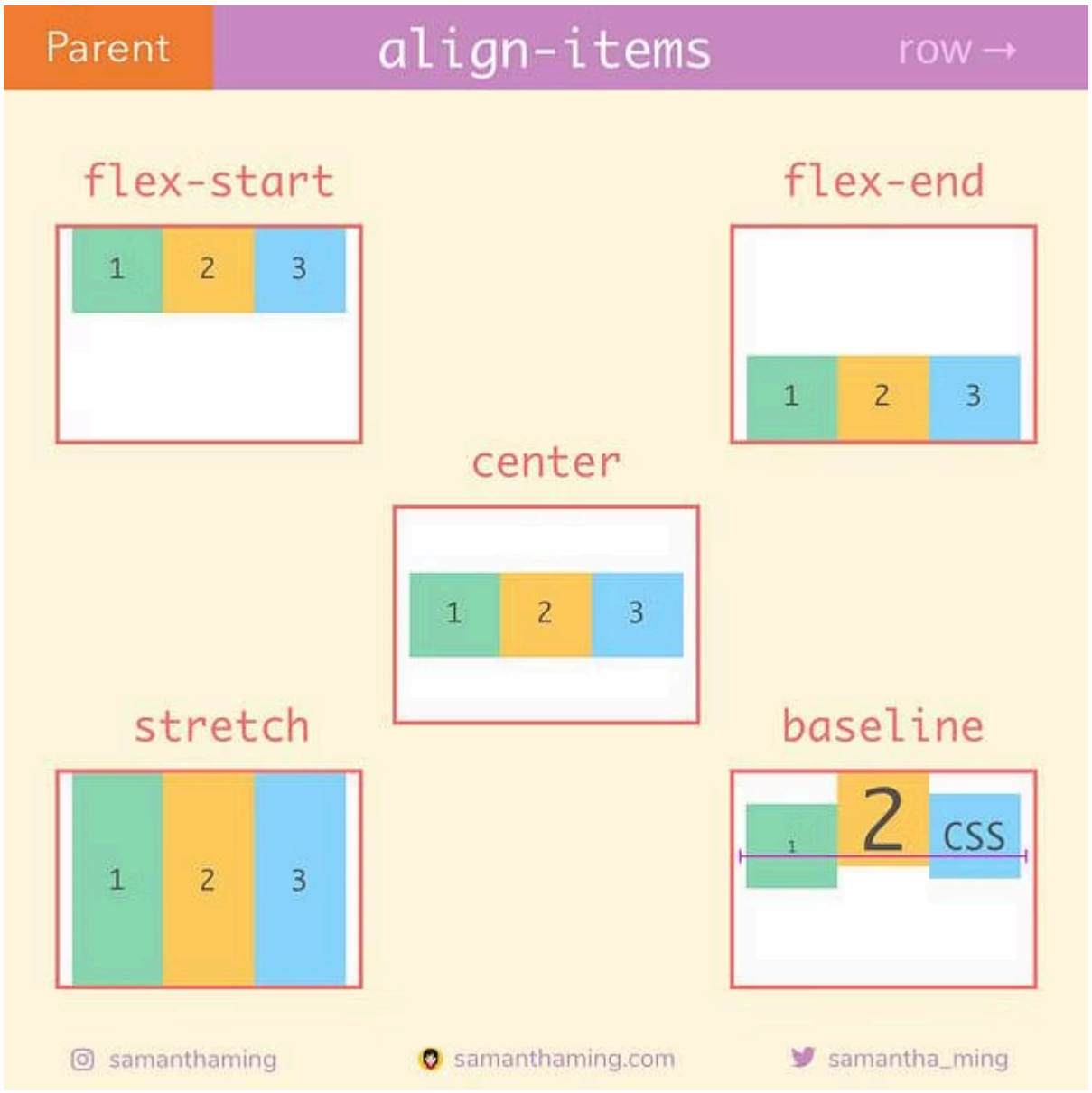
```
.container {
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | ri
ght ... + safe | unsafe;
}
```



▼ **align-items**

É a propriedade que por padrão controla o alinhamento dos elementos na vertical. Assim como o justify-content, caso a direção do bloco seja alterada para column, ela também é invertida e passa a controlar o alinhamento na horizontal.

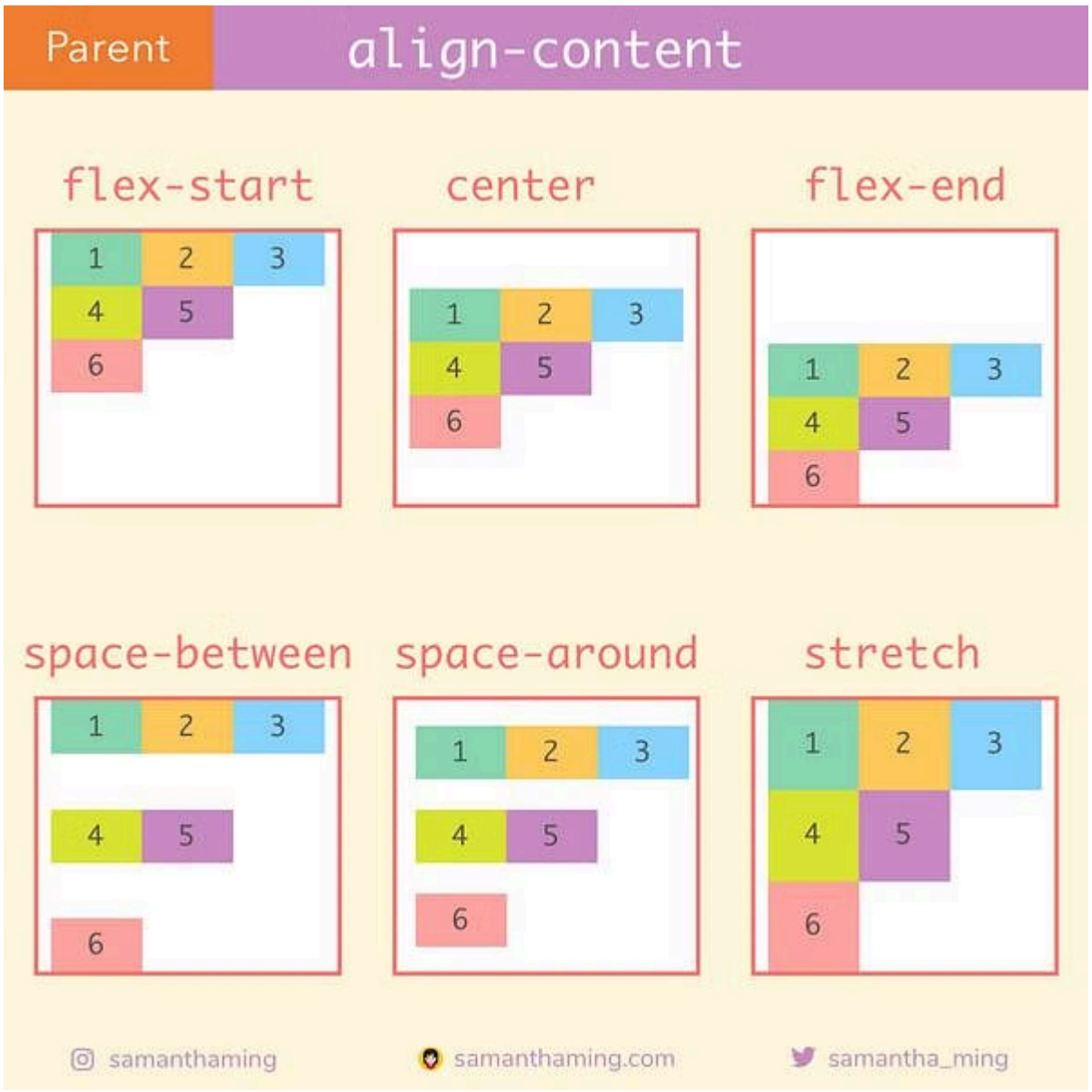
```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe;
}
```



▼ **align-content**

Essa propriedade só irá funcionar com a quebra de linha(wrap) habilitada. Ela determina como nós queremos que as linhas se comportem, parecido com o que as propriedades justify-content e align-items fazem com os elementos, mas ela é focada no comportamento das linhas, por isso só funciona quando temos diversas linhas dentro do nosso bloco flexível.

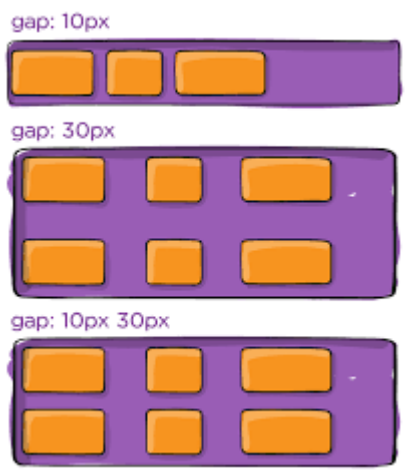
```
.container {
  align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;
}
```



▼ **gap**

Ele é responsável pelo espaçamento entre os elementos. Caso você esteja utilizando por exemplo, justify-content: space-between | space-around, ele se comportará determinando o espaço mínimo entre os elementos.

```
.container {
  display: flex;
  ...
  gap: 10px;
  gap: 10px 20px; /* row-gap column-gap */
  row-gap: 10px;
  column-gap: 20px;
}
```



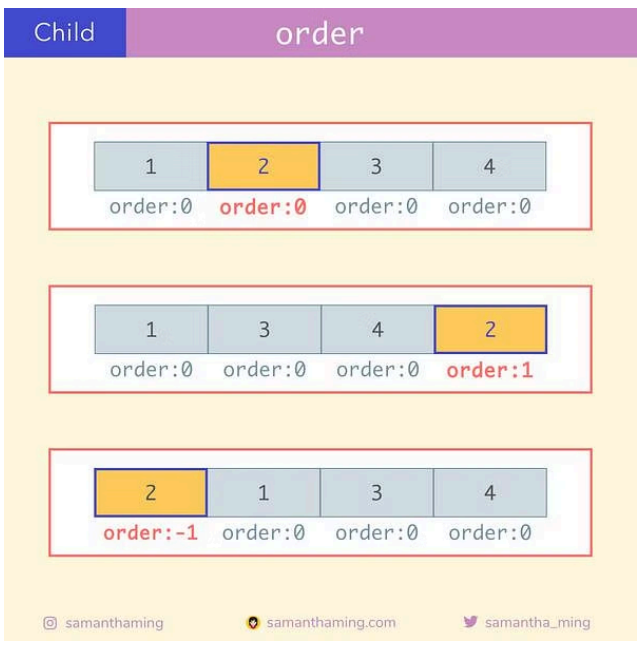
Flex Items

Abaixo estão as propriedades que você deve utilizar diretamente nos elementos de dentro de um elemento com display: flex, conhecidos como flex-items.

▼ order

Por padrão, a ordem dos elementos segue o que está no código, caso você queira mudar a ordem de um elemento em específico, você pode alterar através da propriedade order, que funciona como se fossem camadas, onde a camada 0 é a padrão. Ou seja, um item com order:1, virá depois de todos os outros elementos, já que o valor padrão é 0.

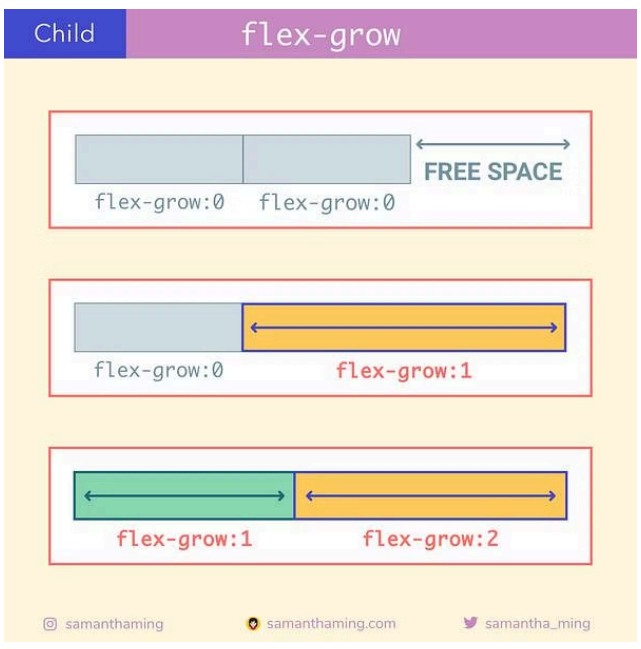
```
.item {
  order: 5; /* default is 0 */
}
```



▼ flex-grow

Ele define se um item em específico pode crescer caso tenha espaço sobrando no bloco em que ele está inserido. O valor serve como uma proporção, ou seja, quanto maior, maior será o espaço ocupado por aquele item em relação aos elementos que estão ao seu lado. O valor padrão dessa propriedade é 0.

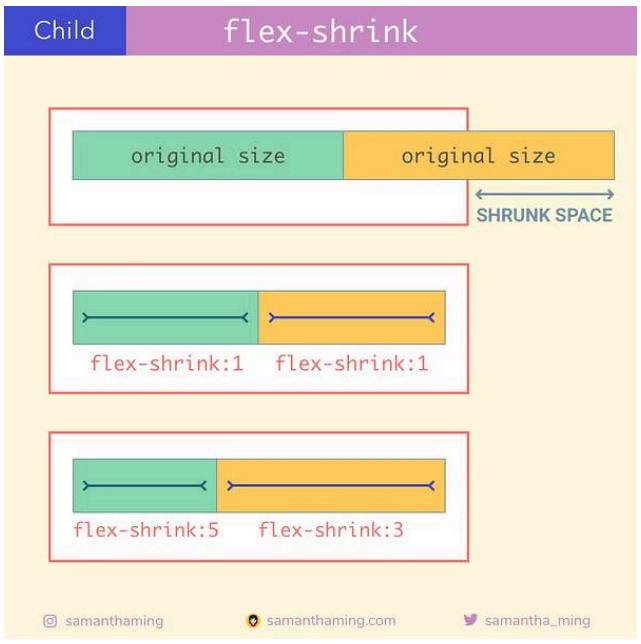
```
.item {
  flex-grow: 4; /* default 0 */
}
```



▼ flex-shrink

Ao contrário do flex-grow, o flex-shrink irá controlar quando o seu item pode ser espremido para caber na caixa. Quanto maior o número, mais ele será apertado. O valor padrão dessa propriedade é 1.

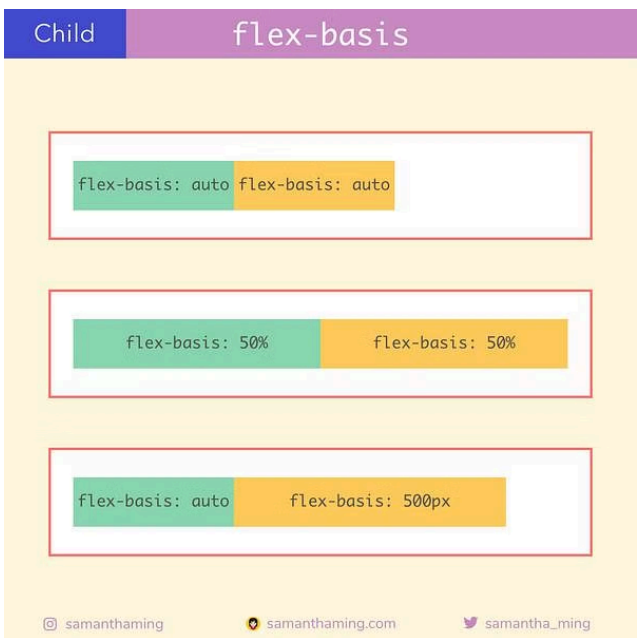
```
.item {
  flex-shrink: 3; /* default 1 */
}
```



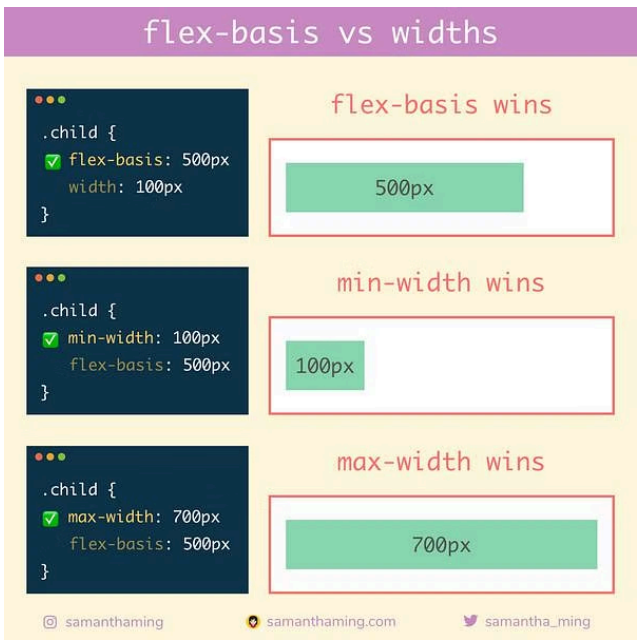
▼ flex-basis

Determina o tamanho inicial do elemento, que pode ser determinado por porcentagem, rem e outros. O valor padrão dessa propriedade é auto, ou seja, segue o tamanho do conteúdo.

```
.item {
  flex-basis:  | auto; /* default auto */
}
```



Existe uma hierarquia entre o flex-basis e o width que o navegador segue para entender o que ele deve seguir e está explicado em mais detalhes na imagem abaixo:



▼ flex

É a forma curta de escrever flex-grow, flex-shrink e flex-basis em uma única propriedade. Caso você não passe um dos 3 valores, será interpretado da seguinte maneira:

The flexible `flex` shorthand property

`flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]`

Keyword value

```
flex: initial;
→ flex: 0 1 auto;

flex: auto;
→ flex: 1 1 auto;

flex: none;
→ flex: 0 0 auto;
```

One-value syntax

```
flex: 1;
→ flex: 1 1 0;

flex: 10em;
→ flex: 1 1 10em;
```

Two-value syntax

```
flex: 1 30px;
→ flex: 1 1 30px;

flex: 2 2;
→ flex: 2 2 0;
```

Three-value syntax

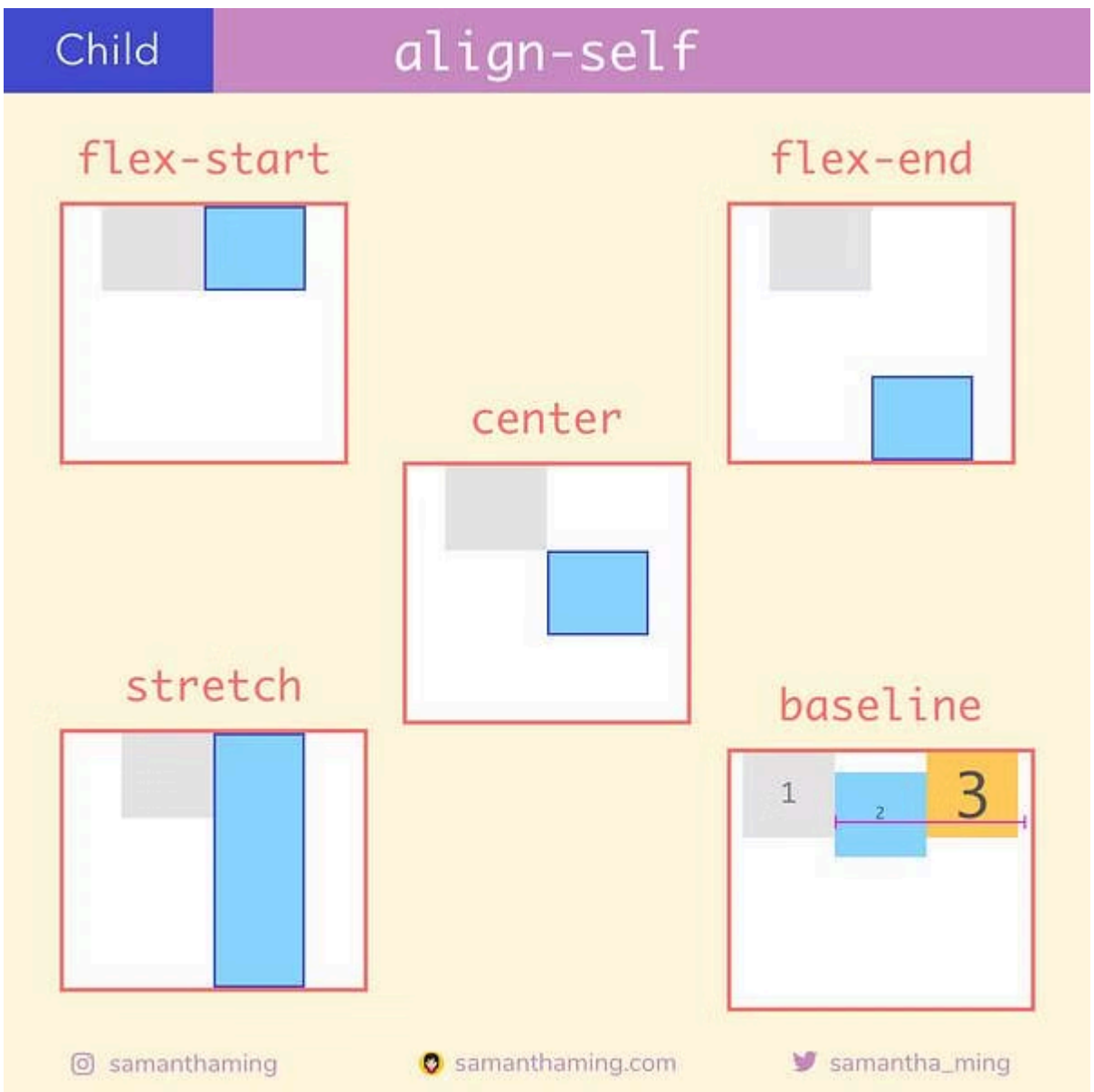
```
flex: 2 2 10%;
```

#devsheets by @stefanjudis

▼ align-self

Essa propriedade permite que a gente controle um elemento em específico com um alinhamento diferente de todo o restante no eixo que a propriedade align-items está controlando(por padrão na vertical). Ela aceita os mesmos valores do align-items.

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```



Referências

https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Test_your_skills/Flexbox

https://www.freecodecamp.org/news/learn-flexbox-build-5-layouts/

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://www.samanthaming.com/flexbox30/

https://medium.com/@lucasjs/como-o-flexbox-funciona-explicado-com-gifs-grandes-e-coloridos-26c42a0bcdc

https://www.youtube.com/watch?v=u044iM9xsWU&list=PLkCo96lupEP_ZYwV7W4CW2u_YaT1OK4no&index=31

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flexible_box_layout/Typical_use_cases_of_flexbox

https://www.freecodecamp.org/news/learn-flexbox-common-use-cases/