ORIGAMID

# Guia Flexbox

## Flex Container



## Flex Item



# Flex Container

O Flex Container é a tag que envolve os itens flex, ao indicar display: flex, essa tag passa a ser um Flex Container.

## 1 • display

Define o elemento como um flex container, tornando os seus filhos flex-itens.

```
display: flex;
// Torna o elemento um flex container automaticamente transformando todos os seus filhos diretos em flex itens.
```

HTML CSS Result EDIT ON LIVE

```
/* Flex */
.flex {
  display: flex;
}

.flex-wrap {
  flex-wrap: wrap;
}

.flex-item-1 {
  flex: 1;
}

/* Flex Item */
.item {
  margin: 5px;
  background: tomato;
```

Resources

# 2 • flex-direction

Define a direção dos flex itens. Por padrão ele é row (linha), por isso quando o display: flex; é adicionado, os elementos ficam em linha, um do lado do outro.

A mudança de row para column geralmente acontece quando estamos definindo os estilos em media queries para o mobile. Assim você garante que o conteúdo seja apresentado em coluna única.

```
flex-direction: row;
// Os itens ficam em linha
```

```
flex-direction: row-reverse;
// Os itens ficam em linha reversa, ou seja 3, 2, 1.
```

```
flex-direction: column;
// Os itens ficam em uma única coluna, um embaixo do outro.
```

```
flex-direction: column-reverse;
// Os itens ficam em uma única coluna, um embaixo do outro, porém em ordem reversa: 3, 2 e 1.
```

HTML CSS Result EDIT ON LIVE

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

/* Flex Container */
.container {
  max-width: 400px;
  margin: 0 auto;
```

Resources

# 3 • flex-wrap

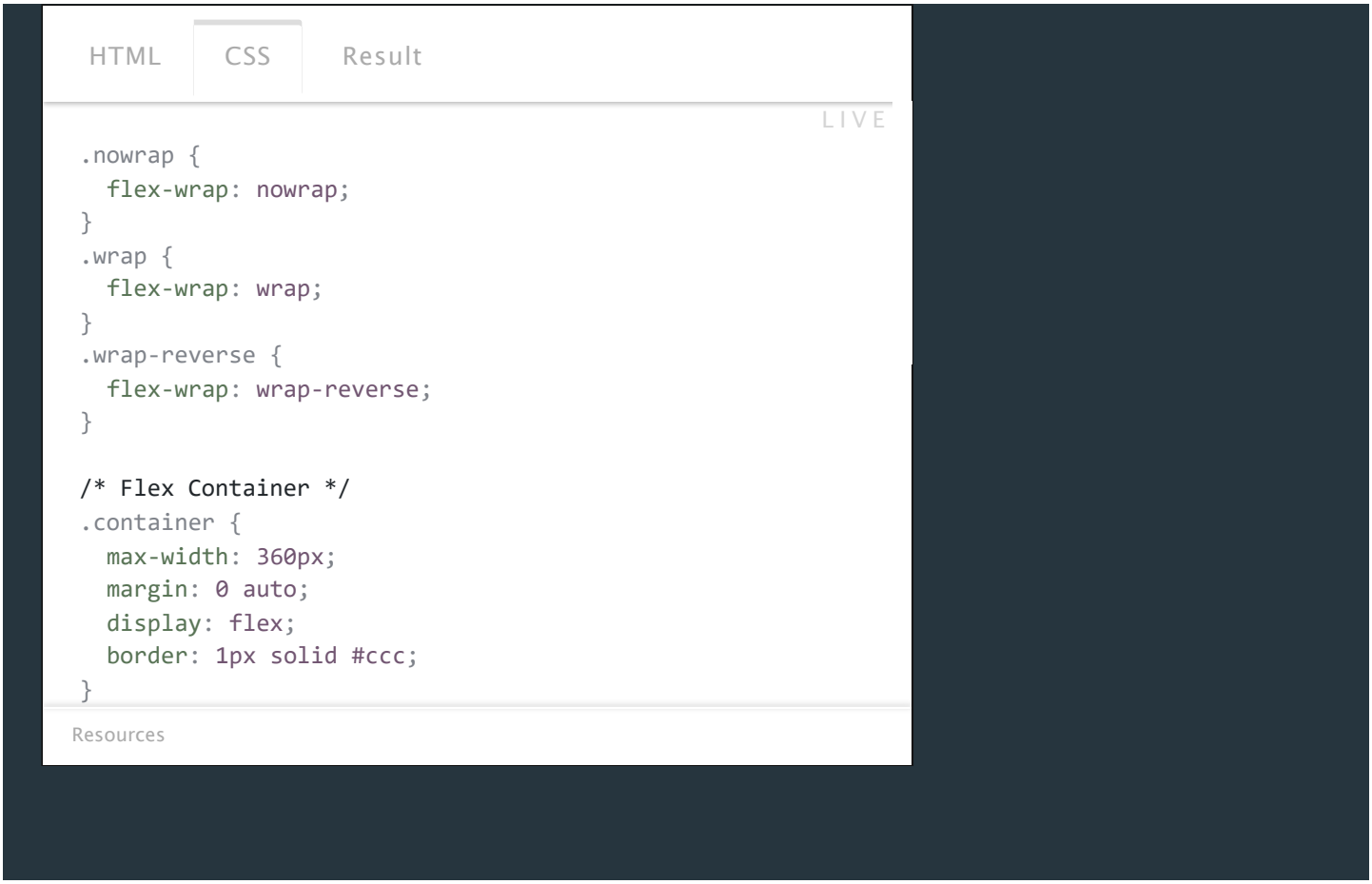
Define se os itens devem quebrar ou não a linha. Por padrão eles não quebram linha, isso faz com que os flex itens sejam compactados além do limite do conteúdo.

Essa é geralmente uma propriedade que é quase sempre definida como flex-wrap: wrap; Pois assim quando um dos flex itens atinge o limite do conteúdo, o último item passa para a coluna debaixo e assim por diante.

```
flex-wrap: nowrap;
// Valor padrão, não permite a quebra de linha.
```

```
flex-wrap: wrap;
// Quebra a linha assim que um dos flex itens não puder mais ser compactado.
```

```
flex-wrap: wrap-reverse;
// Quebra a linha assim que um dos flex itens não puder mais ser compactado. A quebra é na direção contrária, ou seja para a linha acima.
```




# 4 • flex-flow

O flex-flow é um atalho para as propriedades flex-direction e flex-wrap. Você não verá muito o seu uso, pois geralmente quando mudamos o flex-direction para column, mantemos o padrão do flex-wrap que é nowrap.

E quando mudamos o flex-wrap para wrap, mantemos o padrão do flex-direction que é row.

```
flex-flow: row nowrap;
// Coloca o conteúdo em linha e não permite a quebra de linha.
```

```
flex-flow: row wrap;
// Coloca o conteúdo em linha e permite a quebra de linha.
```

```
flex-flow: column nowrap;
// Coloca o conteúdo em coluna e não permite a quebra de linha.
```

HTML CSS Result

LIVE

```
.row-nowrap {
  flex-flow: row nowrap;
}
.column-nowrap {
  flex-flow: column nowrap;
}
.row-wrap {
  flex-flow: row wrap;
}

/* Flex Container */
.container {
  max-width: 360px;
  margin: 0 auto;
  display: flex;
  border: 1px solid #ccc;
}
```

Resources

# 5 • justify-content

Alinha os itens flex no container de acordo com a direção. A propriedade só funciona se os itens atuais não ocuparem todo o container. Isso significa que ao definir flex: 1; ou algo similar nos itens, a propriedade não terá mais função

Excelente propriedade para ser usada em casos que você deseja alinhar um item na ponta esquerda e outro na direita, como em um simples header com marca e navegação.

```
justify-content: flex-start;
// Alinha os itens ao início do container.
```

```
justify-content: flex-end;
// Alinha os itens ao final do container.
```

```
justify-content: center;
// Alinha os itens ao centro do container.
```

```
justify-content: space-between;
// Cria um espaçamento igual entre os elementos. Mantendo o primeiro grudado no início e o último no final.
```

```
justify-content: space-around;
// Cria um espaçamento entre os elementos. Os espaçamentos do meio
```

```
são duas vezes maiores que o inicial
e final.
```

HTML CSS Result

LIVE

```
.flex-start {
  justify-content: flex-start;
}

.flex-end {
  justify-content: flex-end;
}

.center {
  justify-content: center;
}

.space-between {
  justify-content: space-between;
}

.space-around {
```

Resources

# 6 • align-items

O align-items alinha os flex itens de acordo com o eixo do container. O alinhamento é diferente para quando os itens estão em colunas ou linhas.

Essa propriedade permite o tão sonhado alinhamento central no eixo vertical, algo que antes só era possível com diferentes hacks.

```
align-items: stretch;
// Valor padrão, ele que faz com que
os flex itens cresçam igualmente.
```

```
align-items: flex-start;
// Alinha os itens ao início.
```

```
align-items: flex-end;
// Alinha os itens ao final.
```

```
align-items: center;
// Alinha os itens ao centro.
```

```
align-items: baseline;
// Alinha os itens de acordo com a
linha base da tipografia.
```

HTML CSS Result

LIVE

```
.stretch {
  align-items: stretch;
}

.flex-start {
  align-items: flex-start;
}

.flex-end {
  align-items: flex-end;
}

.center {
  align-items: center;
}

.baseline {
```

Resources

# 7 • align-content

Alinha as linhas do container em relação ao eixo vertical. A propriedade só funciona se existir mais de uma linha de flex-itens. Para isso o flex-wrap precisa ser wrap.

Além disso o efeito dela apenas será visualizado caso o container seja maior que a soma das linhas dos itens. Isso significa que se você não definir height para o container, a propriedade não influencia no layout.

```
align-content: stretch;
// Valor padrão, ele que faz com que os flex itens cresçam igualmente na vertical.

align-content: flex-start;
// Alinha todas as linhas de itens ao início.

align-content: flex-end;
// Alinha todas as linhas de itens ao final.

align-content: center;
// Alinha todas as linhas de itens ao centro.

align-content: space-between;
// Cria um espaçamento igual entre as linhas. Mantendo a primeira grudada no topo e a última no bottom.
```

```
align-content: space-around;
// Cria um espaçamento entre as linhas. Os espaçamentos do meio são duas vezes maiores que o top e bottom.
```


# Flex Item

Os Flex Itens são os filhos diretos do Flex Container, lembrado que uma tag se torna um flex container a partir do momento que você definir display: flex.

É possível que um Flex Item seja também um Flex Container, basta definir display: flex nele. Assim os filhos desse item também serão flex itens.

# 1 • flex-grow

Define a habilidade de um flex item crescer. Por padrão o valor é zero, assim os flex itens ocupam um tamanho máximo

```
flex-grow: número;
// Basta definir um número
```

relacionado o conteúdo interno deles ou ao width definido.

Ao definir 1 para todos os Flex Itens, eles tentarão ter a mesma largura e vão ocupar 100% do container. Digo tentarão pois caso um elemento possua um conteúdo muito largo, ele irá respeitar o mesmo.

Se você tiver uma linha com quatro itens, onde três são flex-grow: 1 e um flex-grow: 2, o flex-grow: 2 tentará ocupar 2 vezes mais espaço extra do que os outros elementos.

OBS: justify-content não funciona em items com flex-grow definido.

```
flex-grow: 0;
// Obedece o width do elemento ou o flex-basis.
```

## 2 • flex-basis

Indica o tamanho inicial do flex item antes da distribuição do espaço restante.

```
flex-basis: auto;
// Esse é o padrão, ele faz com que a largura da base seja igual a do item. Se o item não tiver tamanho
```

Quando definimos o flex-grow: 1; e possuímos auto no basis, o valor restante para ocupar o container é distribuído ao redor do conteúdo do flex-item.

```
especificado, o tamanho será de
acordo com o conteúdo.
```

```
flex-basis: unidade;
// Pode ser em %, em, px e etc.
```

```
flex-basis: 0;
// Se o grow for igual ou maior que
1, ele irá tentar manter todos os
elementos com a mesma largura,
independente do conteúdo (por isso 0
é o valor mais comum do flex-basis).
Caso contrário o item terá a largura
do seu conteúdo.
```

# 3 • flex-shrink

Define a capacidade de redução de tamanho do item.

```
flex-shrink: 1;
// Valor padrão, permite que os itens
tenham os seus tamanhos (seja esse
tamanho definido a partir de width ou
flex-basis) reduzidos para caber no
container.
```

```
flex-shrink: 0;
// Não permite a diminuição dos itens, assim um item com flex-basis: 300px; nunca diminuirá menos do que 300px, mesmo que o conteúdo não ocupe todo esse espaço.

flex-shrink: número;
// Um item com shrink: 3 diminuirá 3 vezes mais que um item com 1.
```

# 4 • flex

Atalho para as propriedades flex-grow, flex-shrink e flex-basis. Geralmente você verá a propriedade flex nos flex itens ao invés de cada um dos valores separados.

Para melhor consistência entre os browsers, é recomendado utilizar a propriedade flex ao invés de cada propriedade separada.

No exemplo é possível ver as mesmas configurações do exemplo do flex-basis

```
flex: 1;
// Define flex-grow: 1; flex-shrink: 1; e flex-basis: 0; (em alguns browsers define como 0%, pois estes ignoram valores sem unidades, porém a função de 0 e 0% é a mesma.)

flex: 0 1 auto;
// Esse é o padrão, se você não definir nenhum valor de flex ou para as outras propriedades separadas, o
```

porém agora utilizando apenas a propriedade flex.

```
normal será flex-grow: 0, flex-shrink: 1 e flex-basis: auto.

flex: 2;
// Define exatamente da mesma forma que o flex: 1; porém neste caso o flex-grow será de 2, o flex-shrink continuará 1 e o flex-basis 0.

flex: 3 2 300px;
// flex-grow: 3, flex-shrink: 2 e flex-basis: 300px;
```

# 5 • order

Modifica a ordem dos flex itens. Sempre do menor para o maior, assim order: 1, aparece na frente de order: 5.

```
order: número;
// Número para modificar a ordem padrão. Pode ser negativo.

order: 0;
// 0 é o valor padrão e isso significa que a ordem dos itens será a ordem apresentada no HTML. Se você quiser colocar um item do meio da
```

```
lista no início da mesma, sem
modificar os demais, o ideal é
utilizar um valor negativo para este
item, já que todos os outros são 0.
```

## 6 • align-self

O align-self serve para definirmos o alinhamento específico de um único flex item dentro do nosso container. Caso um valor seja atribuído, ele passara por cima do que for atribuído no align-items do container.

Vale lembrar que o alinhamento acontece tanto em linha quanto em colunas. Por exemplo o flex-start quando os itens estão em linhas, alinha o item ao topo da sua linha. Quando em colunas, alinha o item ao início (esquerda) da coluna.

```
align-self: auto;
// Valor inicial padrão. Vai
respeitar o que for definido pelo
align-items no flex-container.
```

```
align-self: flex-start;
// Alinha o item ao início.
```

```
align-self: flex-end;
// Alinha o item ao final.
```

```
align-self: center;
// Alinha o item ao centro.
```

```
align-self: baseline;
// Alinha o item a linha de base.
```

```
align-self: stretch;
// Estica o item.
```

Origamid © 2012 - 2017. Alguns direitos reservados. CNPJ: 23.811.568/0001-98
Praia de Botafogo, 300, 5º andar - Botafogo - Rio de Janeiro - RJ - 22250-040.
Principais referências utilizadas para a criação deste guia: CSS Tricks e MDN