

# CSS

## Layouts flexíveis (*flexbox*)

---



# Conteúdo da aula

---

Nesta aula vamos ainda discutir sobre o posicionamento de elementos HTML em páginas Web, porém agora usando os layouts flexíveis (*Flexbox Layouts*).

# Introdução

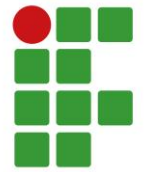
---



# Introdução

---

A forma de posicionar elementos em CSS evoluiu com os anos, atendendo a demandas por layouts mais flexíveis e simples de serem implementados, assim surgiram os módulos ***Flexbox*** e ***Grid***.



# Introdução

---

***Flexbox Layout Module*** e ***Grid Layout Module*** são sistemas de posicionamento de elementos para construção de layouts em páginas Web



# Introdução

---

Antes deles era necessário criar layouts usando as propriedades de ***display (inline e block)***, usando **tabelas (*table layouts*)** ou ainda usando **posicionamento explícito (propriedades *position*)**.



# Introdução

---

***Grid Layout Module*** em tradução livre **Modulo de layout de grade** é um sistema de posicionamento baseado em linhas e colunas (grade), que possibilita a construção de layouts sem uso de das propriedades *position* e *float*.



# Introdução

---

***Flexbox Layout Module*** em tradução livre Modulo de layout de caixa flexível (fica horrível! então vou usar **Modulo de layout *flexbox***) é um modelo de modelo de layout unidirecional.





INSTITUTO  
FEDERAL  
São Paulo

# Introdução

---

Qual utilizar? **Flexbox** ou **Grid**?



INSTITUTO  
FEDERAL  
São Paulo

# Introdução

---

Qual utilizar? **Flexbox** ou **Grid**?

**DEPENDE!**



# Introdução

---

## Quando usar *FlexBox*?

“quando estamos trabalhando com elementos em apenas uma dimensão, seja linha ou coluna.”

Por exemplo, quando criamos um menu (vertical ou horizontal)



Result



Edit in JSFiddle

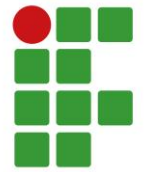
Home

Products

Contact

About

ITO  
AL  
lo



# Introdução

---

## Quando usar o *Grid*?

“quando estamos trabalhando com elementos em duas dimensões, principalmente para definir a estrutura de uma página, o que seria muito mais difícil com *FlexBox*”



Result



Edit in JSFiddle

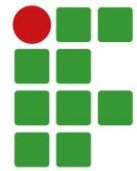
Header

Aside

Main

Footer

ITO  
AL  
lo



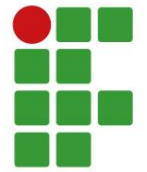
INSTITUTO  
FEDERAL  
São Paulo

# Introdução

---

**É possível utilizam ambos juntos**, o melhor dos  
dois mundos!





**INSTITUTO  
FEDERAL**  
São Paulo

# Introdução

---

Fonte:

<https://www.treinaweb.com.br/blog/flexbox-ou-css-grid>



# *Grid Layouts*

---



# *Grid Layout Module*

---

É o sistema de desenvolvimento de layouts mais recente, tendo seu início efetivo em meados de 2018. E sua recomendação efetiva pelo W3C em 2020.



# *Grid Layout Module*

---

De forma sucinta o *grid layout* é uma estrutura que permite o conteúdo ser empilhado verticalmente e horizontalmente de uma forma consistente e facilmente gerenciável.



# *Grid Layout Module*

---

No entanto, não vamos abordá-lo a fundo, visto que o tempo disponível é limitado, vamos nos dedicar mais ao Flex Layout Module.



# *Grid Layout Module*

---

Aos interessados fica a leitura recomendada de vários conteúdos sobre o Grid:

<https://tableless.com.br/entendendo-sistemas-de-grid-css-do-zero/>

<https://imasters.com.br/css/css-grid-de-um-jeito-facil-parte-01>

[https://www.w3schools.com/css/css\\_grid.asp](https://www.w3schools.com/css/css_grid.asp)



INSTITUTO  
FEDERAL  
São Paulo

# Grid Layout Module

Em especial esse link é o mais completo:

<https://css-tricks.com/snippets/css/complete-guide-grid/>



# *Flexbox Layouts*

---



# *Flexbox Layout Module*

---

Criado com o propósito de organizar os elementos em uma interface, além de possuir capacidades avançadas de alinhamento.

Seu uso torna mais fácil **projetar uma estrutura de layout responsiva e flexível** sem usar flutuação ou posicionamento.





# *Flexbox Layout Module*

---

É importante observar que quando se afirma que o *flexbox* é **unidimensional**, enfatiza-se o fato de que ele lida com o layout em uma dimensão de cada vez - seja uma linha ou uma coluna.



# *Flexbox Layout Module*

---

O *Flexible Box Layout Module* começou a ser pesquisado em 2009, 14 anos depois da primeira versão do CSS. No entanto sua primeira especificação oficial foi apresentada em 2014 sendo a especificação atual do ano 2018.



# *Flexbox Layout Module*

---

O nome completo da especificação é *Flexible Box Layout Module* porque ele é um módulo completo e não uma única propriedade.



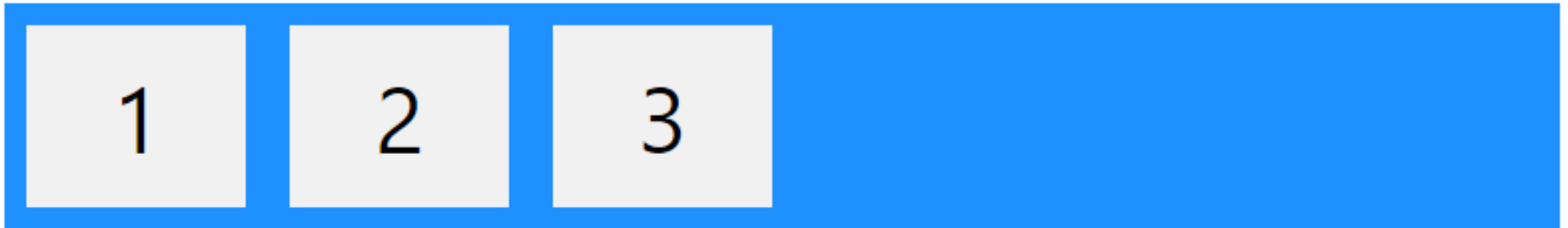
# *Flexbox Layout Module*

---

Seu funcionamento é baseado na declaração de algumas propriedades no container (o **elemento-pai**, que chamamos de ***flex container***), enquanto outras devem ser declaradas nos **elementos-filhos** (os ***flex itens***).

# *Flexbox Layout Module*

Exemplo,



O elemento acima representa um *flex container* (a área azul) com três elementos internos (filhos ou *flex items*).

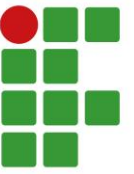


# *Flexbox Layout Module*

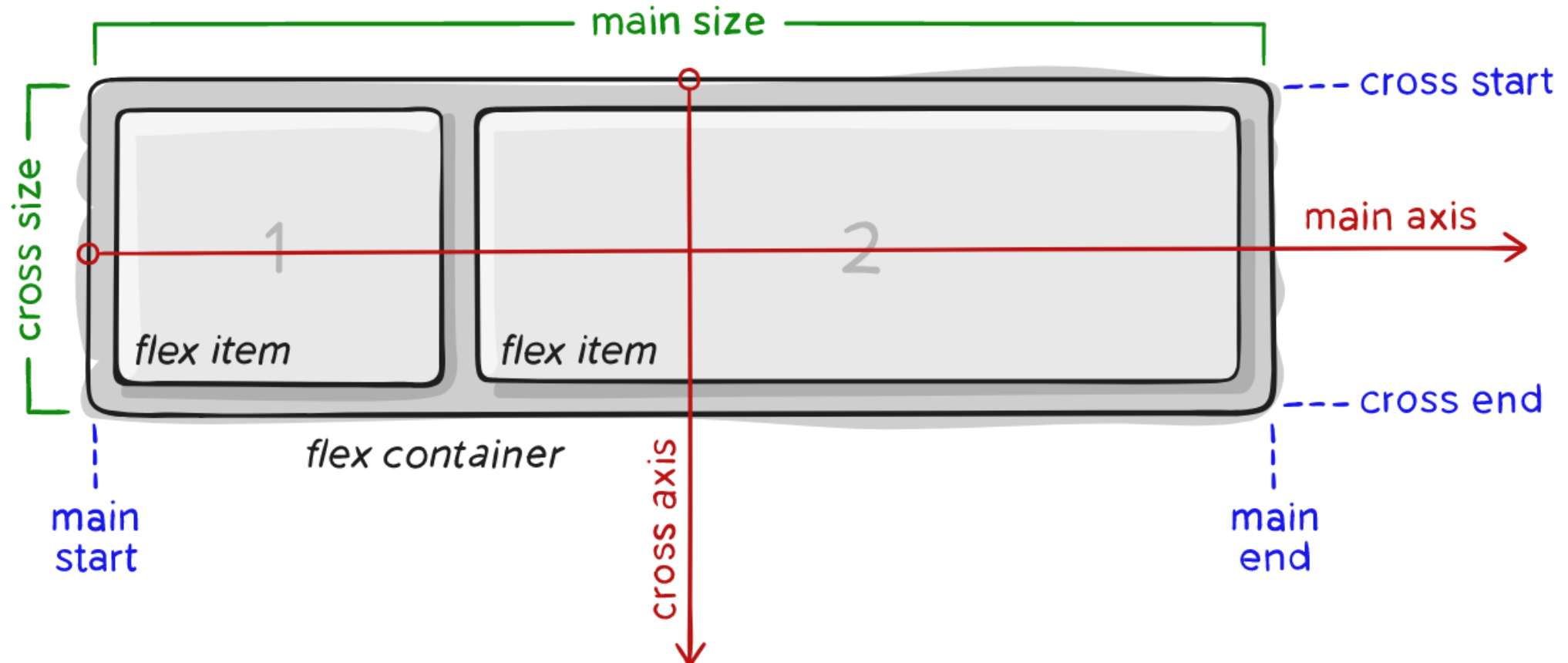
---

Se o layout "padrão" é baseado nas direções *block* e *inline*, o layout Flex é baseado em direções "*flex flow*".

O diagrama da especificação, tenta explicar a ideia central por trás do layout Flex.



# Flexbox Layout Module





# *Flexbox Layout Module*

---

Os itens serão dispostos no layout seguindo ou o eixo principal ou o transversal.

- Eixo principal (**main axis**): o eixo principal de um flex container é o eixo primário e ao longo dele são inseridos os flex items. Cuidado: O eixo principal não é necessariamente horizontal; vai depender da propriedade flex-direction.





# Flexbox Layout Module

---

- **main-start** | **main-end**: os flex items são inseridos dentro do container começando pelo lado start, indo em direção ao lado end.
- Tamanho principal (**main size**): A largura ou altura de um flex item, dependendo da direção do container, é o tamanho principal do item. A propriedade de tamanho principal de um flex item pode ser tanto width quanto height, dependendo de qual delas estiver na direção principal.



# *Flexbox Layout Module*

---

Eixo transversal (**cross axis**): O eixo perpendicular ao eixo principal é chamado de eixo transversal. Sua direção depende da direção do eixo principal.

➤ **cross-start** | **cross-end**: Linhas flex são preenchidas com itens e adicionadas ao container, começando pelo lado cross start do flex container em direção ao lado cross end.



# *Flexbox Layout Module*

---

➤ **cross size**: A largura ou altura de um flex item, dependendo do que estiver na dimensão transversal, é o cross size do item. A propriedade cross size pode ser tanto a largura quanto a altura do item, o que estiver na transversal.



# *Flexbox Layout Module*

---

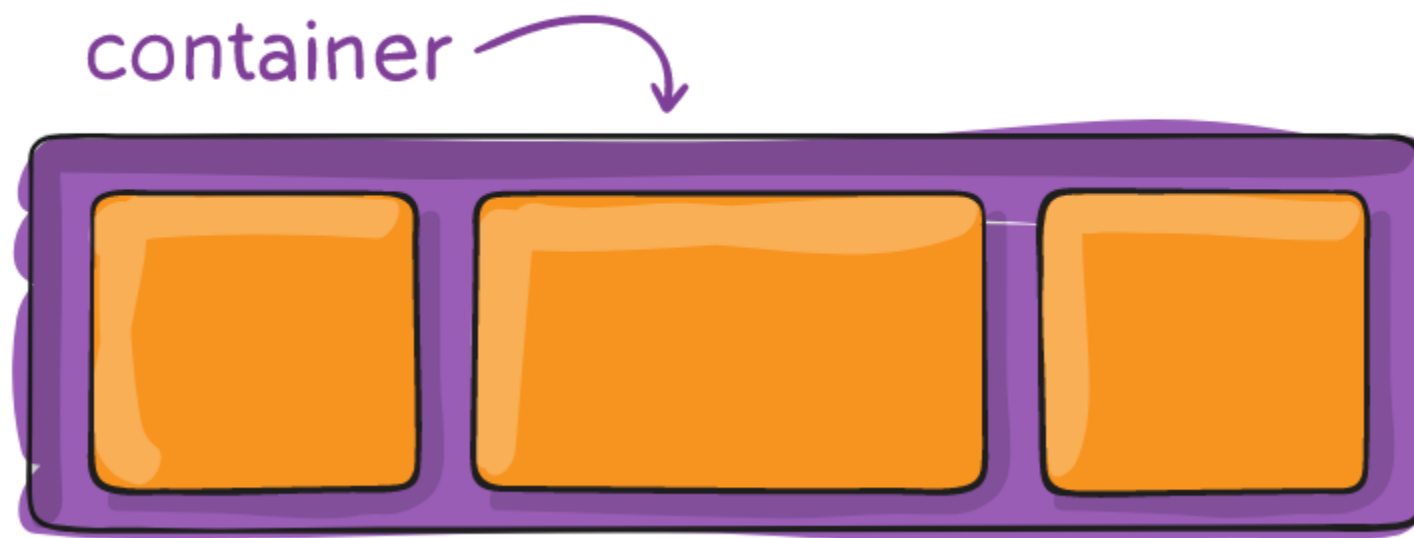
Propriedades para o elemento-pai:

- *display*
- *flex-direction*
- *flex-wrap*
- *flex-flow*
- *justify-content*
- *align-items*
- *align-content*



# *Flexbox Layout Module*

Propriedades para o **elemento-pai:**



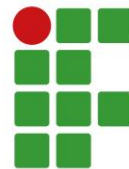


## *display: flex*

---

Esta propriedade define um *flex container*, colocando todos os elementos-filhos diretos num contexto Flex.

É a primeira e a mais importante de todas, pois possibilita o uso de todas as demais.



# Exemplo

Observe o seguinte exemplo:


[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox)

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>Create a Flex Container</h1>
```

## Create a Flex Container



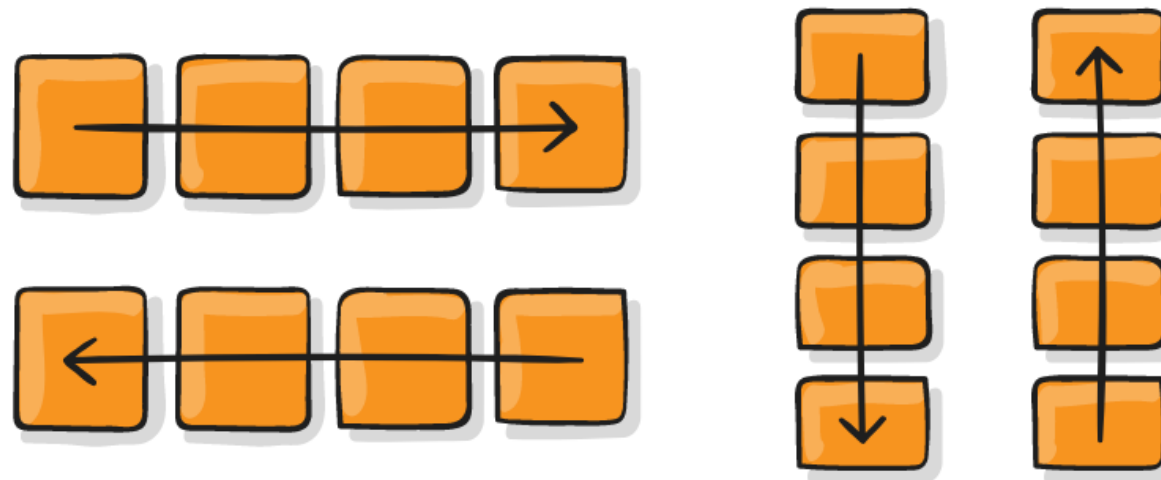
A Flexible Layout must have a parent element with the *display* property set to *flex*.

Direct child elements(s) of the flexible container automatically becomes flexible items.



# *flex-direction*

Estabelece o eixo principal, definindo assim a direção em que os *flex items* são alinhados no *flex container*.





# *flex-direction*

---

As opções são:

- *row* (padrão): esquerda para a direita
- *row-reverse*: direita para a esquerda
- *column*: de cima para baixo
- *column-reverse*: de baixo para cima



# Exemplo

[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox\\_flex-direction\\_column](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-direction_column)

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  flex-direction: column;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>The flex-direction Property</h1>
```

## The flex-direction Property

The "flex-direction: column;" stacks the flex items vertically (from top to bottom):





## *flex-wrap*

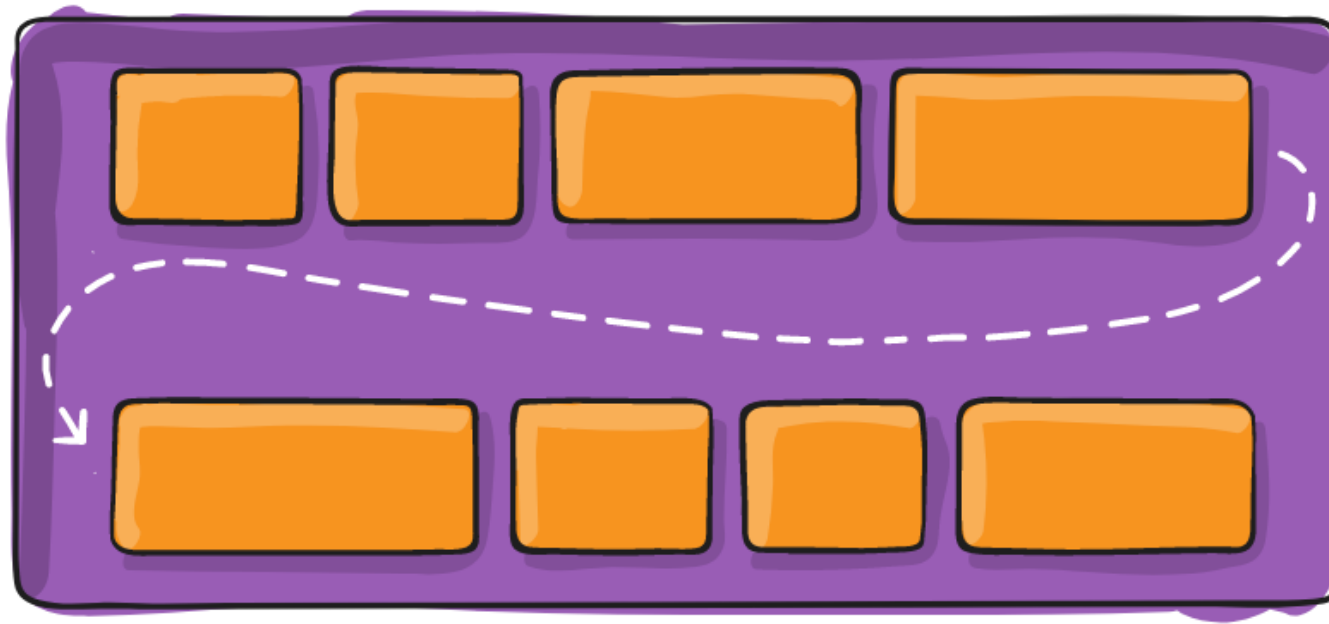
---

Por padrão, os flex items vão todos tentar se encaixar em uma só linha. Com esta propriedade você pode modificar esse comportamento e permitir que os itens quebrem para uma linha seguinte conforme for necessário.



# *flex-wrap*

Graficamente...





# *flex-wrap*

---

As opções são:

- *nowrap* (padrão): todos os flex items ficarão em uma só linha
- *wrap*: os flex items vão quebrar em múltiplas linhas, de cima para baixo
- *wrap-reverse*: os flex items vão quebrar em múltiplas linhas de baixo para cima



# Exemplo

[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox\\_flex-wrap\\_wrap](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-wrap_wrap)

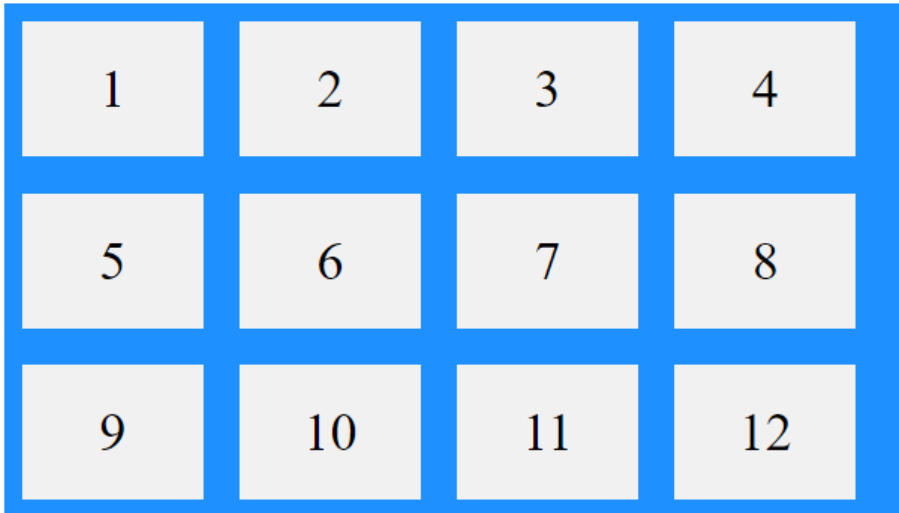
```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>The flex-wrap Property</h1>
```

## The flex-wrap Property

The "flex-wrap: wrap;" specifies that the flex items will wrap if necessary:



Try resizing the browser window.



# *flex-flow*

---

As propriedades ***flex-direction*** e ***flex-wrap*** são usadas tão frequentemente juntas que uma propriedade abreviada ***flex-flow*** foi criada para combiná-las.

Essa propriedade aceita o valor das duas propriedades separados por um espaço.



# Flex-flow - Exemplo

[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox\\_flex-flow\\_row\\_wrap](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-flow_row_wrap)

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  flex-flow: row wrap;
  background-color: DodgerBlue;
}
```

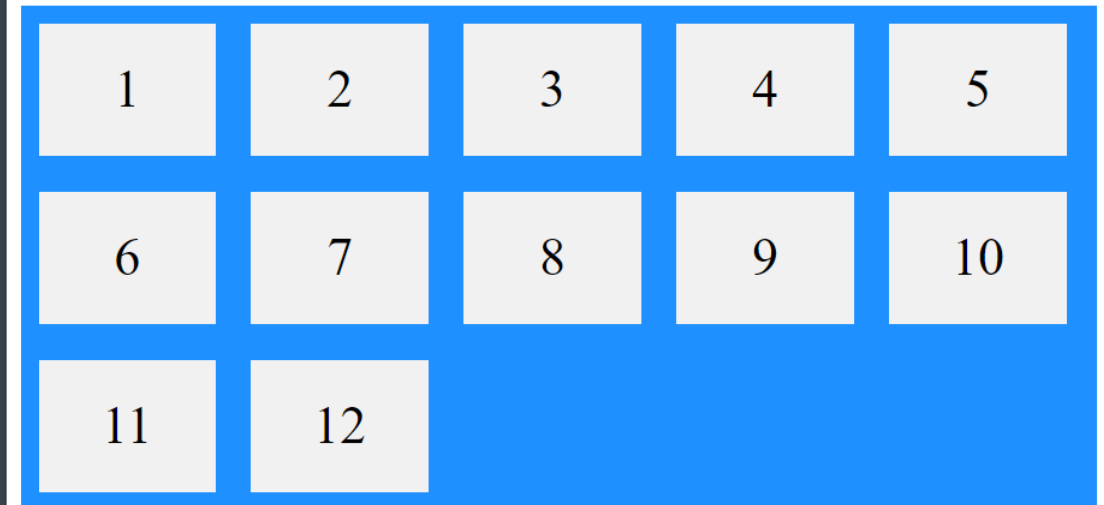
```
.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

```
</style>
</head>
<body>
<h1>The flex-flow Property</h1>
```

```
<p>The flex-flow property is a shorthand property for the flex-direction
and the flex-wrap properties.</p>
```

## The flex-flow Property

The flex-flow property is a shorthand property for the flex-direction and the flex-wrap properties.



Try resizing the browser window.

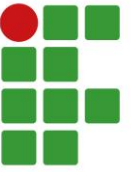




## *justify-content*

---

Esta propriedade define o **alinhamento dos itens ao longo do eixo principal**. Ajuda a distribuir o espaço livre que sobrar no container tanto se todos os *flex items* em uma linha são inflexíveis, ou são flexíveis mas já atingiram seu tamanho máximo. Também exerce algum controle sobre o alinhamento de itens quando eles ultrapassam o limite da linha.



# *justify-content*

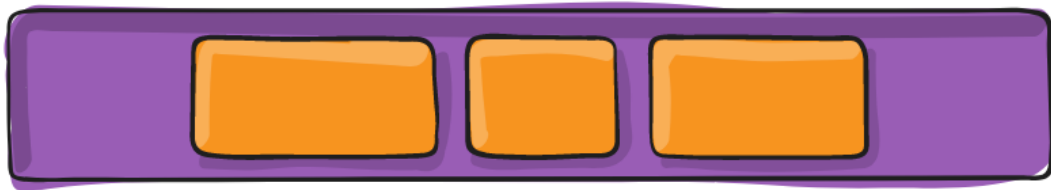
flex-start



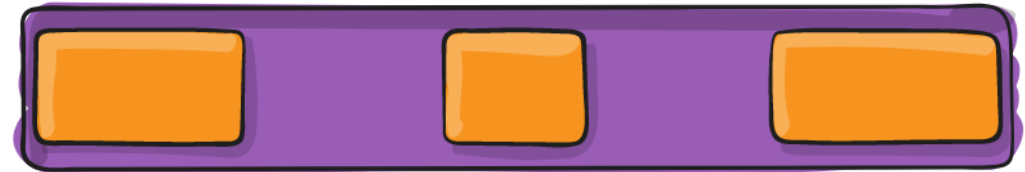
flex-end



center



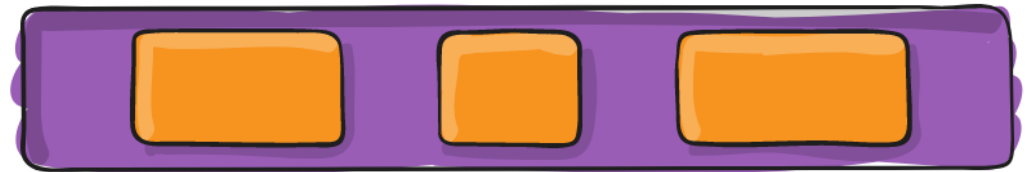
space-between



space-around



space-evenly



# *justify-content*

---

As opções são:

- *flex-start* (padrão): os itens são alinhados junto à borda de início (start) de acordo com qual for a *flex-direction* do container.
- *flex-end*: os itens são alinhados junto à borda final (end) de acordo com qual for a *flex-direction* do container.



## *justify-content*

---

- center: os itens são centralizados na linha.
- space-between: os itens são distribuídos de forma igual ao longo da linha; o primeiro item junto à borda inicial da linha, o último junto à borda final da linha.



## *justify-content*

---

- space-around: os itens são distribuídos na linha com o mesmo espaçamento entre eles. Note que, visualmente, o espaço pode não ser igual, uma vez que todos os itens tem a mesma quantidade de espaço dos dois lados: o primeiro item vai ter somente uma unidade de espaço junto à borda do container, mas duas unidades de espaço entre ele e o próximo item, pois o próximo item também tem seu próprio espaçamento que está sendo aplicado.



## *justify-content*

---

- space-evenly: os itens são distribuídos de forma que o espaçamento entre quaisquer dois itens da linha (incluindo entre os itens e as bordas) seja igual.



# Exemplo

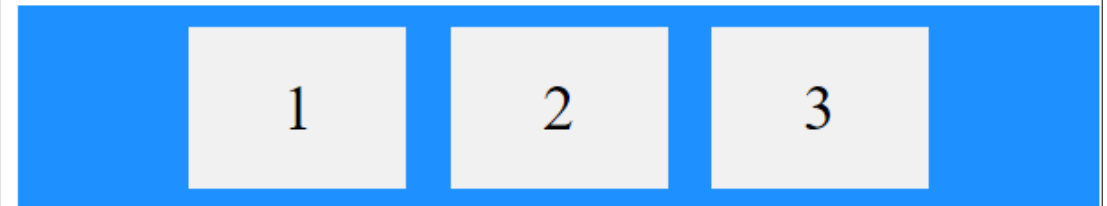
[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox\\_justify-content\\_center](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_justify-content_center)

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  justify-content: center;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

## The justify-content Property

The "justify-content: center;" aligns the flex items at the center of the container:





## *align-items*

---

Define o comportamento padrão de como flex items são alinhados de acordo com o eixo transversal (cross axis).

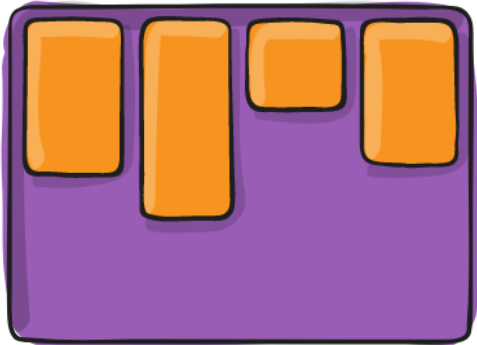
De certa forma, funciona de forma similar ao justify-content, porém no eixo transversal (perpendicular ao eixo principal).



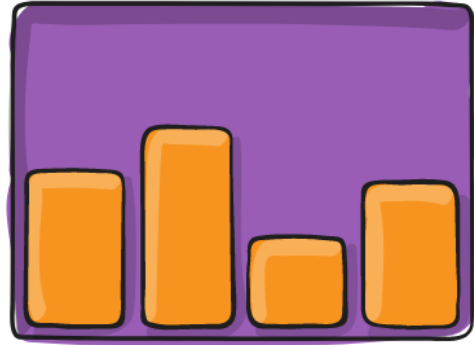


# *align-items*

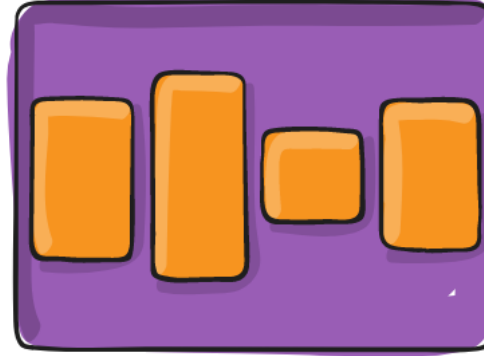
flex-start



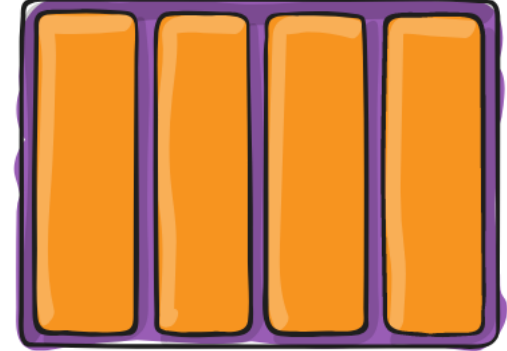
flex-end



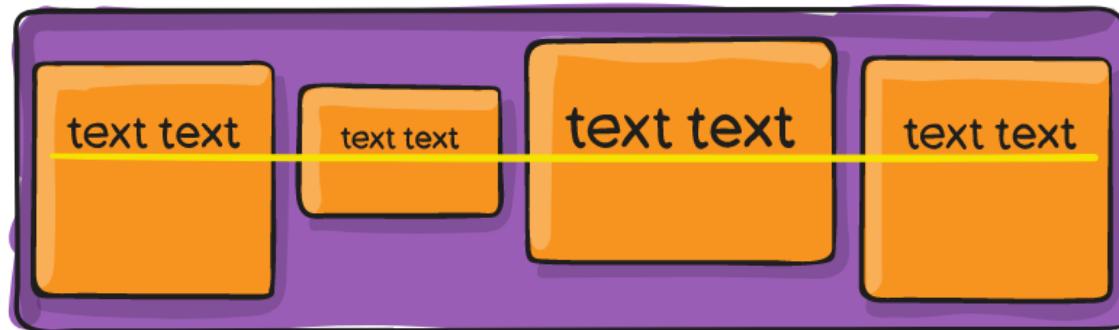
center

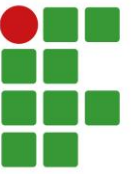


stretch



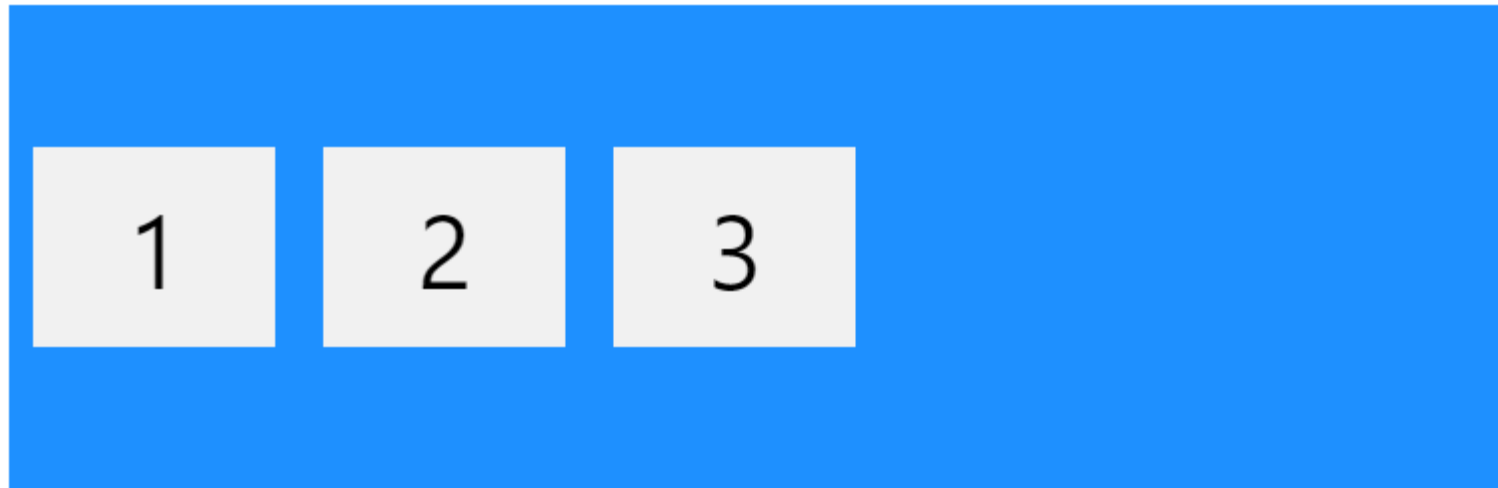
baseline





## *align-items*

Na prática ele alinha os itens no eixo transversal **somente se o elemento pai for maior**, veja o exemplo:





# *align-items*

---

As opções são:

- *stretch* (padrão): estica os itens para preencher o container, respeitando o min-width/max-width).
- *flex-start*: itens são posicionados no início do eixo transversal.
- *flex-end*: itens são posicionados no final do eixo transversal.



# *align-items*

---

As opções são:

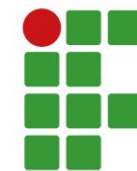
- *center*: itens são centralizados no eixo transversal.
- *baseline*: itens são alinhados de acordo com suas baselines.



## *align-content*

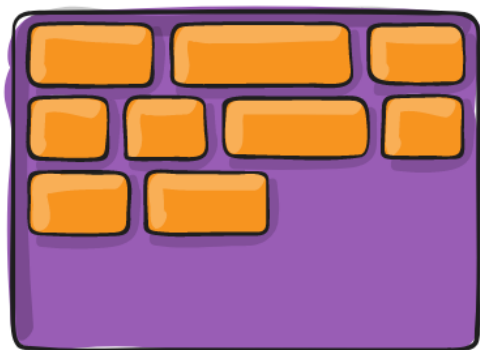
---

Organiza as linhas dentro de um flex container quando há espaço extra no **eixo transversal**, similar ao modo como *justify-content* alinha itens individuais dentro do **eixo principal**.

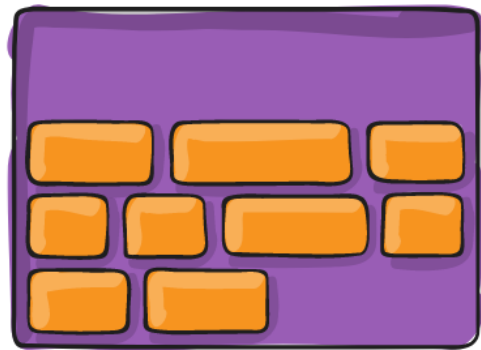


# *align-content*

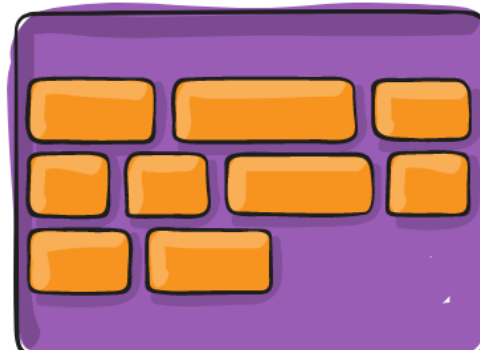
flex-start



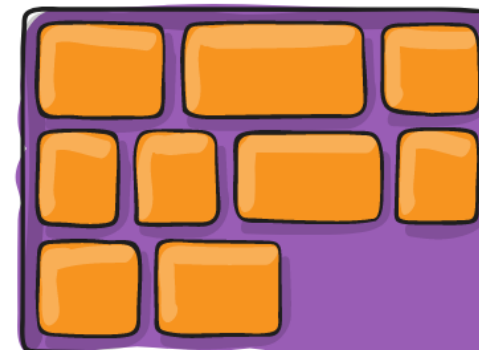
flex-end



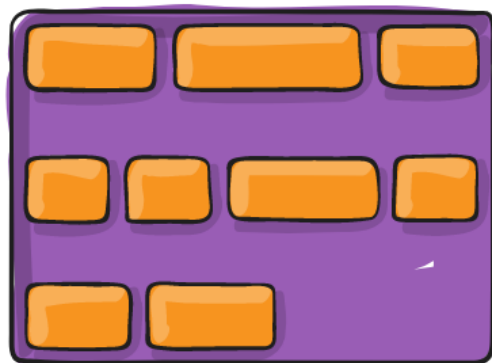
center



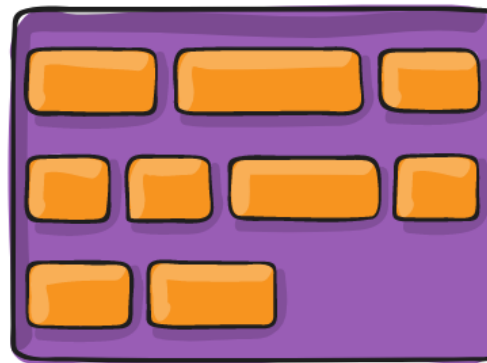
stretch



space-between



space-around





# *align-content*

---

As opções são:

- *flex-start*: itens alinhados com o início do container.
- *flex-end*: itens alinhados com o final do container.
- *center*: itens centralizados no container.
- *stretch* (padrão): itens em cada linha esticam para ocupar o espaço remanescente entre elas.

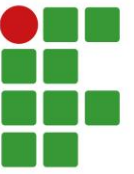


# *align-content*

---

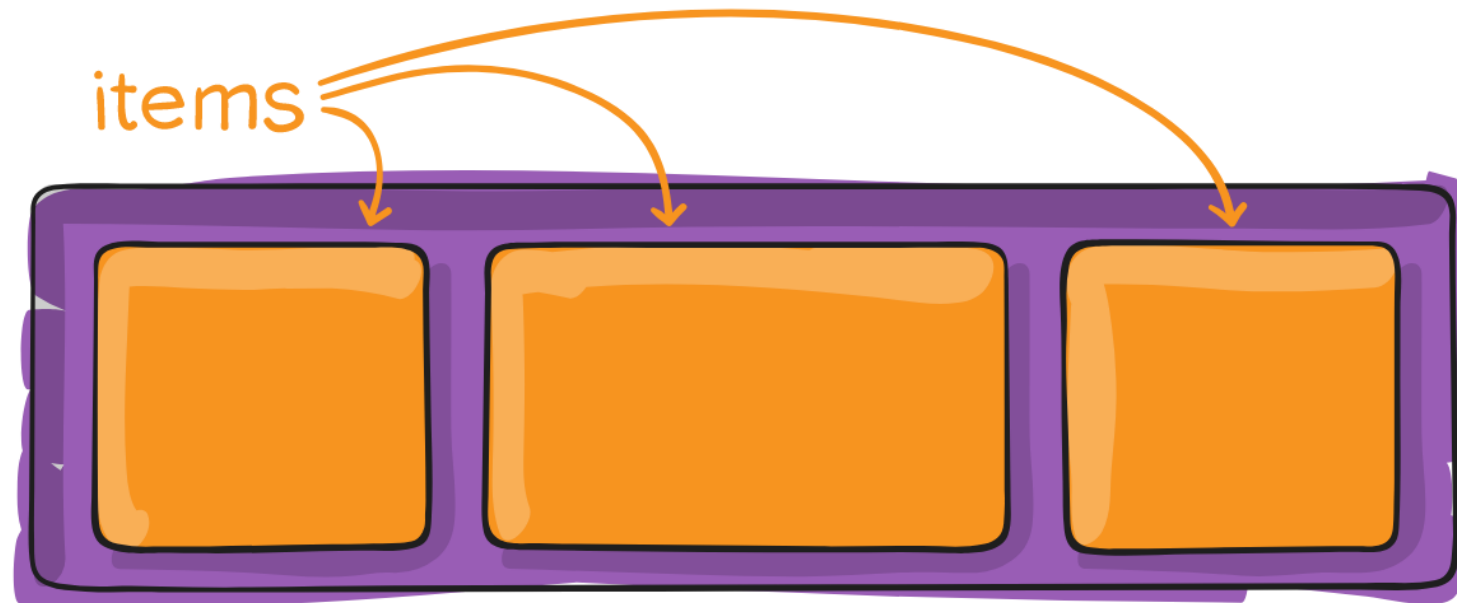
- *space-between*: itens distribuídos igualmente; a primeira linha junto ao início do container e a última linha junto ao final do container.
- *space-around*: itens distribuídos igualmente com o mesmo espaçamento entre cada linha.
- *space-evenly*: itens distribuídos igualmente com o mesmo espaçamento entre eles.





# *Flexbox Layout Module*

Propriedades para o **elementos-filhos:**





# *order*

---

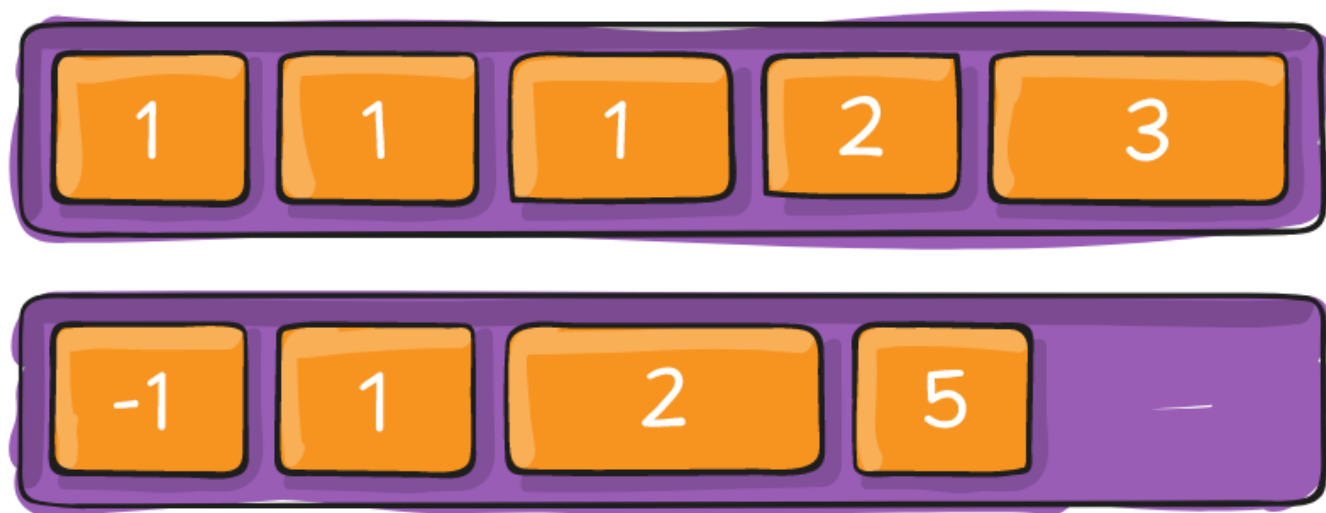
Determina a ordem em que os elementos aparecerão.

Por padrão os flex items são dispostos na tela na ordem do código. Mas a propriedade `order` controla a ordem em que aparecerão no container.

*order*



INSTITUTO  
FEDERAL  
São Paulo





## *flex-grow*

---

Define a habilidade de um **flex item de crescer**, caso necessário. O valor dessa propriedade é um valor numérico sem indicação de unidade, que serve para cálculo de proporção. Este valor dita a quantidade de espaço disponível no container que será ocupado pelo item.

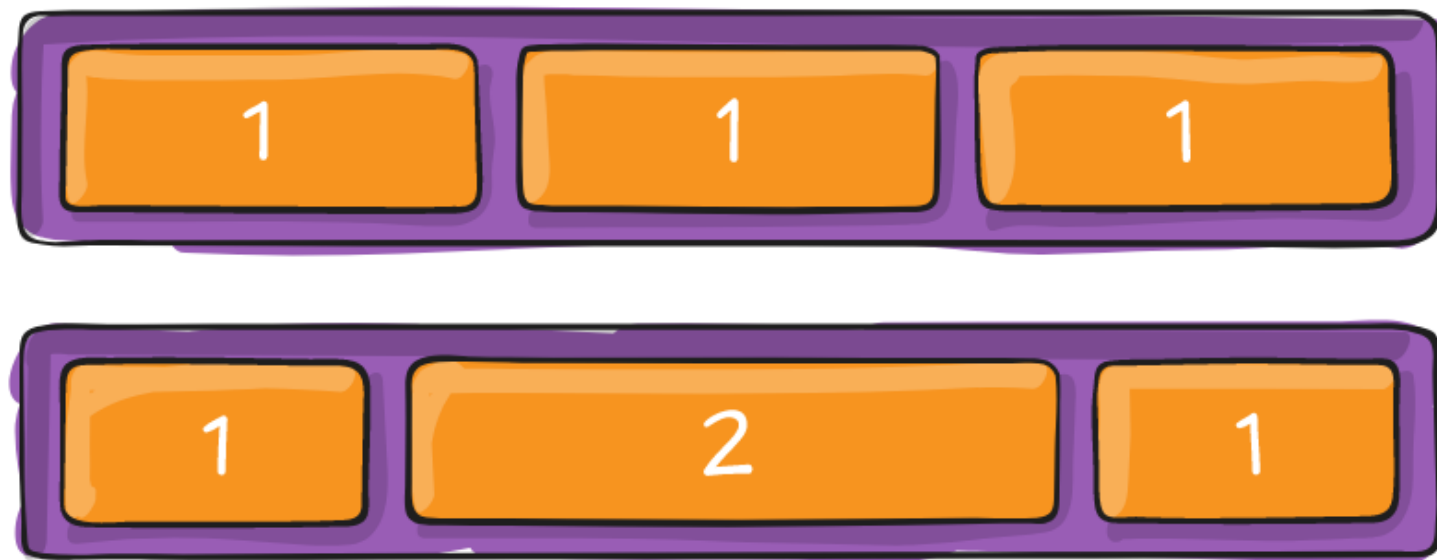


## *flex-grow*

---

Se todos os itens tiverem flex-grow definido em 1, o espaço remanescente no container será distribuído de forma igual entre todos. Se um dos itens tem o valor de 2, vai ocupar o dobro de espaço no container com relação aos outros (ou pelo menos vai tentar fazer isso).

# *flex-grow*





# Exemplo

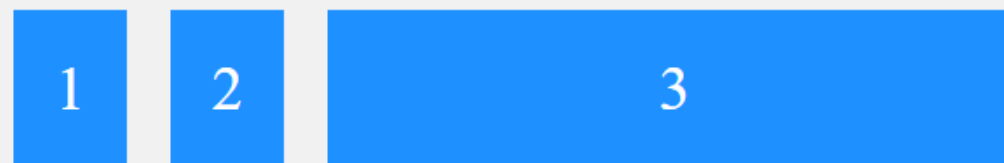
[https://www.w3schools.com/css/tryit.asp?filename=trycss3\\_flexbox\\_flex-grow](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-grow)

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}

.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
```

## The flex-grow Property

Make the third flex item grow eight times faster than the other flex items:





## *align-self*

---

Permite que o alinhamento padrão (ou o que estiver definido por align-items) seja sobrescrito para itens individuais.

Possui os mesmos valores da propriedade *align-items*.





# *align-self*

---

As opções são:

- *stretch* (padrão): estica os itens para preencher o container, respeitando o min-width/max-width).
- *flex-start*: itens são posicionados no início do eixo transversal.
- *flex-end*: itens são posicionados no final do eixo transversal.



# *align-self*

---

As opções são:

- *center*: itens são centralizados no eixo transversal.
- *baseline*: itens são alinhados de acordo com suas baselines.



# *Flexbox Layout Module*

---

Últimos conceitos importantes:

- Para que as propriedades funcionem nos elementos-filhos, as pais devem ter propriedade `display: flex`.
- As propriedades ***float, clear e vertical-align*** não têm efeito em *flex-items*.



# *Flexbox Layout Module*

---

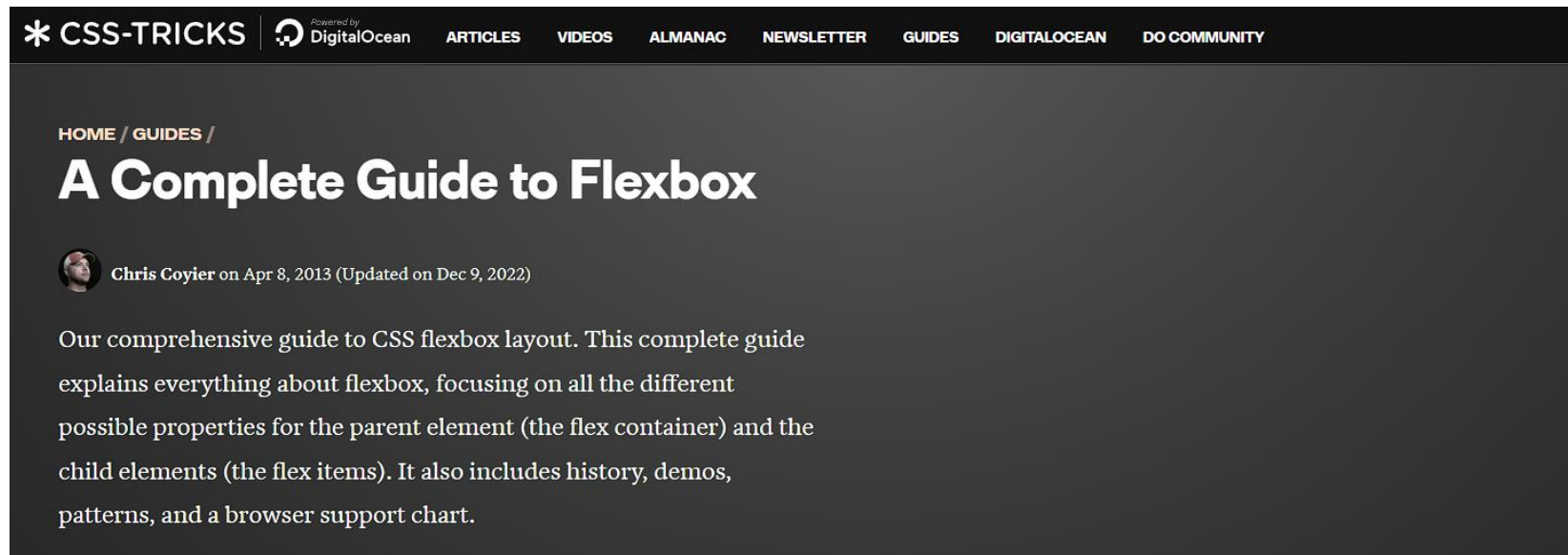
Últimos conceitos importantes:

- O CSS só enxerga a hierarquia pai-filho; não vai aplicar as propriedades **Flex** para elementos que não estejam diretamente relacionados;

# *Flexbox Layout Module*

Leitura complementar e referências:

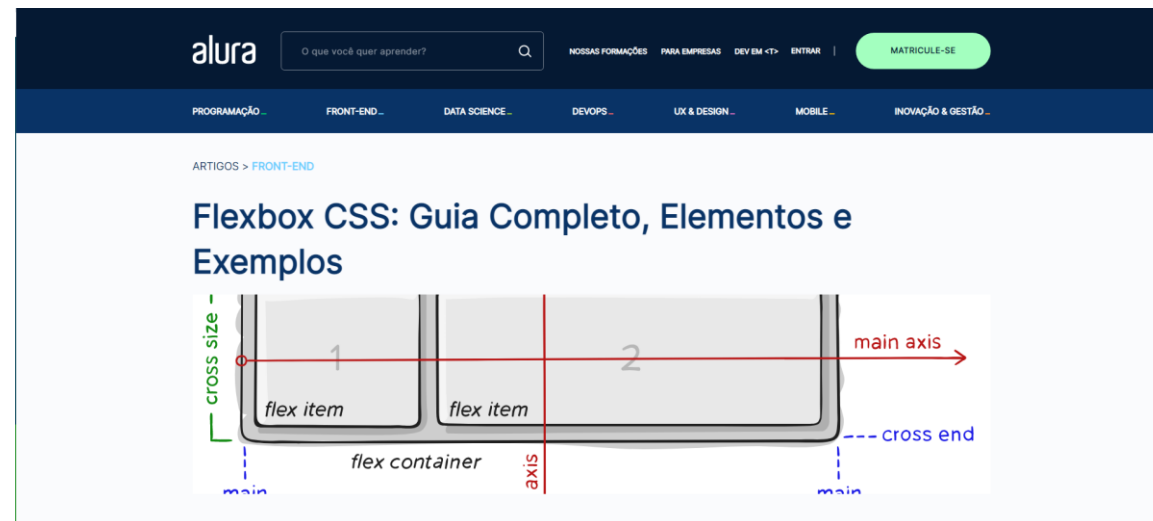
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>



# Flexbox Layout Module

Leitura complementar e referências:

<https://www.alura.com.br/artigos/css-guia-do-flexbox>





# *Flexbox Layout Module*

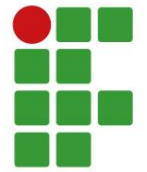
---

Leitura complementar e referências:

[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)

<https://www.chiefdesign.com.br/css-flexbox/>

<https://www.w3.org/TR/css-flexbox-1/>



# Flexbox e Grid

---

Informações complementares, **Flexbox ou Grid?**

<https://www.treinaweb.com.br/blog/flexbox-ou-css-grid>

<https://www.lewagon.com/pt-BR/blog/css-grid-ou-flexbox>

<https://www.felipefialho.com/blog/css-grid-e-flexbox-quando-utilizar/>

<https://www.youtube.com/watch?v=hs3piaN4b5I>





**INSTITUTO  
FEDERAL**  
São Paulo

# Praticando

---

Vamos praticar um pouco com alguns exercícios.





# Praticando

---

## Praticando 1 – *Layout de site com Flexbox*

Utilizar unidades relativas para não mudar o aspecto  
independe da resolução do dispositivo.

# ...: Meu layout de site com Flexbox ...

[Início](#)[Sobre](#)[Produtos](#)[Contato](#)

## Corpo do site

1

*One*

2

*Two*

3

*Three*

4

*Four*

5

*Five*

6

*Six*

7

*Seven*

8

*Eight*

9

*Nine*

10

*Ten*

## Box lateral

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum bibendum elementum mauris, a convallis augue maximus in. Aliquam velit libero, lobortis sit amet odio id, mollis blandit est. Donec volutpat suscipit nibh, at commodo nisl lobortis non. Donec porta sed erat efficitur dignissim. Nullam vulputate, quam eu malesuada condimentum, dui nisl ultricies sapien, hendrerit vulputate mauris nibh quis dui.

# Praticando

---

## Praticando 1 – *Layout de site com Flexbox*

Dicas:

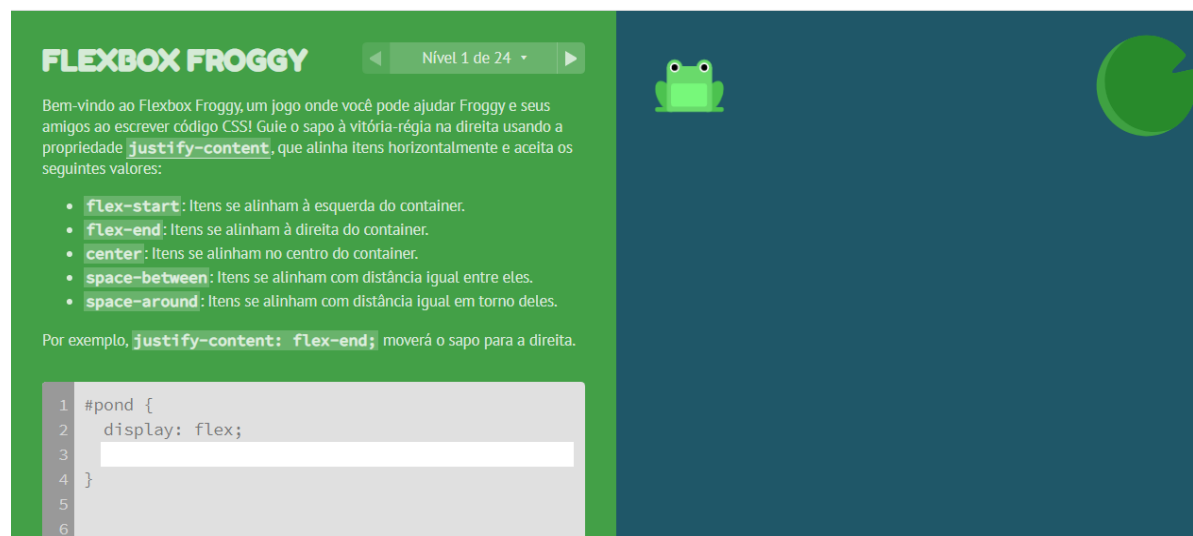
- Fonte “Montserrat”  
<https://fonts.google.com/specimen/Montserrat>
- Cores usadas: `rgb(178, 245, 182)`; e `rgb(14, 165, 25)`;
- Fonte rodapé: 'Courier New'



# Praticando

## Praticando 2 – Chegar ao nível 24 do jogo “*Flexbox Froggy*”

<https://flexboxfroggy.com/#pt-br>



# FLEXBOX FROGGY

← Nível 1 de 24 ▶

Bem-vindo ao Flexbox Froggy, um jogo onde você pode ajudar Froggy e seus amigos ao escrever código CSS! Guie o sapo à vitória-régia na direita usando a propriedade `justify-content`, que alinha itens horizontalmente e aceita os seguintes valores:

- `flex-start`: Itens se alinham à esquerda do container.
- `flex-end`: Itens se alinham à direita do container.
- `center`: Itens se alinham no centro do container.
- `space-between`: Itens se alinham com distância igual entre eles.
- `space-around`: Itens se alinham com distância igual em torno deles.

Por exemplo, `justify-content: flex-end;` moverá o sapo para a direita.

```
1 #pond {
2   display: flex;
3
4 }
5
6
7
8
9
10
```

Próximo

Flexbox Froggy foi criado por [Codepip](#) • [GitHub](#) • [Twitter](#) • [Configurações](#)

Quer aprender CSS grid? Jogue [Grid Garden](#).

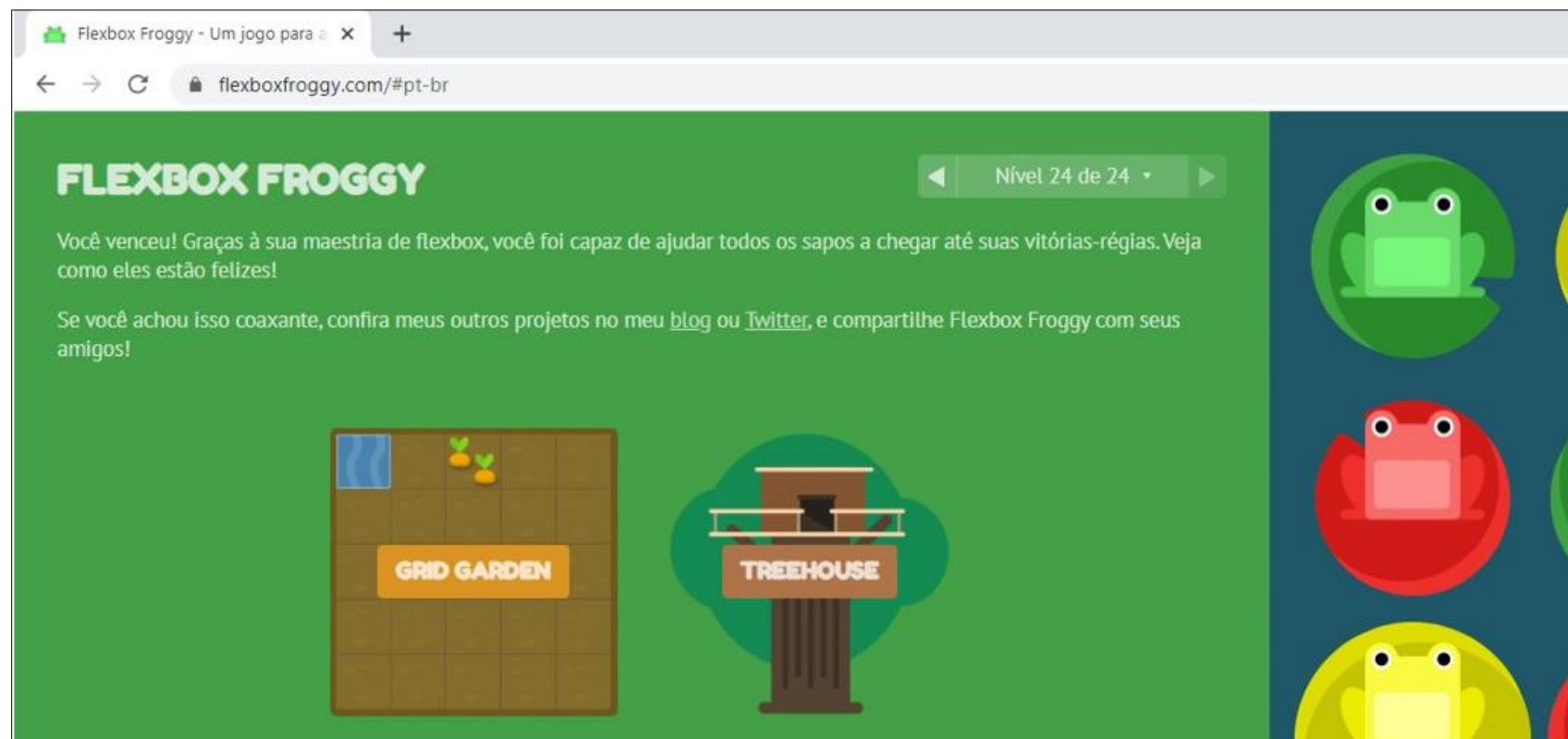


TO  
L



# Praticando

Objetivo:



Dúvidas?

Perguntas?

Sugestões?