



Artigo

Invista em você! Saiba como a DevMedia pode ajudar sua carreira.



# CSS3 Flexbox: Funcionamento e propriedades

Nesta documentação veremos o funcionamento do flexbox no CSS3, suas propriedades e exemplos de aplicações e como criar layouts flexíveis (responsivos) com CSS3.



Anotar



Marcar como concluído

Artigos



CSS



CSS3 Flexbox: Funcionamento e propriedades

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



O **Flexbox** é um conceito do **CSS3** que visa organizar os elementos de uma página HTML dentro de seus containers de forma dinâmica. Ou seja, independente das suas dimensões eles sempre manterão um layout flexível dentro do seu elemento pai, reorganizando-se e acordo com a necessidade.

Neste documento conhiceremos as **propriedades do CSS** que fazem parte desse recurso, analisando seu funcionamento, bem como exemplos práticos de uso.

Saiba mais: [CSS Tutorial Completo](#)

## Tópicos

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar

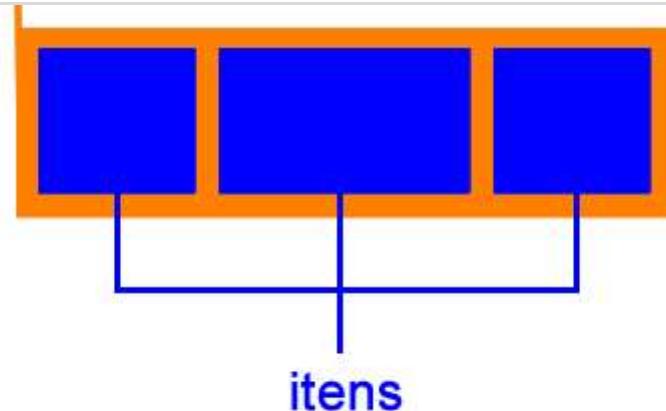
[flex-flow](#)[justify-content](#)[align-content](#)[align-items](#)[order](#)[flex-grow](#)[flex-shrink](#)[flex-basis](#)[flex](#)[align-self](#)[Exemplo prático](#)

## Conceitos iniciais

O **Flexbox** é um conjunto de propriedades que tem por objetivo organizar itens dentro de um elemento pai, normalmente chamado de container, conforme ilustra a **Figura 1**.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

[Aceitar](#)



**Figura 1** Estrutura dos elementos para aplicação do Flexbox

Portanto, para utilizar esse recurso é necessário ter no HTML ao menos um elemento (container) contendo outros (itens), como no código abaixo:

```
1 | <div class="container">
2 |   <div class="item item1"></div>
3 |   <div class="item item2"></div>
4 |   <div class="item item3"></div>
5 | </div>
```

Conforme veremos a seguir, algumas propriedades serão aplicadas ao container, enquanto outras serão aplicadas aos itens.

## Alinhamento dos eixos

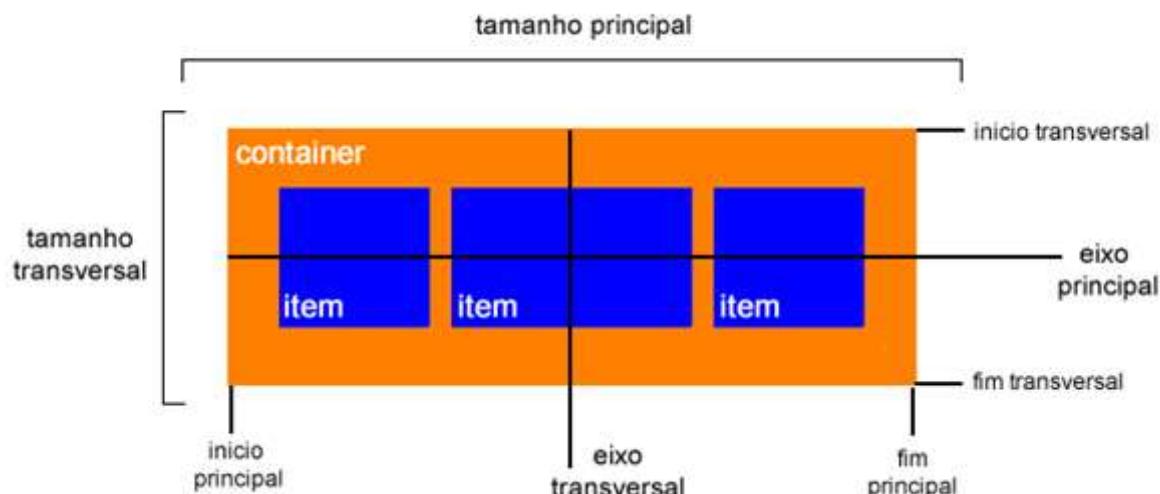
Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



distribuídos no container. O principal deles é o conceito de eixo principal e eixo transversal, que depende do valor atribuído à propriedade flex-direction. Se essa propriedade receber o valor row ou row-reverse (organização em linhas), o eixo principal do container será o horizontal. Já se essa propriedade receber o valor column ou column-reverse (organização em coluna), o eixo principal será o vertical. Consequentemente isso definirá qual é o eixo transversal. Se o principal for o vertical, o transversal será o horizontal e vice-versa.

Na **Figura 2** podemos ver esse e outros conceitos ilustrados para o caso de o eixo principal ser o horizontal.



**Figura 2.** Alinhamento dos eixos do Flexbox

- **Tamanho principal:** É a dimensão do elemento na direção do eixo principal (largura, caso horizontal e altura caso vertical);

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



- **Início transversal e final transversal:** Representam o início e o fim do eixo transversal.

Essas informações serão importantes para compreendermos o funcionamento das diversas **propriedades do Flexbox** que veremos a seguir.

## display

O primeiro passo para **utilizar o Flexbox** é definir a propriedade `display` do container com o valor `flex`. Isso é necessário para que as demais propriedades apresentem o resultado esperado.

A sintaxe de uso dessa propriedade é a seguinte:

```
1 | .container {  
2 |     display: flex;  
3 | }
```

## flex-direction

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

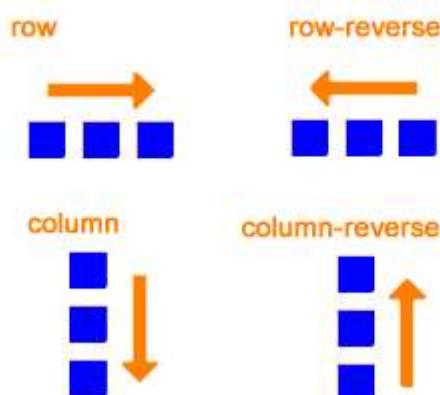
Aceitar



```
1 .container {  
2     display: flex;  
3     flex-direction: [row / row-reverse / column / column-reve  
4 }
```

- **row** (padrão): Os itens são organizados em forma de linha da esquerda para a direita;
- **row-reverse**: Os itens são organizados em forma exibição em linha da direita para a esquerda;
- **column**: Os itens são organizados em forma de colunas iniciando de cima para baixo;
- **column-reverse**: Os itens são organizados em forma de colunas iniciando de baixo para cima.

A **Figura 3** ilustra o funcionamento de cada valor.



**Figura 3.** Funcionamento da propriedade `flex-direction`

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



Por padrão os itens do container tentarão se ajustar em uma única linha dentro do container, mas para isso a sua largura original pode ser ajustada para que todos caibam na largura do elemento pai. Com a propriedade `flex-wrap` aplicada ao container podemos alterar esse comportamento, fazendo com que ocorra a “quebra de linha” nos itens.

A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:

```
1 | .container {  
2 |     display: flex;  
3 |     flex-wrap: [nowrap / wrap / wrap-reverse];  
4 | }
```

- **nowrap** (padrão): Todos os itens serão dispostos em uma linha;
- **wrap**: Ocorrerá a quebra de linha e os itens mais à direita serão deslocados para a linha de baixo;
- **wrap-reverse**: Ocorrerá a quebra de linha e os itens mais à direita serão deslocados para a linha de cima;

## flex-flow

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).



forma:

```
1 | .container {  
2 |   display: flex;  
3 |   flex-direction: column;  
4 |   flex-wrap: wrap;  
5 | }
```

Já com o flex-flow podemos escrever as duas de forma simplificada:

```
1 | flex-flow: column wrap;
```

## justify-content

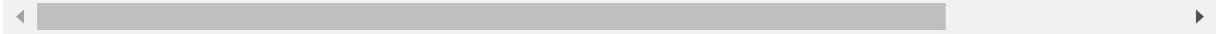
A propriedade `justify-content` define o alinhamento dos itens ao longo do eixo principal do container. A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:

```
justify-content
```

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



- **flex-start (padrão):** Os itens são alinhados a partir do início do eixo
- 
- **flex-end:** Os itens são alinhados a partir do fim do eixo principal;
- **center:** Os itens são alinhados ao centro do eixo principal;
- **space-between:** O primeiro item é deslocado para o início do eixo principal, o último é deslocado para o final do eixo principal e os demais são distribuídos uniformemente entre eles;
- **space-around:** Os itens são uniformemente distribuídos ao longo do eixo principal. Aqui, porém, são atribuídas margens iguais à esquerda e à direita (ou acima e abaixo, dependendo da direção do eixo principal). Por isso o primeiro e o último item não ficam “colados” nas bordas do container.

A **Figura 4** ilustra o funcionamento de cada valor:

**Figura 4** Representação da propriedade justify-content

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).



linha, ou seja, se ele tiver elementos suficientes para quebrar a linha e a propriedade `flex-wrap:wrap` tiver sido definida. A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:

```
1 .container {  
2     display: flex;  
3     align-content: [stretch/flex-start/flex-end/center/space-  
4 }
```

- **stretch** (padrão): As linhas são distribuídas uniformemente ao longo do eixo transversal, ocupando todo o espaço disponível;
- **flex-start**: As linhas são distribuídas a partir do início do eixo transversal;
- **flex-end**: As linhas são distribuídas a partir do fim do eixo transversal;
- **center**: As linhas são mantidas no centro do eixo transversal;
- **space-between**: A primeira linha é deslocada para o início do eixo transversal, a última é deslocada para o final do eixo transversal e as demais são distribuídas uniformemente entre eles;
- **space-around**: As linhas são uniformemente distribuídas ao longo do eixo transversal. Aqui, porém, são atribuídas margens iguais à esquerda e à direita (ou acima e abaixo, dependendo da direção do eixo transversal). Por isso a primeira e a última linha não ficam “coladas” nas bordas do container.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



## Figura 5. Funcionamento da propriedade align-content

### align-items

Essa propriedade define como os itens são distribuídos ao longo do eixo transversal do container. A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:

```
1 | .container {  
2 |   display: flex;  
3 |   align-items: [stretch/flex-start/flex-end/center/baseline];  
4 | }
```

- **stretch** (padrão): Os itens serão esticados para preencher toda a dimensão do eixo transversal (altura ou largura);
- **flex-start**: Os itens são deslocadas para o início do eixo transversal;
- **flex-end**: Os itens são deslocadas para o final do eixo transversal;
- **center**: Os itens são centralizados no eixo transversal;
- **baseline**: Os itens são alinhados a partir da base da primeira linha de texto de cada um.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



**Figura 6.** Representação da propriedade align-items

## Order

Por padrão, os itens são distribuídos no container na ordem em que são inseridos no HTML. No entanto, essa ordem pode ser alterada por meio da propriedade order, cuja sintaxe vemos abaixo:

```
1 | .item2 {  
2 |   order: [número];  
3 | }
```

O valor numérico atribuído a essa propriedade define a ordem do item. Por exemplo, o valor 2 faz com que o item seja o segundo item ao longo do eixo principal, enquanto o valor -1 faz com que ele apareça antes do primeiro.

## flex-grow

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



```
1 | .item2 {  
2 |   flex-grow: [número];  
3 | }
```

## flex-shrink

Esta propriedade define a proporção com que um item deve encolher caso seja necessário. Essa propriedade aceita apenas valores positivos, e seu valor padrão é 1.

A sintaxe dessa propriedade é a seguinte:

```
1 | .item2 {  
2 |   flex-shrink: [número];  
3 | }
```

## flex-basis

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



eixo principal (horizontal ou vertical), essa propriedade define a largura ou altura mínima do elemento antes que ele seja redimensionado por outras propriedades.

A sintaxe dessa propriedade é a seguinte:

```
1 | .item2 {  
2 |   flex-basis: [número];  
3 | }
```

O valor atribuído a essa propriedade pode ser em percentual, em pixels, ou a palavra auto, que é o valor padrão (considera as dimensões do item - width e height).

## flex

Esta propriedade é uma forma abreviada para a escrita das propriedades `flex-grow`, `flex-shrink` e `flex-basis`, nesta ordem.

A sintaxe dessa propriedade é a seguinte:

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



3 | 5

## align-self

Esta propriedade permite sobrescrever no item o comportamento que foi definido pela propriedade align-items.

A sintaxe e os valores possíveis para essa propriedade são apresentados a seguir:

```
1  .item2 {  
2      align-self: [auto/stretch/flex-start/flex-end/center/base  
3  }
```

- **auto** (padrão): Respeita o comportamento definido no container por meio do align-items;
- **stretch**: O item será esticado para preencher toda a dimensão do eixo transversal (altura ou largura);
- **flex-start**: O item é deslocado para o início do eixo transversal;
- **flex-end**: O item é deslocado para o final do eixo transversal;
- **center**: O item é centralizado no eixo transversal;

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



## Exemplo prático

A seguir podemos ver um exemplo prático de uso do Flexbox.

Primeiramente temos a estrutura do HTML na qual foram aplicadas as propriedades que vimos anteriormente:

```
1 <div class="container">
2     <div class="item item1">1</div>
3     <div class="item item2">2</div>
4     <div class="item item3">3</div>
5 </div>
```

E abaixo temos o código CSS responsável pela formatação desse layout:

```
1 .container {
2     background-color: #FF5722;
3     height: 100vh;
4     width: 100%;
5     display: flex;
6     flex-direction: column;
7     justify-content: center;
8     align-items: center;
9 }
```

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



```
17  
18 .item1{  
19     height: 100px;  
20     width: 100px;  
21 }  
22  
23 .item2{  
24     height: 50px;  
25     width: 150px;  
26 }  
27  
28 .item3{  
29     height: 100px;  
30     width: 200px;  
31 }
```

Run

**Linha 2:** Definimos a cor do plano de fundo do container para facilitar a visualização;

**Linhas 3 e 4:** Definimos a altura e largura do container para ocupar 100% da página;

**Linha 5:** Definimos o display flex no container para que as demais propriedades surtam o efeito esperado;

**Linha 6:** Com a propriedade flex-direction definida como column definimos que os itens devem ser organizados em coluna (um abaixo do outro);

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



**Linhas 11 a 30:** Neste trecho estamos definindo algumas características dos itens. Eles possuem diferentes alturas e larguras, o que facilita a visualização do resultado final (mesmo com diferentes dimensões, o layout é mantido organizado).

Curso relacionado: [CSS3 Display Flex: Como organizar cards e boxes automaticamente](#)

Tecnologias:

CSS

HTML



Anotar



Marcar como concluído



**Confira outros conteúdos:**

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



## FONTES

Lista de Fontes padrão no CSS

## UNIDADES

CSS: Unidades



Por Fernando

Em 2013

## Comentários nesta publicação

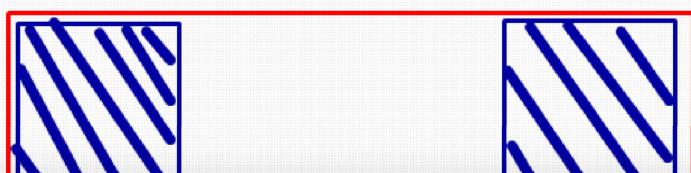
Escrever um comentário sobre conteúdo



Thiago Lessa

Nível 35

Depois de um tempo sem conseguir estudar aqui, estou com uma dúvida. Caso eu tenha duas divs dentro de uma outra div, e eu quero deixar essas duas divs uma em um canto e a outra no outro canto, com esse espaço vazio no meio entre elas, como eu poderia estar fazendo com flex box? O space-between funcionaria jogando ambos para os cantos?



Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar

**Juliano**

DevMedia

Boa noite Thiago blz?

Sim, você pode usar a propriedade "justify-content" do flexbox para alinhar as duas divs nos cantos opostos e criar um espaço vazio no meio. A opção "space-between" distribui o espaço disponível igualmente entre os itens flexíveis, empurrando o primeiro item para o começo do contêiner e o último item para o final, deixando o espaço vazio no meio.

o contêiner é definido como um flex container e a propriedade "justify-content" é definida como "space-between". Isso fará com que as duas divs sejam alinhadas aos cantos opostos e criará um espaço vazio no meio.

As divs são alinhadas aos cantos opostos usando a propriedade "align-self" com o valor "flex-start" para a primeira div e "flex-end" para a segunda div.

Você pode ajustar o espaço vazio no meio ajustando a largura do contêiner ou das divs internas.

```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
display: flex;
justify-content: space-between;
}

.div-1 {
align-self: flex-start;
background-color: red;
width: 100px;
height: 100px;
}
```

Lia Souza

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



```
</head>
<body>
<div class="container">
<div class="div-1">Div 1</div>
<div class="div-2">Div 2</div>
</div>
</body>
</html>
```

Ficou claro?

há 8 meses



**Felypi Tanida**

Nível 16

Professor, por que neste exemplo possui numeros no começo, igual na imagem que vou mostrar

```
1 .container {
2   display: flex;
3   flex-direction: column;
4   04   flex-wrap: wrap;
5   05 }
```

há 8 meses



**Rodolfo Gomes**

DevMedia

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



Respondido em tempo real

**Felypi Tanida**

Nível 16

**O que é eixo transversal?**

há 8 meses

**Rodolfo Gomes**

DevMedia

**Fala Felype, blz?**

O eixo transversal será o alinhamento na vertical, onde alinharemos os itens de cima para baixo e vice versa.

**Ficou mais claro?!**

há 8 meses

Respondido em tempo real

**Thiago Lessa**

Nível 35

Opa, estou com dúvida em relação ao align-items e o align-content. Não entendi muito bem a diferença de ambos. Para mim os dois são a mesma

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar

**Juliano**

DevMedia

Boa noite Thiago blz?

align-items é usado para alinhar os itens dentro do container flex ou grid ao longo do eixo transversal (verticalmente, em um layout flexbox, e horizontalmente, em um layout grid). Em outras palavras, ele controla a posição vertical dos itens em relação ao container.

Por outro lado, align-content é usado para alinhar o próprio conteúdo do container flex ou grid ao longo do eixo transversal quando há mais de uma linha de itens. Ou seja, ele controla o posicionamento vertical do container em relação aos itens que estão dentro dele.

A principal diferença é que align-items aplica-se aos itens individualmente, enquanto align-content aplica-se ao container como um todo quando há mais de uma linha de itens.

Resumindo, align-items alinha os itens dentro do container e align-content alinha o próprio container quando há mais de uma linha de itens

Ficou claro amigo?

há 8 meses

Respondido em tempo real

**Andrey Borges**

Nível 18

Eu não entendi direito align-content e align-items.  
Fiz umas box aqui e no CSS alinhei tudo no centro, e usei align-content e depois, align-items e ficou a mesma coisa.  
Tem alguma diferença?

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários,  
consulte nossa [política de privacidade](#).

 Aceitar



```
</div>
<div class="flex-box">
<h1>item3</h1>
</div>
</div>
```

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
.container {
  display: flex;
  flex-flow: row wrap;
  align-content: center;
  justify-content: center;
  background-color: black;
  padding: 10px;
  gap: 20px;
  height: 50vh;
}

.flex-box {
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: yellow;
  width: 100px;
  height: 100px;
}
```

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar

**Rodolfo Gomes**

DevMedia

Opa Andrey, tudo bem?

Veja que a principal diferença entre as propriedades está na definição do -content:

Essa propriedade define como as linhas são distribuídas ao longo do eixo transversal do container. Ela só terá efeito se o elemento tiver mais de uma linha, ou seja, se ele tiver elementos suficientes para quebrar a linha e a propriedade flex-wrap:wrap tiver sido definida.

Isso quer dizer que a propriedade align-content alinhará as linhas e não os itens em si.

Então como no seu exemplo tevem poucos itens não viu diferença prática entre eles, mas fique com essa recordação que a utilização do align-items será para alinhar os itens em si no eixo principal ou transversal em função do flex-direction definido.

E o align-content alinhará as linhas do container.

Ficou mais claro?!

há 10 meses

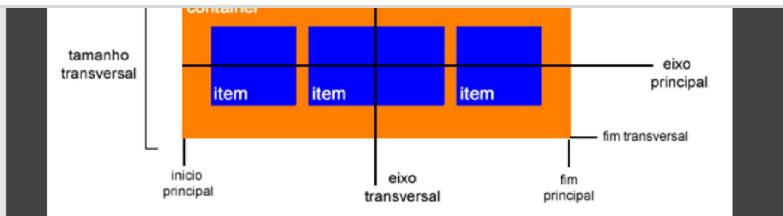
**Andrey Borges**

Nível 18

Não entendi esse eixo transversal e principal. Na imagem o transversal parece ser vertical e principal o horizontal.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



**Figura 2.** Alinearmento dos eixos do Flexbox

há 10 meses

[Ver comentários anteriores \(6\)](#)



**Rodolfo Gomes**

DevMedia

Ah legal meu amigo, fico feliz em ajuda lo.

Isso mesmo, essas são as direções de cada eixo 😊

há 10 meses



**Andrey Borges**

Nível 18

Aqui nesse artigo mostra bem pouco das propriedades flex-grow e flex-shrink. Ficou meio vago e sem exemplos, tem como explicar melhor?

há 10 meses



**Rodolfo Gomes**

DevMedia

Opa Andrey

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

[Aceitar](#)



tamanho dentro do espaço disponível no container.

E flex-shrink possui o comportamento inverso, onde ele reduzirá o tamanho na proporção definida, se definir 2 ele reduzirá em duas vezes o tamanho dos itens do que seria necessário para caber dentro do espaço disponível.

há 10 meses

Respondido em tempo real



Silvio Junior

Nível 34



Bom dia, professor! Tudo bem?

Tem uma dúvida no flex-wrap. Por que ele precisa ser colocado no flex-direction: column se o column já é um em baixo do outro?

Não vai ter quebra de linha

há 10 meses

[Ver comentários anteriores \(4\)](#)



Rodolfo Gomes

DevMedia

Isso mesmo irmão!! 😊

há 10 meses

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



esquerda.

2. Caso necessário deve permitir quebra de linha.
3. Utilize apenas uma propriedade para essas definições.

```
digite
{
display: flex;
```

```
digite
;
}
```

nao consigo acha nenhuma opção de codigo que de certo

há 10 meses



**Rodolfo Gomes**

DevMedia

Fala André, tudo bem?

Assim:

```
header
{
display: flex;

flex-flow : wrap row-reverse;
}
```

Repare que o flex-flow é uma forma abreviada para a escrita das propriedades flex-direction e flex-wrap, nesta ordem. Desta forma:

wrap - 2. Caso necessário deve permitir quebra de linha.

row-reverse - 1. Os elementos devem ficar um ao lado do outro invertidos da direita para esquerda.

Ficou mais claro?!

há 10 meses

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



Nível 13

Queria ter um documento em PDF sobre esses assuntos acima, seria bom para via de consulta. Pode ser?

há 11 meses



Rodolfo Gomes

DevMedia

Opa Jooelberth, tudo bem?

Não temos este conteúdo num material offline somente o disponível na plataforma.

Mas obrigado pela sugestão meu amigo 😊

TMJ

há 11 meses

Respondido em tempo real



Escola de Programação

# AQUI TODO MUNDO

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



Em caso de dúvidas chame no whatsapp

MELHOR OPÇÃO

## Plano Recorrente

R\$ 99,00 /MÊS PRIMEIROS  
3 MESES

R\$ 59,00 /MÊS A PARTIR  
DO 4º MÊS

12 MESES = R\$ 828,00

Formação FullStack completa

+10mil exercícios gamificados

+50 projetos reais

Supporte online

Pra quem tem pouco limite no cartão

Fidelidade de 12 meses

## Plano Anual

12X R\$ 69,00

12 MESES = R\$ 828,00

Formação FullStack completa

+10mil exercícios gamificados

+50 projetos reais

Supporte online

Matricule-se

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários,  
consulte nossa [política de privacidade](#).

Aceitar



**Por que a programação se tornou a profissão mais promissora da atualidade?**



**Como faço para começar a estudar?**



**Em quanto tempo de estudo vou me tornar um programador?**



**Sim, você pode se tornar um programador e não precisa ter diploma de curso superior!**



**O que eu irei aprender estudando pela DevMedia?**



**Principais diferenciais da DevMedia**



**Qual o investimento financeiro que preciso fazer para me tornar um programador?**



**Como funciona a forma de pagamento da DevMedia?**



## Nossos casos de sucesso



**Leonardo Carlos**



Eu sabia pouquíssimas coisas de programação antes de começar a estudar

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



Estudo aqui na Dev desde o meio do ano passado! Nesse período a Dev me ajudou a crescer muito aqui no trampo.

**Fui o primeiro desenvolvedor contratado pela minha empresa. Hoje eu lidero um time de desenvolvimento!**

Minha meta é continuar estudando e praticando para ser um Full-Stack Dev!



**Heráclito Júnior**



Economizei 3 meses para assinar a plataforma e sendo sincero valeu muito a pena, pois a **plataforma é bem intuitiva e muuuuito didática a metodologia de ensino.** Sinto que estou EVOLUINDO a cada dia. Muito obrigado!



**Julio Cahlen**



Nossa! Plataforma maravilhosa. To amando o curso de desenvolvimento front-end, tinha coisas que eu ainda não tinha visto. **A didática é do jeito que qualquer pessoa consegue aprender.** Sério, to apaixonado, adorando demais.



**Joelberth Sena**



Adquiri o curso de vocês e logo percebi que são os melhores do Brasil. É um passo a passo incrível. **Só não aprende quem não quer. Foi o melhor investimento da minha vida!**



**Felipe Nunes**



Foi um dos melhores investimentos que já fiz na vida e tenho aprendido bastante com a plataforma. Vocês estão fazendo parte da minha jornada nesse mundo da programação, **irei assinar meu contrato como programador gracas a plataforma**.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

Aceitar



aprender, estão de parabéns!



José Lucas



Obrigado DevMedia, nunca presenciei **uma plataforma de ensino tão presente na vida acadêmica de seus alunos**, parabéns!



Eduardo Dorneles



Aprendi React na plataforma da DevMedia há cerca de 1 ano e meio... **Hoje estou há 1 ano empregado** trabalhando 100% com React!



Adauto Junior



Já fiz alguns cursos na área e **nenhum é tão bom quanto o de vocês**. Estou aprendendo muito, muito obrigado por existirem. Estão de parabéns... Espero um dia conseguir um emprego na área.

[Ver todos os casos de sucesso](#)

## Menu

- Assine agora
- Quem Somos
- Fale conosco
- Plano para Instituição de ensino
- Assinatura para empresas



Hospedagem web por Porta 80 Web Hosting.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários, consulte nossa [política de privacidade](#).

[Aceitar](#)



SQL e Banco de Dados • Engenharia de Software • Canal Mais • Grátis

### Artigos:

Front-End • Javascript • Iniciantes • Angular • Dart • Engenharia • Mobile • Node.js • Python • React Native • Vue.js • Android • Banco de Dados • Delphi • Flutter • Java • Kotlin • .Net • PHP • React • Spring • Grátis

### DevCast:

HTML e CSS • Javascript • Angular • Engenharia • Mobile • Node.js • Python • React Native • Android • Banco de Dados • Delphi • Flutter • Java • Automação • .Net • PHP • React • Spring • Grátis • Canal Mais

### Guia:

Fundamentos • .NET • PHP • Python • Java • Delphi • HTML e CSS • JavaScript • Node • React Native • Flutter • Banco de Dados • Mobile • Spring • Arquitetura • Automação • Engenharia • + Assuntos

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários,  
consulte nossa [política de privacidade](#).

Aceitar