Flexbox CSS – Guia completo – display: flex.

Modelo para desenvolver layouts CSS em um nível mais elevado

David Arty

Eae? Tudo bele?

Para quem está iniciando no desenvolvimento de interfaces web um dos maiores desafios é criar layouts. Isso porque se trata de um fundamento importante responsável por dar "cara" a interface e servir de "abrigo" para os conteúdos.

O que se espera de um profissional responsável por projetar layouts para aplicações e web sites? Você já pensou sobre isso? No mínimo que se conheça os principais métodos para layoutar e que com esses conhecimentos seja possível aplica-los em aspectos importantes do layout. Mas quais os principais métodos para isso?

No passado todo o processo de construção de layouts era concebido por tabelas HTML e recursos CSS como floats, posicionamentos, inline-blocks, entre outros, que podem ser considerados ultrapassados em relação a outros métodos que surgiram.

Um destes métodos mais modernos é um modelo conhecido como **Flexbox CSS** ou **Flexibe Box Layout** e chegou para layoutar de forma a não se apoiar em comportamentos inadequados e sem as limitações que eram encontradas nos métodos mais antigos.

Nesse artigo você vai conhecer as propriedades que envolvem este incrível método para que você possa a partir de hoje praticar e aplicar em seus projetos. Então vamos nessa?

Para começa vamos saber exatamente o que é Flexbox.

O que é Flexbox CSS?

Flexbox é um recurso CSS3 que serve de modelo para desenvolvimento de layouts para websites e aplicações web visando organizar elementos dentro de contêiners de forma flexível conforme sua necessidade.

E essa flexibilidade se caracteriza pela capacidade de alterar a largura e / ou a altura dos elementos (que são tratados como itens) para se adequarem ao espaço disponível em qualquer dispositivo de exibição. Um recipiente flexível expande os itens para preencher o espaço livre disponível ou encolhe-los para evitar o transbordamento.

Além disso, CSS Flexbox permite que se alinhem os itens horizontalmente e verticalmente, ordenando-os em diferentes posições no layout independente de como aparecem no documento HTML, e também permite que disponha os itens na horizontal (linhas) e na vertical (colunas).

Com essas características do Flexbox CSS com certeza você poderá construir layouts de forma dinâmica.

E antes de prosseguir com o conteúdo quero te informar que se você deseja **aprender Flexbox**, o Chief of Design oferece conteúdo em vídeo sobre essa ferramenta no **Curso em Vídeo de Fluência em HTML & CSS**. Dê uma conferida!!!

Flex Container e Flex itens

O que indica que um layout está utilizando CSS Flexbox é definição da propriedade **display** de um **contêiner**, o Flex Container, com o valor **flex**, ou inline-flex. Veja o código a seguir.

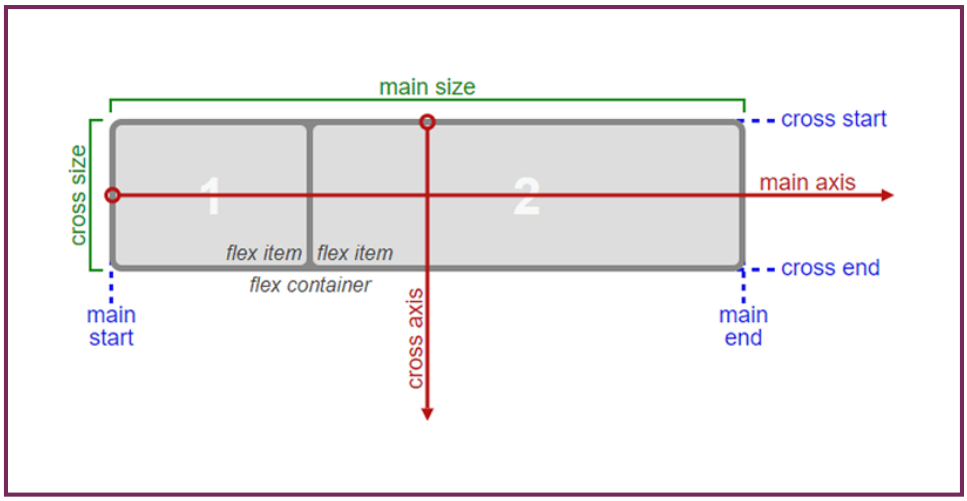
Código CSS:

```
.container {
     display: flex;
}
```

Mas mesmo que esta propriedade esteja definida não quer dizer que Flexbox funcionará. Ele não funciona apenas declarando CSS para um único elemento HTML. Deve existir uma relação entre um elemento pai, o contêiner que recebe o display: flex; e pelo menos um filho para que a mágica aconteça.

Quando você declara no elemento pai o display: flex você está convertendo todos os elementos filhos, os Flex Itens, para se comportarem como caixas flexíveis. Detalhe importante; o elemento pai não se torna flexível ao menos que ele seja filho de outro elemento que recebeu display: flex;.

Um Flex Container possui termos que definem os tamanhos, eixos e direções aplicados, no caso da imagem a seguir, a uma linha.



Existe uma propriedade em particular chamada flex-direction, e que a entenderemos mais a frente, que dispõe os Flex Itens de um Flex Container ou no eixo horizontal (Main Axis) ou no eixo vertical (Cross Axis).

Por padrão, Main Axis, que é o Eixo principal está na horizontal e faz com que os Flex Itens se aninhem um ao lado do outro em uma linha, enquanto Cross Axis é o eixo secundário. Porém isso pode ser invertido para que Main Axis fique na vertical e os Flex Itens se aninhem um abaixo do outro em uma coluna.

Por padrão, ou melhor dizendo, quando Main Axis está na horizontal, main-size corresponde a largura, mas se setarmos Main Axis para horizontal main-size será a altura.

E também por padrão, podemos dizer que Main Axis começa no ponto extremo à esquerda (main-start) e termina a direita (main-end) e Cross Axis começa no ponto extremo acima (cross-start) e termina abaixo (cross-end).

Propriedades para Flex Container

Vou agora apresentar uma a uma as propriedades Flexbox CSS que estilizam um elemento Flex Container:

display

A propriedade CSS display especificica como elementos HTML devem ser apresentados em relação ao tipo de caixa de renderização.

Com o valor “flex” ou “inline-flex” atribuído a esta propriedade, o contêiner e se torna flexível. Consequentemente seus filhos se tornam Flex Itens. Veja os códigos a seguir.

Código HTML:

```
<div class="container">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
```

Código CSS:

```
.container {
    display: flex; /* ou inline-flex */
    background-color: #cdf6eb;
```

```
    margin: 20px auto;
    max-width: 500px;
    font-family: sans-serif;
}
.container div {
    background: #028082;
    margin: 8px 4px;
    font-size: 1em;
    color: #fff;
}
```

A seguir podemos ver os resultados usando o JSFiddle para versão com **display: flex** e, logo em seguida com **display: inline-flex**. Para melhor entendimento você pode navegar entre as abas result, HTML e CSS para ver o aspecto visual e a codificação usada.

1 – display: flex:

Edit in JSFiddle

- Result
- HTML
- CSS

Item 1 Item 2 Item 3

```
<div class="container">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
```

2 – display: inline-flex:

Edit in JSFiddle

- Result
- HTML
- CSS

Item 1 Item 2 Item 3

```
<div class="container">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
```

flex-direction

Flex-direction é aplicada no contêiner, mas define o fluxo de exibição que os Flex Itens serão dispostos. É esta propriedade que pode mudar o Eixo principal da posição horizontal (Flex Itens aninhados lado a lado em linha) para na vertical (Flex Itens aninhados um abaixo do outro em coluna). Conheça os possíveis valores para esta propriedade:

- **row:** Este é o valor padrão, onde os itens são organizados para exibição em forma de linha da esquerda para a direita;
- **row-reverse:** Os itens também são organizados em linha, só que em ordem reversa em relação ao valor anterior. 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, só que iniciando de baixo para cima.

Vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>row</h4>
<div class="container row">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
<h4>row-reverse</h4>
<div class="container row-reverse">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
<h4>column</h4>
<div class="container column">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
<h4>column-reverse</h4>
<div class="container column-reverse">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
}
.row {
     flex-direction: row;
}
.row-reverse {
     flex-direction: row-reverse;
}
.column {
     flex-direction: column;
}
.column-reverse {
     flex-direction: column-reverse;
}
.container div {
    background: #028082;
```

```
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
.row div, .row-reverse div {
    margin: 8px 4px;
    width: 80px;
}
.column div, .column-reverse div {
    margin: 8px;
}
h4 {
    margin: 20px 0 0 20px;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver cada valor isoladamente no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

row

Item 1 Item 2 Item 3 Item 4

```
<h4>row</h4>
<div class="container row">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
<h4>row-reverse</h4>
<div class="container row-reverse">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
<h4>column</h4>
<div class="container column">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
<h4>column-reverse</h4>
<div class="container column-reverse">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
</div>
```

```
body {
  margin:0;
        padding: 0;
}
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
}
.row {
     flex-direction: row;
}
.row-reverse {
     flex-direction: row-reverse;
}
.column {
     flex-direction: column;
}
.column-reverse {
     flex-direction: column-reverse;
}
.container div {
    background: #028082;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as propriedades a partir daqui alinham o texto no centro */
    display: flex;
```

flex-wrap

A propriedade flex-wrap determina se um Flex Container é de linha única ou multilinhas. O que ocorre é que por padrão os Flex Itens tentarão se ajustar dentro de uma linha mesmo que ultrapassem visualmente a largura do elemento pai. O flex-wrap oferece a opção de quebra de linha para que os elementos filhos se ajustem adequadamente.

Conheça os possíveis valores para esta propriedade Flexbox:

- **nowrap:** Este é o valor padrão. Com ele todos os itens inseridos dentro do Flex Container serão dispostos em uma linha mesmo que ultrapasse a largura do contêiner;
- **wrap:** Ocorrerá a quebra de linha se alguns dos itens ultrapassar a largura do Flex Container. Os itens mais à direita serão deslocados para a linha de baixo;
- **wrap-reverse:** Também ocorrerá a quebra de linha se alguns dos itens ultrapassar a largura do Flex Container, só que neste caso os itens mais à direita serão deslocados para a linha de cima.

Vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>nowrap</h4>
<div class="container nowrap">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
</div>
<h4>wrap</h4>
<div class="container wrap">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
</div>
<h4>wrap-reverse</h4>
<div class="container wrap-reverse">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
</div>
```

Código CSS:

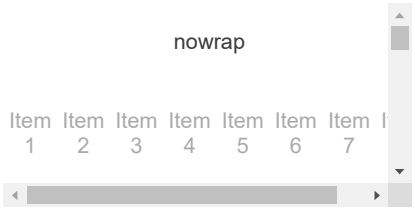
```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
}
.nowrap {
     flex-wrap: nowrap;
}
.wrap {
     flex-wrap: wrap;
}
.wrap-reverse {
     flex-wrap: wrap-reverse;
}
.container div {
```

```
    background: #028082;
    margin: 8px 4px;
    width: 80px;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
h4 {
    margin: 20px 0 0 20px;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver cada valor isoladamente no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS




```
<h4>nowrap</h4>
<div class="container nowrap">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
</div>
<h4>wrap</h4>
<div class="container wrap">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
</div>
<h4>wrap-reverse</h4>
```

flex-flow

A propriedade flex-flow é um propriedade de declaração única para escrita das propriedades flex-direction e flex-wrap.

Ao usar as propriedades flex-direction e flex-wrap usualmente a declaração fica da seguinte forma:

```
.container {
     display: flex;
     flex-direction: row;
     flex-wrap: wrap;
}
```

Agora a forma abreviada com flex-flow:

```
.container {
     display: flex;
     flex-flow: row wrap;
}
```

justify-content

Justify-content é uma propriedade que define o alinhamento dos Flex Itens ao longo do eixo principal do contêiner.

- **flex-start**: Esse é o valor padrão. Os Flex Itens são alinhados a partir do início do contêiner;
- **flex-end:** Os Flex Itens são alinhados a partir do fim do contêiner;
- **center:** Os itens são alinhados ao centro do contêiner;
- **space-between:** Cria um alinhamento uniforme entre os Flex Itens com um espaçamentos entre esses elementos. O primeiro item é deslocado para o início, e o último é deslocado para o final do contêiner;
- **space-around:** Os Flex Itens também são distribuídos ao longo contêiner, onde é criado um espaçamento ao redor dos elementos. Com isso, o primeiro e o último item possuirão margens e não ficaram grudados as extremidades do contêiner. Visualmente os espaçamentos antes do primeiro e depois do último item são menores que os outros espaçamentos. O que acontece é que os espaçamentos são aplicados no lado esquerdo e direito de cada Flex Item, portanto os espaçamentos que não estão nas extremidades tem tamanho dobrado porque soma o espaçamento a direita do item antecessor e o espaçamento a esquerda do item sucessor.
- **space-evenly:** Os Flex Itens também são distribuídos ao longo contêiner, onde também é criado um espaçamento ao redor dos elementos. O que difere do space-around é que os espaçamentos entre os itens e os espaçamentos nas extremidades do contêiner são distribuídos de forma igualitária.

Vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>flex-start</h4>
<div class="container flex-start">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>flex-end</h4>
<div class="container flex-end">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
```

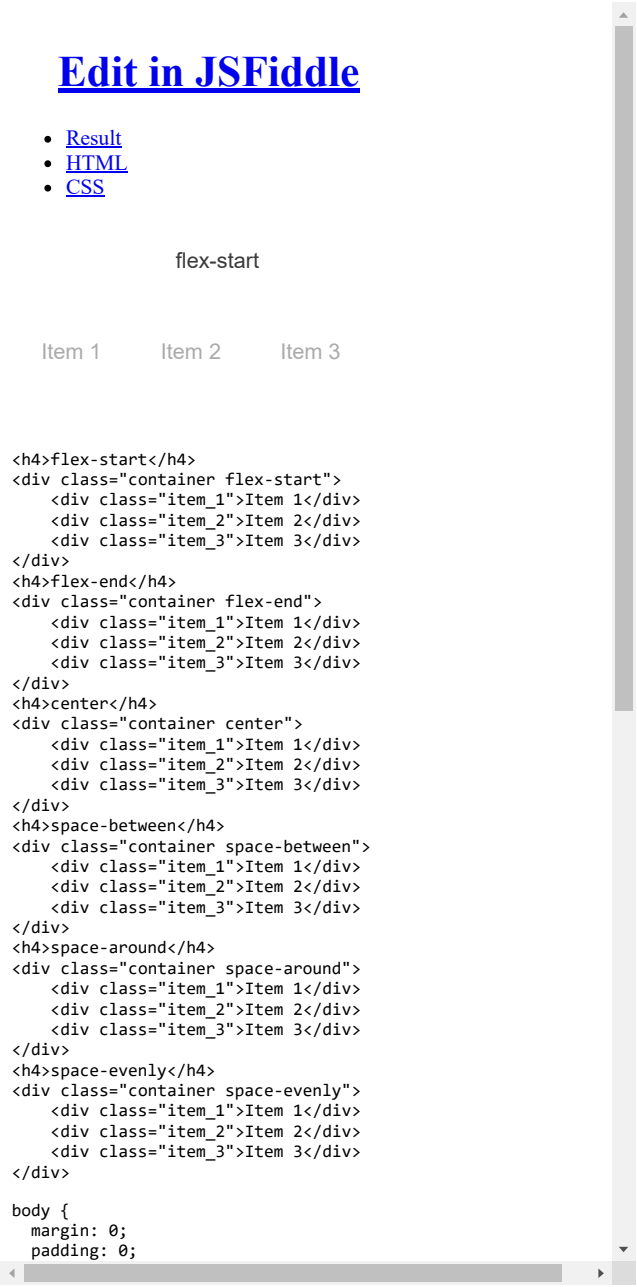
```
</div>
<h4>center</h4>
<div class="container center">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>space-between</h4>
<div class="container space-between">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>space-around</h4>
<div class="container space-around">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>space-evenly</h4>
<div class="container space-evenly">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
}
.flex-start {
     justify-content: flex-start;
}
.flex-end {
     justify-content: flex-end;
}
.center {
     justify-content: center;
}
.space-between {
     justify-content: space-between;
}
.space-around {
     justify-content: space-around;
}
.space-evenly {
     justify-content: space-evenly;
}
.container div {
    background: #028082;
    margin: 8px 4px;
    height: 80px;
    width: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
h4 {
    margin: 20px 0 0 20px;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
```

```
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver cada valor isoladamente no JSFiddle.



Observação: Com essas mesmas aplicações é possível verificar o comportamento dos Flex Itens mudando o Eixo Principal da posição horizontal para vertical através da propriedade **flex-direction** com o valor **column**.

align-items

Align-items é uma propriedade que define como os Flex Itens serão distribuídos ao longo do eixo transversal do contêiner. Conheça os possíveis valores para esta propriedade:

- **stretch:** Esse é o valor padrão. Neste caso os Flex Itens serão esticados para preencher toda a dimensão do eixo transversal igualmente;
- **flex-start:** Desloca os Flex Itens para o início do eixo transversal;

- **flex-end:** Desloca os Flex Itens para o final do eixo transversal;
- **center:** Os Flex Itens são centralizados no eixo transversal;
- **baseline:** Alinha os Flex Itens a partir da base da primeira linha de texto de cada um deles. Na aplicação abaixo você verá um exemplo onde os Flex Itens tem o mesmo tamanho de fonte o que não modifica o posicionamentos das caixas, também terá um exemplo com tamanhos diferentes de fontes e sem alinhando baseline e um terceiro também com tamanhos de fontes diferentes, mas com o alinhamento a partir de base da tipografia o que explicita melhor como funciona o valor baseline e mostra a mudança no posicionamento das caixas.

Vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>stretch</h4>
<div class="container stretch">
    <div class="item_1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>flex-start</h4>
<div class="container flex-start">
    <div class="item_1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>flex-end</h4>
<div class="container flex-end">
    <div class="item_1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>center</h4>
<div class="container center">
    <div class="item_1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>baseline e fontes do mesmo tamanho
Itens ficam alinhados pela base da tipografia</h4>
<div class="container baseline">
    <div class="item_1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>Sem baseline e fontes de tamanhos diferentes
Itens não ficam alinhados pela base da tipografia</h4>
<div class="container">
    <div class="item_1 fonte-1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2 fonte-2">Item 2</div>
    <div class="item_3 fonte-3">Item 3</div>
</div>
<h4>Com baseline e fontes de tamanhos diferentes
Itens ficam alinhados pela base da tipografia</h4>
<div class="container baseline">
    <div class="item_1 fonte-1">Este bloco que se inicia é correspondente ao Item 1</div>
    <div class="item_2 fonte-2">Item 2</div>
    <div class="item_3 fonte-3">Item 3</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 400px;
    font-family: sans-serif;
}
.stretch {
     align-items: stretch;
}
.flex-start {
     align-items: flex-start;
}
.flex-end {
     align-items: flex-end;
}
.center {
     align-items: center;
}
.baseline {
     align-items: baseline;
}
.container .fonte-1 {
  font-size: 0.6em;
}
.container .fonte-2 {
  font-size: 1.2em;
}
.container .fonte-3 {
  font-size: 0.8em;
}
.container div {
    background: #028082;
    margin: 8px 4px;
    width: 80px;
    font-size: 1em;
    color: #fff;
    flex: 1;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
h4 {
    margin: 20px 0 0 20px;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver cada valor isoladamente no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

stretch

Este bloco que se inicia é correspondente ao Item 1 | Item 2 | Item 3

```
<h4>stretch</h4>
<div class="container stretch">
    <div class="item_1">Este bloco que se inicia é corresponde
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>flex-start</h4>
<div class="container flex-start">
    <div class="item_1">Este bloco que se inicia é corresponde
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>flex-end</h4>
<div class="container flex-end">
    <div class="item_1">Este bloco que se inicia é corresponde
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>center</h4>
<div class="container center">
    <div class="item_1">Este bloco que se inicia é corresponde
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>baseline e fontes do mesmo tamanho<br>Itens ficam alinhado
<div class="container baseline">
    <div class="item_1">Este bloco que se inicia é corresponde
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
</div>
<h4>Sem baseline e fontes de tamanhos diferentes<br>Itens não
<div class="container">
    <div class="item_1 fonte-1">Este bloco que se inicia é cor
    <div class="item_2 fonte-2">Item 2</div>
    <div class="item_3 fonte-3">Item 3</div>
</div>
<h4>Com baseline e fontes de tamanhos diferentes<br>Itens fica
<div class="container baseline">
    <div class="item_1 fonte-1">Este bloco que se inicia é cor
    <div class="item_2 fonte-2">Item 2</div>
    <div class="item_3 fonte-3">Item 3</div>
</div>
```

```
bpdy {
  margin: 0;
  padding: 0;
}
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 400px;
    font-family: sans-serif;
}
.stretch {
     align-items: stretch;
}
.flex-start {
     align-items: flex-start;
}
.flex-end {
     align-items: flex-end;
}
.center {
     align-items: center;
}
.baseline {
     align-items: baseline;
}
.container .fonte-1 {
  font-size: 0.6em;
}
.container .fonte-2 {
  font-size: 1.2em;
}
.container .fonte-3 {
  font-size: 0.8em;
}
.container div {
```

```
    background: #028082;
    margin: 8px 4px;
    width: 80px;
    font-size: 1em;
    color: #fff;
    flex: 1;
    /* as proriedades a partir daqui alinham o texto no centro
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
```

Observação: Com essas mesmas aplicações é possível verificar o comportamento dos Flex Itens mudando o Eixo Principal da posição horizontal para vertical através da propriedade **flex-direction** com o valor **column**.

align-content

Align-content é uma propriedade que define como as linhas são distribuídas ao longo do eixo transversal do contêiner. Como está propriedade trabalha com a distribuição das linha, ele é a aplicada quando o Flex Container é multilinhas, ou seja, recebe **flex-wrap** com o valor **wrap**. Conheça os possíveis valores para esta propriedade:

- **stretch:** Este é o valor padrão. As linhas são distribuídas uniformemente ao longo do eixo transversal.
- **flex-start:** Distribui as linhas a partir do início do eixo transversal;
- **flex-end:** Distribui as linhas a partir do fim do eixo transversal;
- **center:** Mantém as linhas no centro do eixo transversal;
- **space-between:** Cria um espaçamento entre as linhas. Onde a primeira linha é deslocada para o início do eixo transversal, a última é deslocada para o final do eixo transversal;
- **space-around:** As linhas são uniformemente distribuídos ao longo do eixo transversal, onde é criado um espaçamento ao redor delas. Com isso, a primeira e o última linha possuirão margens e não ficaram grudadas as extremidades do contêiner.

Vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>stretch</h4>
<div class="container stretch">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>flex-start</h4>
<div class="container flex-start">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
```

```
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>flex-end</h4>
<div class="container flex-end">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>center</h4>
<div class="container center">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>space-between</h4>
<div class="container space-between">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>space-around</h4>
<div class="container space-around">
```

```
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    height: 300px;
    font-family: sans-serif;
    flex-wrap: wrap;
}
.stretch {
     align-content: stretch;
}
.flex-start {
     align-content: flex-start;
}
.flex-end {
     align-content: flex-end;
}
.center {
     align-content: center;
}
.space-between {
     align-content: space-between;
}
.space-around {
     align-content: space-around;
}
.container div {
    background: #028082;
    margin: 8px 4px;
    font-size: 1em;
    color: #fff;
}
h4 {
    margin: 20px 0 0 20px;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver cada valor isoladamente no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

stretch

Item 1 Item 2 Item 3 Item 4 Item 5

Item 6 Item 7 Item 8 Item 9

```
<h4>stretch</h4>
<div class="container stretch">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>flex-start</h4>
<div class="container flex-start">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>flex-end</h4>
<div class="container flex-end">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>center</h4>
<div class="container center">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>space-between</h4>
<div class="container space-between">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
```

```
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
<h4>space-around</h4>
<div class="container space-around">
    <div class="item_1">Item 1</div>
    <div class="item_2">Item 2</div>
    <div class="item_3">Item 3</div>
    <div class="item_4">Item 4</div>
    <div class="item_5">Item 5</div>
    <div class="item_6">Item 6</div>
    <div class="item_7">Item 7</div>
    <div class="item_8">Item 8</div>
    <div class="item_9">Item 9</div>
    <div class="item_10">Item 10</div>
    <div class="item_11">Item 11</div>
    <div class="item_12">Item 12</div>
    <div class="item_13">Item 13</div>
    <div class="item_14">Item 14</div>
    <div class="item_15">Item 15</div>
    <div class="item_16">Item 16</div>
</div>
```

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    height: 300px;
    font-family: sans-serif;
    flex-wrap: wrap;
}
.stretch {
     align-content: stretch;
}
 flex-start {
```

Observação: Com essas mesmas aplicações é possível verificar o comportamento dos Flex Itens mudando o Eixo Principal da posição horizontal para vertical através da propriedade **flex-direction** com o valor **column**.

Propriedades para Flex Itens

Vou agora apresentar uma a uma as propriedades que estilizam Flex Itens:

flex-grow

A propriedade flex-grow serve para especificar o fator de crescimento que um Flex Item terá em relação aos outros Flex Itens encontrados em um Flex Container. O valor especificado deve ser um número positivo. Por padrão o valor de flex-grow é 0, ou seja, os Flex Itens não crescem.

Quanto maior o valor de flex-grow, mais o Flex Item poderá crescer relativamente aos outros items. Para entender melhor imagine que você possui em uma linha de um contêiner 3 Flex Itens. Dois deles foram determinados flex-grow: 1, e um flex-grow: 2. O flex-grow: 2 tentará ocupar duas vezes mais espaço do espaço disponível que os outros itens.

Uma coisa que não devemos nos confundir é sobre a dimensão do quanto de espaço que um Flex Item tentará ocupar. Não é porque um flex-grow tenha um fator de crescimento 2 que ele terá o dobro de ocupação. Existe um cálculo envolvendo este fator mais o espaço disponível que determina quanto de largura, caso flex-direction: row, ou altura, caso flex-direction: column, um Flex Item deve ter.

Imagine o exemplo anterior onde um Flex Item tem o flex-grow: 2 e os outros dois tem flex-grow: 1. Vamos estipular para esses três Flex Itens uma largura de 100 pixels **(flex-basis: 100px;)** onde o contêiner, elemento pai destes itens tem uma largura (width) de 500 pixels.

O flex-grow quando possui valor acima de 0 faz com que os Flex Itens cresçam (se flex-wrap: wrap, não estiver declarado) para ocupar a largura do contêiner. Ou seja, os três Flex Itens, mesmo com a somas das larguras declaradas dando 300 pixels se distribuem para ocupar a largura de 500 pixels do contêiner. E aí que entra o cálculo da unidade de crescimento.

Observação: flex-basis, citado no parágrafo anterior, será melhor explicado no próximo tópico. Enquanto isso basta saber que esta propriedade define o tamanho inicial (largura ou altura) que um Flex Item deve ter antes que o espaço ao seu redor seja distribuído por outras propriedades.

Unidade de crescimento = espaço disponível / soma dos fatores de crescimento de todos itens

Sendo que, espaço disponível é o valor da largura do Flex Container (500 pixels) menos a soma das largura dos Flex Itens (300 pixels). Ou seja 200 pixels. E a soma dos fatores de crescimento dos Flex Itens é 4. Agora aplicando na fórmula:

Unidade de crescimento = 200px / 4 = 50px.

Agora para saber quanto cada Flex Item cresceu basta multiplicar o fator de crescimento por 50 pixels e somar pela largura (flex-basis: 100px;).

Resultado:

1 – Flex Itens com flex-grow: 1: (50px x 1) + 100px = 150 pixels.
2 – Flex Item com flex-grow: 2: (50px x 2) + 100px = 200 pixels.

Agora vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>flex-grow: 0;</h4>
<div class="container">
    <div class="grow_0">Item 1</div>
    <div class="grow_0">Item 2</div>
    <div class="grow_0">Item 3</div>
    <div class="grow_0">Item 4</div>
</div>
<h4>flex-grow: 1;</h4>
<div class="container">
    <div class="grow_1">Item 1</div>
    <div class="grow_1">Item 2</div>
    <div class="grow_1">Item 3</div>
    <div class="grow_1">Item 4</div>
</div>
<h4>flex-grow: 0; e flex-grow: 2;</h4>
<div class="container">
    <div class="grow_0">Item 1</div>
    <div class="grow_2">Item 2</div>
    <div class="grow_0">Item 3</div>
    <div class="grow_0">Item 4</div>
</div>
<h4>flex-grow: 0; | flex-grow: 2 | flex-basis: 100px ;</h4>
<div class="container">
    <div class="grow_1 basis">Item 1</div>
    <div class="grow_2 basis">Item 2</div>
    <div class="grow_1 basis">Item 3</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
    flex-wrap; wrap;
}
.grow_0 {
    flex-grow: 0;
}
.grow_1 {
    flex-grow: 1;
}
.grow_2 {
    flex-grow: 2;
}

.container div {
    background: #028082;
    margin: 8px 4px;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}

h4 {
    margin: 20px 0 0 20px;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver os resultados no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

flex-grow: 0;

Item 1 Item 2 Item 3 Item 4

```
<h4>flex-grow: 0;</h4>
<div class="container">
    <div class="grow_0">Item 1</div>
    <div class="grow_0">Item 2</div>
    <div class="grow_0">Item 3</div>
    <div class="grow_0">Item 4</div>
</div>
<h4>flex-grow: 1;</h4>
<div class="container">
    <div class="grow_1">Item 1</div>
    <div class="grow_1">Item 2</div>
    <div class="grow_1">Item 3</div>
    <div class="grow_1">Item 4</div>
</div>
<h4>flex-grow: 0; e flex-grow: 2;</h4>
<div class="container">
    <div class="grow_0">Item 1</div>
    <div class="grow_2">Item 2</div>
    <div class="grow_0">Item 3</div>
    <div class="grow_0">Item 4</div>
</div>
```

flex-basis

Como já citado anteriormente, flex-basis define o tamanho inicial que um Flex Item deve ter antes que o espaço ao seu redor seja distribuído por outras propriedades. No caso flex-grow e flex-shrink, está útltima ainda a ser retratada neste artigo.

Quando o eixo principal for horizontal, esta propriedade define a largura mínima antes que espaço restante seja distribuído, quando for vertical defina e altura mínima.

Por padrão seu valor é auto (flex-basis: auto) que quer dizer o tamanho da largura (ou altura) do Flex Item. Caso se para esse Flex Item não for determinado um tamanho de width, e caso o eixo principal for horizontal, ou height, caso vertical, então flex-basis: auto; equivalerá ao tamanho do conteúdo.

Além do valor auto, esta propriedade também pode receber valores para dimensões, como pixels e porcentagens.

Observação: Ao utilizar o valor **auto** para flex-basis é possível alterar tamanho de um Flex Item (Largura, caso eixo principal na horizontal e altura, caso eixo principal na vertical) quando conjuntamente usamos **width** e **height**. Mas somente nesta situação. Quando você define um valor diferente de auto qualquer declaração com width e height perde o efeito.

Agora vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>flex-basis: auto (flex-grow: 1; procurará distribuir os itens para ocupar o espaço do
<div class="container">
    <div class="grow_1 basis_auto">Item 1</div>
    <div class="grow_1 basis_auto">Item 2</div>
```

```
    <div class="grow_1 basis_auto">Item 3</div>
    <div class="grow_1 basis_auto">Item 4</div>
</div>
<h4>flex-basis: auto (Com flex-grow: 0; os itens não ocuparão o espaço do contêiner e a l
<div class="container">
    <div class="grow_0 basis_auto">Item 1</div>
    <div class="grow_0 basis_auto">Item 2</div>
    <div class="grow_0 basis_auto">Item 3</div>
    <div class="grow_0 basis_auto">Item 4</div>
</div>
<h4>flex-basis: auto (Com flex-grow: 0 e largura definida pela propriedade width)</h4>
<div class="container">
    <div class="grow_0 basis_auto largura">Item 1</div>
    <div class="grow_0 basis_auto largura">Item 2</div>
    <div class="grow_0 basis_auto largura">Item 3</div>
    <div class="grow_0 basis_auto largura">Item 4</div>
</div>
<h4>flex-basis: 125px;</h4>
<div class="container">
    <div class="grow_1 basis_125">Item 1</div>
    <div class="grow_1 basis_125">Item 2</div>
    <div class="grow_1 basis_125">Item 3</div>
    <div class="grow_1 basis_125">Item 4</div>
</div>
<h4>flex-basis: 100px (A somatória da largura dos Flex Itens é igual a 400 pixels. Mas me
<div class="container">
    <div class="grow_1 basis_100">Item 1</div>
    <div class="grow_1 basis_100">Item 2</div>
    <div class="grow_1 basis_100">Item 3</div>
    <div class="grow_1 basis_100">Item 4</div>
</div>
<h4>flex-basis: 100px (Com flex-grow: 0; os itens não ocuparão o espaço do contêiner e a
<div class="container">
    <div class="grow_0 basis_100">Item 1</div>
    <div class="grow_0 basis_100">Item 2</div>
    <div class="grow_0 basis_100">Item 3</div>
    <div class="grow_0 basis_100">Item 4</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
    flex-wrap; wrap;
}
.grow_0 {
    flex-grow: 0;
}
.grow_1 {
    flex-grow: 1;
}
.basis_auto {
    flex-basis: auto;
}
.basis_100 {
    flex-basis: 100px;
}
.basis_125 {
    flex-basis: 125px;
}
.largura {
    width: 80px;
}
.container div {
    background: #028082;
```

```
    margin: 8px 4px;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
h4 {
    margin: 20px 0 0 20px;
    padding: 0 6%;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver os resultados no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

flex-basis: auto (flex-grow: 1; procurará distribuir os itens para ocupar o espaço do container);

Item 1 Item 2 Item 3 Item 4

```
<h4>flex-basis: auto (flex-grow: 1; procurará distribuir os itens para ocupar o espaço do contai
<div class="container">
    <div class="grow_1 basis_auto">Item 1</div>
    <div class="grow_1 basis_auto">Item 2</div>
    <div class="grow_1 basis_auto">Item 3</div>
    <div class="grow_1 basis_auto">Item 4</div>
</div>
<h4>flex-basis: auto (Com flex-grow: 0; os itens não ocuparão o espaço do container e a largura
<div class="container">
    <div class="grow_0 basis_auto">Item 1</div>
    <div class="grow_0 basis_auto">Item 2</div>
    <div class="grow_0 basis_auto">Item 3</div>
    <div class="grow_0 basis_auto">Item 4</div>
</div>
<h4>flex-basis: auto (Com flex-grow: 0 e largura definida pela propriedade width)</h4>
<div class="container">
    <div class="grow_0 basis_auto largura">Item 1</div>
    <div class="grow_0 basis_auto largura">Item 2</div>
    <div class="grow_0 basis_auto largura">Item 3</div>
    <div class="grow_0 basis_auto largura">Item 4</div>
</div>
<h4>flex-basis: 125px;</h4>
<div class="container">
    <div class="grow_1 basis_125">Item 1</div>
    <div class="grow_1 basis_125">Item 2</div>
    <div class="grow_1 basis_125">Item 3</div>
    <div class="grow_1 basis_125">Item 4</div>
</div>
<h4>flex-basis: 100px (A somatória da largura dos Flex Itens é igual a 400 pixels. Mas mesmo ass
<div class="container">
    <div class="grow_1 basis_100">Item 1</div>
    <div class="grow_1 basis_100">Item 2</div>
    <div class="grow_1 basis_100">Item 3</div>
    <div class="grow_1 basis_100">Item 4</div>
</div>
<h4>flex-basis: 100px (Com flex-grow: 0; os itens não ocuparão o espaço do container e a largura
<div class="container">
    <div class="grow_0 basis_100">Item 1</div>
    <div class="grow_0 basis_100">Item 2</div>
    <div class="grow_0 basis_100">Item 3</div>
    <div class="grow_0 basis_100">Item 4</div>
</div>

.container {
    display: flex;
```

flex-shrink

Enquanto no flex-grow define a capacidade dos Flex Itens de crescerem o flex-shrink faz exatamente o contrário ele define a redução, ou seja, ele serve para especificar o fator de encolhimento que um Flex Item terá em relação aos outros Flex Itens encontrados em um Flex Container.

Por padrão o valor de flex-shrink é 1, ou seja, os Flex Itens reduzem de tamanho para caber dentro do espaço disponível no contêiner.

Quanto maior o valor de flex-shrink, mais o Flex Item poderá reduzir de tamanho relativamente aos outros items.

E em qual situação o flex-shrink reduz o tamanho dos Flex Itens? Quando o espaço disponível é negativo, ou seja, quando a somatória das larguras (ou alturas, se o eixo principal for na vertical) dos Flex Itens ultrapassar o tamanho do Flex Container.

Com isso deduzimos que também existe uma fórmula para sabermos em quanto os Flex Itens são reduzidos para caber no container.

Unidade de encolhimento = espaço disponível / soma dos fatores de encolhimento de todos itens

Sendo que, espaço disponível é o valor da largura do Flex Container menos a soma das largura dos Flex Itens. O que resulta em um valor negativo.

Vamos imaginar a seguinte situação. Temos um Flex Container com 500 pixels enquanto a largura dos Flex Itens somadas dá 800 pixels (flex-basis: 200px; para cada um de quatro itens) e a soma dos fatores de encolhimento de todos eles é 5 (um deles possui fator 2 e os restantes fator 1). Vamos transpor esses valore à formula.

Unidade de encolhimento = (500px – 800px) / 5 = -60 pixels;

Agora para saber quanto cada Flex Item encolheu basta multiplicar o fator de encolhimento por -60 pixels e somar pela largura (flex-basis: 100px;).

Resultado:

1 – Flex Itens com flex-shrink: 1: (-60px x 1) + 200px = 140 pixels.
2 – Flex Item com flex-shrink: 2: (-60px x 2) + 200px = 80 pixels.

Agora vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>flex-shrink com valor maior que zero define que Flex Itens devem encolher para caber
<div class="container">
    <div class="shrink_1 basis">Item 1</div>
    <div class="shrink_2 basis">Item 2</div>
    <div class="shrink_1 basis">Item 3</div>
    <div class="shrink_1 basis">Item 4</div>
</div>
<h4>flex-shrink: 0; (Flex Itens não encolhem)</h4>
<div class="container">
    <div class="shrink_0 basis">Item 1</div>
    <div class="shrink_0 basis">Item 2</div>
    <div class="shrink_0 basis">Item 3</div>
    <div class="shrink_0 basis">Item 4</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
    flex-wrap; wrap;
}
.shrink_0 {
    flex-shrink: 0;
}
.shrink_1 {
    flex-shrink: 1;
}
.shrink_2 {
```

```
    flex-shrink: 2;
}
.basis {
    flex-basis: 200px;
}
.container div {
    background: #028082;
    margin: 8px 4px;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
h4 {
    margin: 20px 0 0 20px;
    padding: 0 6%;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver os resultados no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

flex-shrink com valor maior que zero define que Flex Itens devem encolher para caber no container

```
<h4>flex-shrink com valor maior que zero define que Flex Itens devem encolher para caber no cont
<div class="container">
    <div class="shrink_1 basis">Item 1</div>
    <div class="shrink_2 basis">Item 2</div>
    <div class="shrink_1 basis">Item 3</div>
    <div class="shrink_1 basis">Item 4</div>
</div>
<h4>flex-shrink: 0; (Flex Itens não encolhem)</h4>
<div class="container">
    <div class="shrink_0 basis">Item 1</div>
    <div class="shrink_0 basis">Item 2</div>
    <div class="shrink_0 basis">Item 3</div>
    <div class="shrink_0 basis">Item 4</div>
</div>

.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
    flex-wrap; wrap;
}
.shrink_0 {
    flex-shrink: 0;
```

flex

Esta é a abreviatura para flex-grow, flex-shrink, flex-basis combinados nesta sequencia. O padrão é flex: 0 1 auto;. O uso da abreviatura é recomendada em vez de definir as propriedades separadamente.

```
.item {
  flex: 0 1 auto;
}
```

A declaração acima faz do flex item inflexível, quando há espaço disponível no contêiner, mas permite encolher quando há espaço insuficiente.

order

O padrão faz com que os Flex Itens aparecem no contêiner na ordem que são inseridos no HTML. Mas essa ordem pode ser alterada através da propriedade **order** sendo que se inicia de uma valor menor para o maior. O valor inicial de order é 0 e também é possível especificar valores negativos.

Agora vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>order - exemplo 1</h4>
<div class="container row">
    <div class="grow_1 order_3">Item 1</div>
    <div class="grow_1">Item 2</div>
    <div class="grow_1 order_4">Item 3</div>
    <div class="grow_1 order_2">Item 4</div>
</div>
<h4>order - exemplo 2</h4>
<div class="container row">
    <div class="grow_1 order_2">Item 1</div>
    <div class="grow_1 order_4">Item 2</div>
    <div class="grow_1 order_2">Item 3</div>
    <div class="grow_1 order_menosUm">Item 4</div>
</div>
<h4>order - exemplo 3</h4>
<div class="container column">
    <div class="grow_1 order_2">Item 1</div>
    <div class="grow_1">Item 2</div>
    <div class="grow_1 order_4">Item 3</div>
    <div class="grow_1 order_2">Item 4</div>
</div>
<h4>order - exemplo 3</h4>
<div class="container column">
    <div class="grow_1 order_3">Item 1</div>
    <div class="grow_1 order_4">Item 2</div>
    <div class="grow_1 order_2">Item 3</div>
    <div class="grow_1 order_menosUm">Item 4</div>
</div>
```

Código CSS:

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
    flex-wrap; wrap;
}
```

```
.order_menosUm {
   order: -1;
}
.order_1 {
   order: 1;
}
.order_2 {
   order: 2;
}
.order_3 {
   order: 3;
}
.order_4 {
   order: 4;
}
.grow_1 {
   flex-grow: 1
}
.column {
    flex-direction: column;
}
.container div {
    background: #028082;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as propriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
.row div {
    margin: 8px 4px;
    width: 80px;
}
.column div {
    margin: 8px;
}
h4 {
    margin: 20px 0 0 20px;
    padding: 0 2%;
    font-family: sans-serif;
    font-weight: normal;
    font-size: 1em;
    color: #3b3b3b;
    text-align: center;
}
```

Agora vamos ver os resultados no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

order - exemplo 1

Item 2 Item 4 Item 1 Item 3

```
<h4>order - exemplo 1</h4>
<div class="container row">
    <div class="grow_1 order_3">Item 1</div>
    <div class="grow_1">Item 2</div>
    <div class="grow_1 order_4">Item 3</div>
    <div class="grow_1 order_2">Item 4</div>
</div>
<h4>order - exemplo 2</h4>
<div class="container row">
    <div class="grow_1 order_2">Item 1</div>
    <div class="grow_1 order_4">Item 2</div>
    <div class="grow_1 order_2">Item 3</div>
    <div class="grow_1 order_menosUm">Item 4</div>
</div>
<h4>order - exemplo 3</h4>
<div class="container column">
    <div class="grow_1 order_2">Item 1</div>
    <div class="grow_1">Item 2</div>
    <div class="grow_1 order_4">Item 3</div>
    <div class="grow_1 order_2">Item 4</div>
</div>
<h4>order - exemplo 3</h4>
<div class="container column">
    <div class="grow_1 order_3">Item 1</div>
    <div class="grow_1 order_4">Item 2</div>
    <div class="grow_1 order_2">Item 3</div>
    <div class="grow_1 order_menosUm">Item 4</div>
</div>
```

```
.container {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 500px;
    font-family: sans-serif;
    flex-wrap; wrap;
}
.order_menosUm {
   order: -1;
}
.order_1 {
   order: 1;
}
.order_2 {
   order: 2;
}
.order_3 {
   order: 3;
}
.order_4 {
   order: 4;
}
.grow_1 {
   flex-grow: 1
}
.column {
    flex-direction: column;
}
.container div {
    background: #028082;
    height: 80px;
    font-size: 1em;
    color: #fff;
    /* as proriedades a partir daqui alinham o texto no centro */
    display: flex;
    text-align: center;
    justify-content: center;
    align-items: center;
}
.row div {
    margin: 8px 4px;
    width: 80px;
}
.column div {
```

align-self

Esta propriedade permite definir um alinhamento de um único Flex Item dentro de um contêiner sobrescrevendo o que foi definido no align-itens do Flex Container. Conheça os possíveis valores para esta propriedade:

- **auto:** Este é o valor padrão. Com ele declarado o comportamento definido no container por meio do align-items é respeitado.
- **stretch:** Neste caso o item será esticado para preencher toda a dimensão do eixo transversal (largura ou altura) igualmente;
- **flex-start:** Desloca o item para o início do eixo transversal;
- **flex-end:** Desloca o item para o final do eixo transversal;
- **center:** O item é centralizado no eixo transversal;
- **baseline:** Alinha o item a partir da base da primeira linha de texto dos demais itens.

Vamos a alguns exemplos de aplicação em códigos.

Código HTML:

```
<h4>Todo os valores de align-self + algin-items: flex-end</h4>
<div class="container1 align-items_flex-end">
    <div class="item_1">Item 1</div>
    <div class="item_2 stretch">Item 2</div>
    <div class="item_3 flex-start">Item 3</div>
    <div class="item_4 flex-end">Item 4</div>
    <div class="item_4 center">Item 5</div>
    <div class="item_6 baseline">Item 6</div>
</div>
<h4>Todo os valores de align-self + algin-items: flex-end</h4>
<div class="container2 align-items_flex-end column">
    <div class="item_1">Item 1</div>
    <div class="item_2 stretch">Item 2</div>
    <div class="item_3 flex-start">Item 3</div>
    <div class="item_4 flex-end">Item 4</div>
    <div class="item_4 center">Item 5</div>
    <div class="item_6 baseline">Item 6</div>
</div>
```

Código CSS:

```
.container1, .container2 {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 400px;
    font-family: sans-serif;
}
.container1 {
    height: 100px;
}
.stretch {
     align-self: stretch;
}
.flex-start {
     align-self: flex-start;
}
.flex-end {
     align-self: flex-end;
}
.center {
     align-self: center;
}
.baseline {
```

```
      align-self: baseline;
  }
  .align-items_flex-end {
       align-items: flex-end;
  }
  .column {
       flex-direction: column;
  }
  .container1 div, .container2 div {
      background: #028082;
      margin: 4px;
      font-size: 1em;
      color: #fff;
  }
  h4 {
      margin: 20px 0 0 20px;
      font-family: sans-serif;
      font-weight: normal;
      font-size: 1em;
      color: #3b3b3b;
      text-align: center;
  }
```

Agora vamos ver os resultados no JSFiddle.

Edit in JSFiddle

- Result
- HTML
- CSS

Todo os valores de align-self + algin-items: flex-end

Item 2 Item 3 Item 5 Item 6 Item Item

```
<h4>Todo os valores de align-self + algin-items: flex-end</h4>
<div class="container1 align-items_flex-end">
    <div class="item_1">Item 1</div>
    <div class="item_2 stretch">Item 2</div>
    <div class="item_3 flex-start">Item 3</div>
    <div class="item_4 flex-end">Item 4</div>
    <div class="item_4 center">Item 5</div>
    <div class="item_6 baseline">Item 6</div>
</div>
<h4>Todo os valores de align-self + algin-items: flex-end</h4>
<div class="container2 align-items_flex-end column">
    <div class="item_1">Item 1</div>
    <div class="item_2 stretch">Item 2</div>
    <div class="item_3 flex-start">Item 3</div>
    <div class="item_4 flex-end">Item 4</div>
    <div class="item_4 center">Item 5</div>
    <div class="item_6 baseline">Item 6</div>
</div>

.container1, .container2 {
    display: flex;
    background-color: #cdf6eb;
    margin: 10px auto 30px;
    max-width: 300px;
    font-family: sans-serif;
}
.container1 {
    height: 100px;
}
.stretch {
     align-self: stretch;
}
.flex-start {
     align-self: flex-start;
}
.flex-end {
```

Exemplo de um Layout Básico feito com Flexbox CSS

Edit in JSFiddle

- Result
- HTML
- CSS

Lorem Ipsum?

- Home
- Sobre

```
<section class="container">
        <header class="top box"><h1>Lorem Ipsum?</h1></header>
        <nav class="mainnav box">
                <ul>
                        <li><a href="">Home</a></li>
                        <li><a href="">Sobre</a></li>
                        <li><a href="">Contato</a></li>
                </ul>
        </nav>
        <article class="wrapper">
                <header class="title box">
                        <h2>Definição</h2>
                </header>
                <div class="conteudo box">
                  <h3>O que é Lorem Ipsum?</h3>
                        <p>Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográf
                        <p>Lorem Ipsum sobreviveu não só a cinco séculos, como também ao salto par
                </div>
                <aside class="complementar box">
                        <h3>Porque nós o usamos?</h3>
                        <p>A vantagem de usar Lorem Ipsum é que ele tem uma distribuição normal de
                </aside>
                </article>
                <footer class="rodape box"><p>Todos os direitos reservados</p></footer>
</section>
```

```
* {
        box-sizing: border-box;
}
body {
        background-color: #382232;
        font-family: Arial, sans-serif;
        padding: 0;
        margin: 0;
}
body , a {
        color: #fff;
}
.container {
        display: flex;
        flex-wrap: wrap;
        width: 95%;
        margin: 0 auto;
        border: 5px solid #fff;
}
.box {
        padding: 2% 5%;
        border: 5px solid #fff;
}
.top {
        flex: 1 0 100%;
}
.mainnav {
        flex: 1 0 100%;
}
.wrapper {
        flex: 1 0 70%;
        display: flex;
        flex-wrap: wrap;
}
.title {
        flex: 1 0 100%;
}
.conteudo {
        flex: 1 0 70%;
}
.complementar {
        flex: 1 0 27%;
        margin-right: 0;
}
.top, .mainnav, .rodape {
        background-color: #01d29e;
}
.title, .conteudo, .complementar {
        background-color:#267fed;
}
.rodape {
        flex: 1 0 100%;
}
@media screen and (min-width: 585px){
        .mainnav {
```

```
        flex: 1 0 30%;
        }
}
```

Neste layout básico as caixas são posicionadas utilizando a propriedade **flex** que determina entre tantos valores a largura destas caixas.

Em conjunto com a propriedade **flex-wrap** de valor **wrap** os Flex Itens, que naturalmente ficariam lado a lado em uma só linha, são distribuídos em outras linhas conforme o espaço disponível já esteja ocupado. E é neste momento em que a “mágica” começa a acontecer.

Sabendo que alguns itens devem ocupar 100% de uma linha, como é o caso do topo e do rodapé, definimos uma largura de 100% para eles e os outros itens vão automaticamente para as próximas linhas. E no caso necessitamos de dois ou mais itens em uma mesma linha, caso dos itens **.wrapper** e **.mainnav** para Desktop, dividimos o espaço disponível de 100% entre os itens desta linha e conforme a largura individual de cada item estipulada no projeto.

Bônus

Interessante que você dedique um tempo praticando para absorver bem estas informações sobre Flexbox CSS aí no seu editor favorito. Entretanto quero te passar uma dica bônus que vai garantir divertimento enquanto você aprende a manipular as propriedades do Flexbox. Me refiro ao **Flexbox Froggy**!!! Um pequeno game online onde você ajuda um sapo e seus amigos dando a eles comandos para execução de tarefas através de propriedades CSS do Flexbox.

O game contém 24 níveis onde você poderá praticar as principais propriedade do Flexbox.

Conclusão sobre Flexbox CSS

Podemos concluir que o uso de CSS Flexbox aperfeiçoa ainda mais a forma como desenvolvemos layouts por causa de suas características avançadas de como manipular os itens.

O próximo passo após a leitura deste artigo e praticar cada propriedade mencionada. Até aqui tivemos uma pincelada sobre aspectos introdutórios deste método. Para uma pesquisa mais avançada você pode recorrer a **documentação W3C**.

Uma observação importante: O mundo do desenvolvimento web sempre avança e sempre tem coisa nova. Quando falamos de CSS então sabemos que ele teveum avanço significativo nos últimos tempos. Tanto que quando falamos de layouts temos um recurso semelhante e mais atual ao Flexbox CSS chamado CSS Grid Layout e que possui um **artigo aqui no blog que sugiro que leia**. E porque estou citando ele?

Porque rapidamente é importante que você saiba que ambos trabalham com a construção de layout e que existe uma diferença fundamental entre eles.

O Flexbox CSS destina-se a layouts unidimensionais mais simples que possam ser dispostos em linha reta. É mais apropriado para os componentes de um aplicativo e layouts em pequena escala.

Já o CSS Grid Layout destina-se a layouts bidimensionais mais complexos. É adequado para organizar a estrutura de layout de elementos de nível mais alto, como cabeçalhos, rodapés, sidebar e seções.

Tendo isso esclarecido, agora me diga o que achou do Flexbox CSS. Já utilizou alguma vez em seus projetos? Deixe sua opinião e aproveite e compartilhe sobre esse assunto com seus amigos!

Forte abraço.

Até mais.

Referências:

https://www.youtube.com/watch?v=_YUJ37FARrU

https://www.youtube.com/watch?v=v4GIqmPiLmo

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

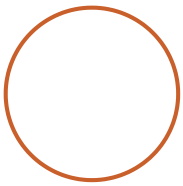
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

http://desenvolvimentoparaweb.com/css/flexbox/

https://www.devmedia.com.br/css3-flexbox-funcionamento-e-propriedades/29532

Imagem:

https://www.w3.org/TR/css-flexbox-1/images/flex-direction-terms.svg



David Arty

Olá. Sou David Arty, fundador do blog Chief of Design. Sou natural de São Paulo, Brasil. Trabalho com design, principalmente com design para web, desde 2009. Procuro transformar ideias loucas e complexas em peças simples, atrativas e funcionais.

Saiba mais sobre mim

Artigos Relacionados

35 Ferramentas para FrontEnd e Web Designer

O que são meta tags? Para que servem as meta tags?

Livro Redescubra a Fotografia – 25%

Críticas no Design: Designer criticando outro Designer? Pode isso?

3 Comentários Entrar ▾

Participe da discussão...

FAZER LOGIN COM OU REGISTRE-SE NO DISQUS

Nome

1 **Compartilhar** Mais votados Mais recentes Mais antigos

Bruno Cessel
3 anos atrás

David, ótima tarde! Eu tenho uma dúvida muito específica envolvendo Flex e tem me assombrado em alguns projetos. Estou recebendo muitos projetos em que criam-se colunas distintas de cidades com lista de lojas. Consigo alinhá-las lado a lado com flex: 1; e tentando adicionar flex-wrap: wrap; Porém ao acrescentar diversas lojas na cidade 1 (bloco 1), ele cresce verticalmente, mantendo o alinhamento de 50% nos dois blocos de cidades. Gostaria que ele crescesse horizontalmente até atingir o limite de 100% da linha e quebrasse para continuar. Assim, ao concluir as lojas da primeira cidade, iniciar o segundo bloco de cidade com demais lojas. É possível que um bloco se estenda até o total da linha, quebre e continue abaixo até o término do conteuúdo, somente assim iniciando o segundo bloco?

0 0 Responder • Compartilhar ›

Ernane Augusto
6 anos atrás

Excelente artigo David e ótimos exemplos também. Parabéns!!!

0 0 Responder • Compartilhar ›

David Arty Mod ↗ Ernane Augusto
6 anos atrás

Fala Ernane.

Valeu. Muito obrigado :D

Forte abraço.

0 0 Responder • Compartilhar ›

Inscreva-se Privacidade Política de Proteção de Dados