



Programação Web Front-End

Aula 3 - CSS

Profa. Rosangela de Fátima Pereira Marquesone
romarquesone@utfpr.edu.br

Proposta: apresentar os conceitos referentes ao posicionamento dos elementos a partir do conceito *flexbox layout*.

Objetivos: espera-se que após essa aula, você tenha habilidade para compreender os seguintes tópicos:

1. [Aprender o conceito de *flex-box*](#)
2. [Aprender as propriedades do elemento pai \(*flex container*\)](#)
3. [Aprender as propriedades dos elementos filhos \(*flex items*\)](#)

Dicas de aprendizado:

- Execute todos os passos com atenção, compreendendo o que está sendo realizado;
- Procure não copiar código, para ter a prática de digitar o código desenvolvido;
- Pergunte quando tiver alguma dúvida;
- Mantenha um histórico dos códigos desenvolvidos, seja no github ou em algum outro meio de armazenamento (e-mails, google drive, etc.);
- Tenha curiosidade e explore os recursos apresentados.

Tópicos anteriores:

- Compreender o que é HTML
- Compreender o que são tags HTML básicas
- Criar um arquivo .html no Visual Studio (VS) Code
- Abrir o arquivo .html em um navegador
- Visualizar o código-fonte de uma página em um navegador
- Inspeccionar a página em um navegador
- Utilizar o Live Server no VS Code
- Aprender a utilizar tags semânticas
- Aprender a inserir links
- Aprender a inserir listas
- Aprender a criar uma página com seu Curriculum Vitae (CV) (atividade prática)
- Aprender a inserir figuras
- Aprender a utilizar a tag semântica <figure>
- Inserir figuras em seu Curriculum Vitae (CV) (atividade prática)
- Aprender a criar formulários
- Criar um formulário (atividade prática)
- Descobrir o que é CSS
- Aprender a sintaxe do CSS

- Aprender os tipos de seletores CSS
- Aprender as formas de inclusão de CSS
- Aprender a definir cores
- Aprender a alterar as propriedades de texto
- Aprender o conceito de modelo de caixa do CSS
- Aprender a trabalhar com a margem
- Aprender a trabalhar com a borda
- Aprender a trabalhar com o preenchimento (padding)
- Aprender a usar a propriedade display
- Aprender a utilizar a propriedade float
- Aprender a utilizar a propriedade overflow
- Estruturar páginas por meio do modelo de caixa (atividade prática)

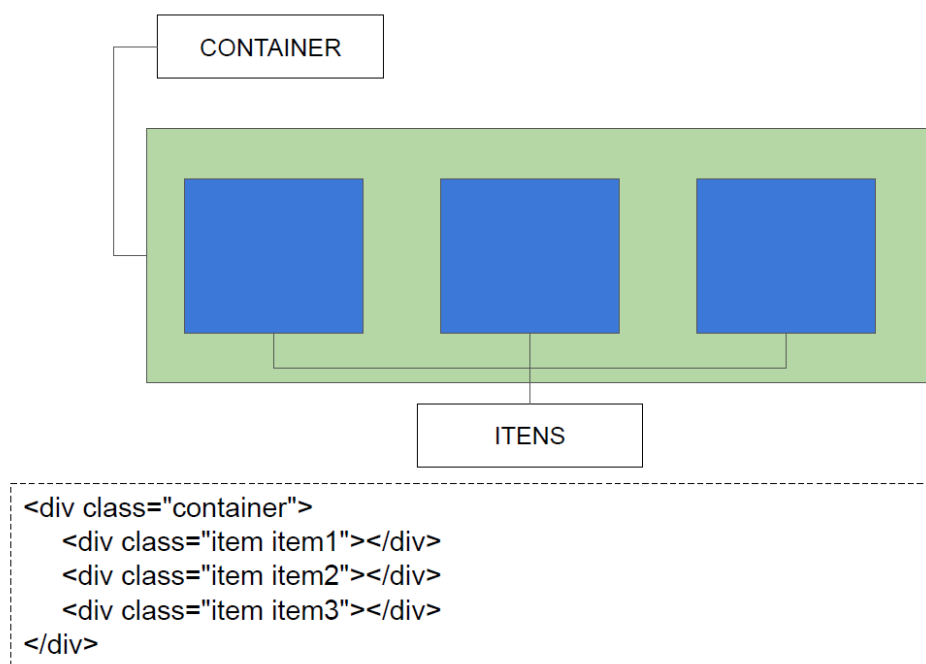
Passo 1 - Aprender o conceito flexbox

Vimos no tutorial anterior sobre a propriedade *display*, no qual aprendemos sobre os valores: *inline*, *block*, *inline-block* e *none*. Nesse passo veremos sobre um outro modo de posicionamento e alinhamento de elementos: o *flexbox*.

Flexbox (*flexible box*) é considerado um modelo de *layout* flexível, proposto em 2009. Ou seja, ela consegue lidar com o *layout* em uma dimensão de cada vez - seja uma linha ou uma coluna. Além disso, ele permite organizar os elementos de uma página HTML em *flex containers* (elemento pai), de forma dinâmica.

De forma um pouco mais simplificada, podemos entender o conceito de flexbox como um recurso que possibilita você tornar os elementos de uma página responsivos, que se organizam de forma automática conforme a página é redimensionada, gerenciando questões de alinhamento, posicionamento e distribuição dos elementos por meio de regras CSS.

Veja um exemplo a seguir para compreender como ficam dispostos os elementos *flex items* (elementos filhos) dentro de um container.



Conforme descrito pelo [MDN](#), a introdução do flexbox possibilitou auxiliar nas seguintes questões sobre posicionamento e *layout*:

- Centralizar um bloco de conteúdo verticalmente dentro de seu pai;
- Fazer com que os elementos filhos de um container ocupem uma quantidade igual de largura/altura disponível;
- Fazer todas as colunas de um *layout* com múltiplas colunas terem a mesma altura, mesmo que contenham uma quantidade diferente de conteúdo.

Para saber posicionar e ajustar os elementos de um flexbox, é importante saber quais propriedades devem ser aplicadas ao **elemento-pai** e quais devem ser aplicadas aos **elementos-filhos**. No passo a seguir, veremos detalhes de cada propriedade.

Passo 2 - Aprender as propriedades do elemento pai (*flex container*)

Veja a seguir algumas das propriedades que podemos utilizar no elemento pai (*flex container*) de um flexbox:

- `display`
- `flex-direction`
- `flex-wrap`
- `justify-content`
- `align-items`
- `align-content`

display

A propriedade **display** é a que define que o container terá o valor flex em seus elementos. Para isso, considerando o código apresentado na figura anterior, podemos aplicar esse valor da seguinte maneira:

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


flex-direction

Essa propriedade é utilizada para estabelecer o eixo principal, definindo assim a direção em que os itens são alinhados no container. Essa propriedade pode receber os seguintes valores:

- **row**: esquerda para a direita
- **row-reverse**: direita para a esquerda
- **column**: mesmo que row, mas de cima para baixo
- **column-reverse**: mesmo que row-reverse, mas de baixo para cima




Veja alguns exemplos a seguir com os valores possíveis:

css	html
<pre>.container{ display: flex; flex-direction: row; background-color: rgb(8, 155, 96); } .item{ margin: 10px; padding: 20px; background-color: beige; }</pre>	<pre><!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <div class="container"> <div class="item">ODS 1</div></pre>

	<pre> <div class="item">ODS 2</div> <div class="item">ODS 3</div> <div class="item">ODS 4</div> </div> </body> </html> </pre>
resultado	
	

PRATICANDO:

- Abra o arquivo “**aula3-css.zip**” e e descompacte o arquivo;
- Abra a pasta descompactada no VS Code;
- Acesse o arquivo **ex-flex-direction.html** e faça sua visualização via Live Server;
- Altere as propriedades de flex-direction e visualize o formato do layout, para obter os resultados a seguir:

Conseguiu descobrir quais valores das propriedades foram incluídos em cada opção?

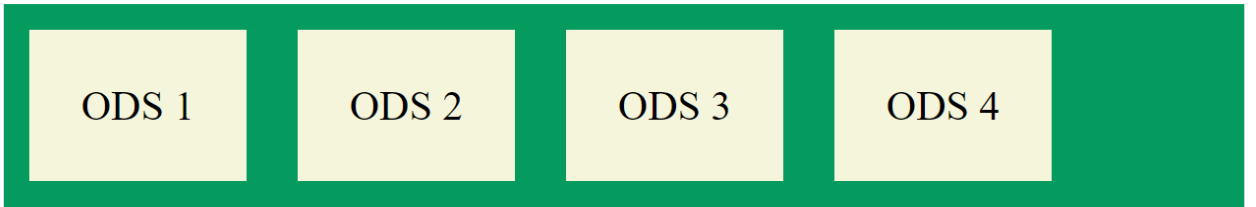
flex-wrap

Essa propriedade especifica se os itens posicionados irão quebrar ou não de linha ao serem redimensionados. Para isso, os seguintes valores podem ser aceitos:

- **nowrap**: todos os itens ficarão em uma só linha (valor padrão). Ou seja, não haverá quebra de linha.
- **wrap**: os flex items vão quebrar em múltiplas linhas, de cima para baixo.
- **wrap-reverse**: os itens vão quebrar em múltiplas linhas de baixo para cima.

PRATICANDO:

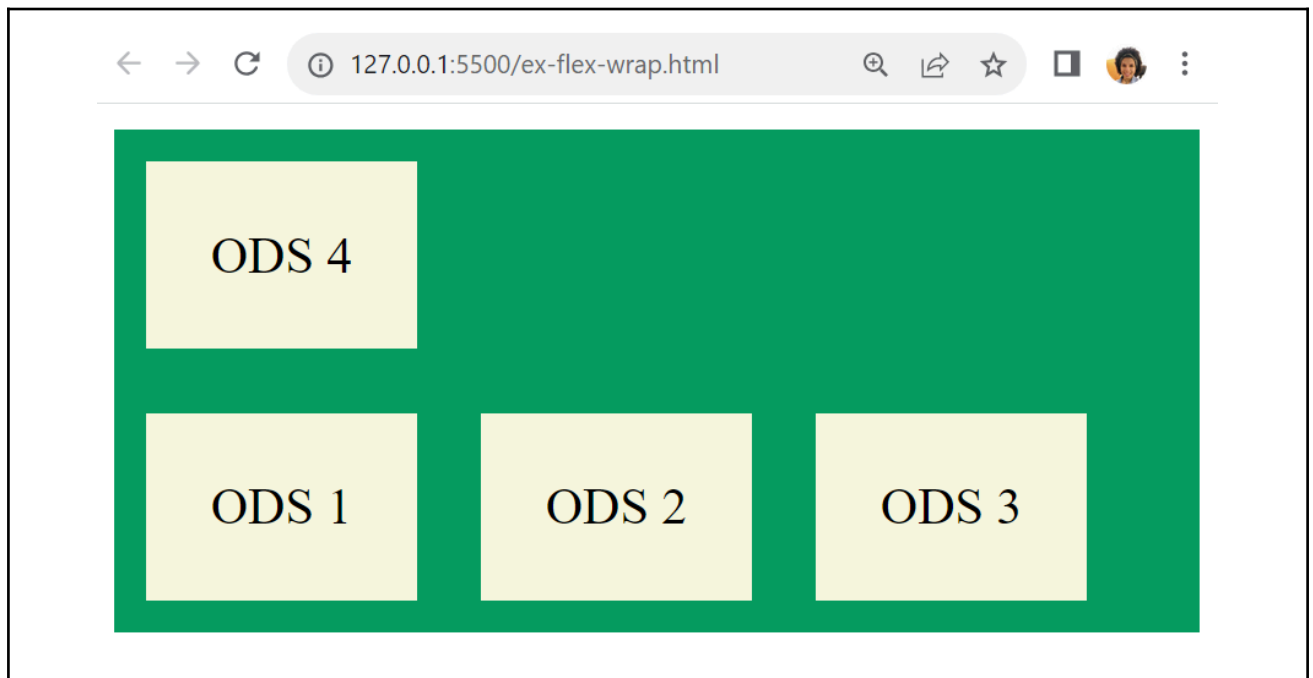
- Abra o arquivo “**ex-flex-wrap.html**” e visualize o resultado via Live Server.
- Faça alterações nas propriedades em vermelho para visualizar novas possibilidades de layout.

css	html
<pre>.container{ background-color: rgb(8, 155, 96); display: flex; flex-wrap: wrap; } .item{ margin: 10px; padding: 20px; background-color: beige; }</pre>	<pre><!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <div class="container"> <div class="item">ODS 1</div> <div class="item">ODS 2</div> <div class="item">ODS 3</div> <div class="item">ODS 4</div> </div> </body> </html></pre>
resultado	
	
resultado (com redimensionamento da tela)	



Ao aplicar o valor *wrap-reverse*, os itens vão quebrar em múltiplas linhas de baixo para cima, conforme exemplo a seguir:

css	html
<pre>.container{ background-color: rgb(8, 155, 96); display: flex; flex-wrap: wrap-reverse; } .item{ margin: 10px; padding: 20px; background-color: beige; }</pre>	<pre><!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <div class="container"> <div class="item">ODS 1</div> <div class="item">ODS 2</div> <div class="item">ODS 3</div> <div class="item">ODS 4</div> </div> </body> </html></pre>
resultado (com redimensionamento da tela)	



justify-content

Essa propriedade é utilizada para definir o alinhamento dos itens ao longo do eixo principal. Veja alguns dos valores possíveis:

- **flex-start**: valor padrão que alinha os itens no início do container;
- **flex-end**: valor que alinha os itens no final do container;
- **center**: alinha os itens no centro do container;
- **space-around**: cria um espaçamento entre os elementos, de forma que os espaçamentos do meio são duas vezes maiores que o inicial e final;
- **space-between**: cria um espaçamento igual entre os elementos, porém ele mantém o primeiro grudado no início do container e o último ao final.

PRATICANDO:

- Abra o arquivo “**ex-flex-justify-content.html**” e visualize o resultado via Live Server.
- Faça alterações nas propriedades em vermelho para visualizar novas possibilidades de layout.

css	html
<pre> .flex-start { justify-content: flex-start; } .flex-end { justify-content: flex-end; } .center { justify-content: center; } .space-between { justify-content: space-between; </pre>	<pre> <!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h1>justify-content: flex-start;</h1> <section class="container flex-start"> <div class="item">ODS 1</div> <div class="item">ODS 2</div> </pre>

```

}

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

.container {
    display: flex; /*definindo o display como
flex ao elemento pai*/
    max-width: 500px;
    margin: 0 auto; /*centralizar os itens*/
    border: 1px solid #ccc;
}

.column {
    min-height: 200px;
    flex-direction: column;
}

.item {
    margin: 5px;
    padding: 0 10px;
    background: rgb(8, 155, 96);
    color: beige;
    text-align: center;
    font-size: 1.5em;
}

h1 {
    text-align: center; /*centralizar os textos*/
    margin: 20px 0 0 0;
    font-size: 1.25em;
}

body {
    font-family: monospace;
    color: #333;
}

```

```

<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

```

<h1>justify-content: flex-end;</h1>
<section class="container flex-end">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

```

<h1>justify-content: center;</h1>
<section class="container center">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

```

<h1>justify-content: space-between;</h1>
<section class="container space-between">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

```

<h1>justify-content: space-around;</h1>
<section class="container space-around">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

```

<h1>justify-content: flex-start;
flex-direction: column</h1>
<section class="container flex-start
column">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

```

<h1>justify-content: flex-end; flex-direction:
column</h1>
<section class="container flex-end
column">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

```

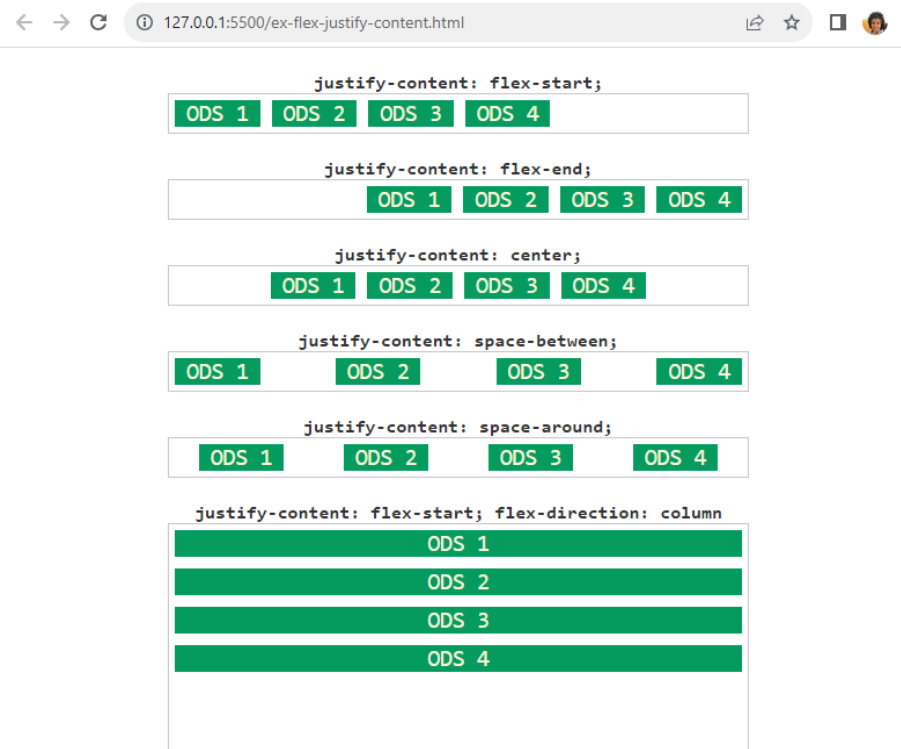
```
<h1>justify-content: center; flex-direction:
column</h1>
<section class="container center column">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

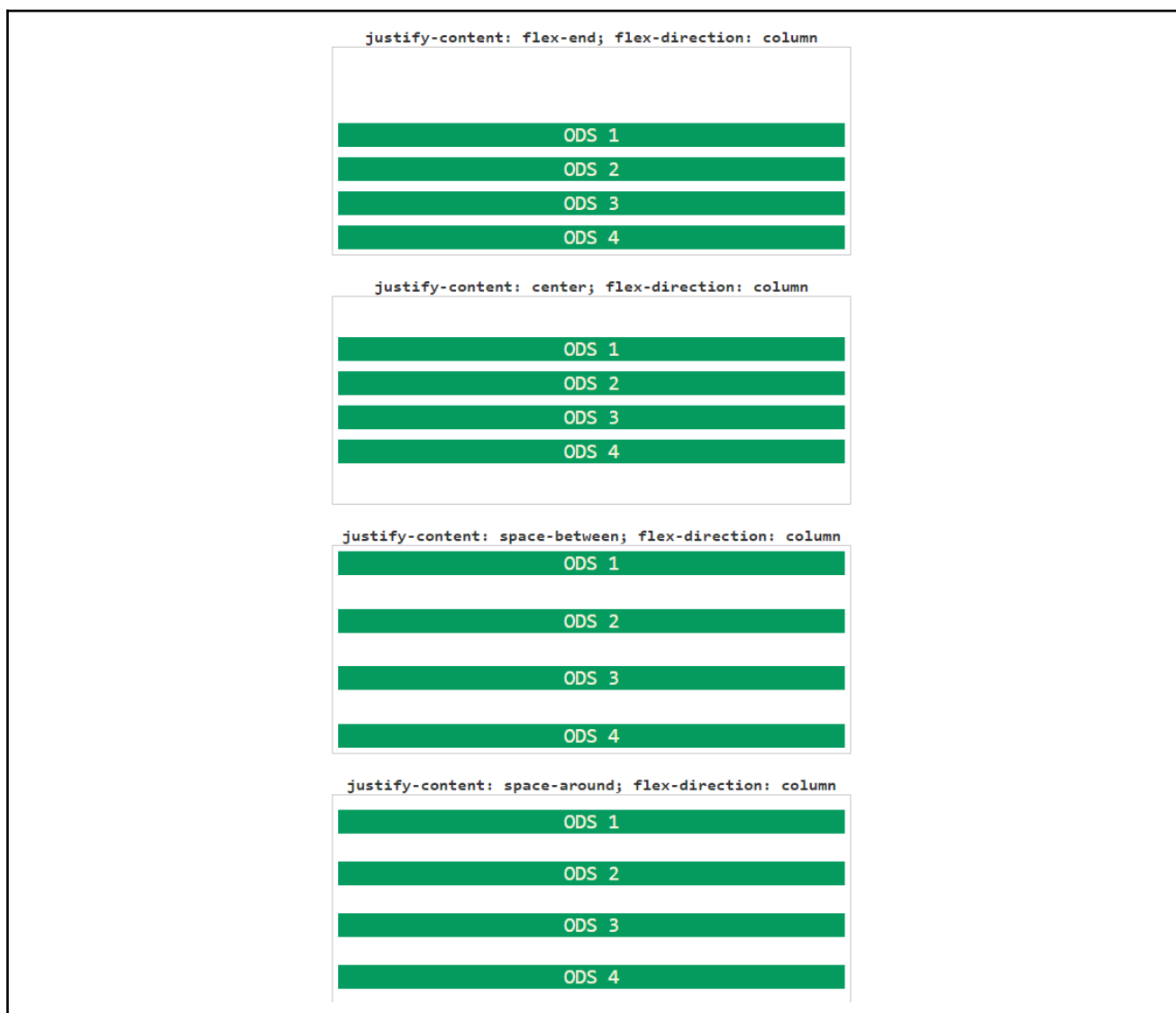
<h1>justify-content: space-between;
flex-direction: column</h1>
<section class="container space-between
column">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

<h1>justify-content: space-around;
flex-direction: column</h1>
<section class="container space-around
column">
<div class="item">ODS 1</div>
<div class="item">ODS 2</div>
<div class="item">ODS 3</div>
<div class="item">ODS 4</div>
</section>

</body>
</html>
```

resultado





align-items

Essa propriedade é utilizada para alinhar os itens de acordo com o eixo de um container. Confira a seguir alguns dos valores possíveis que podem ser utilizados:

- ***stretch***: nesta opção, caso os itens sejam menores que o container, os itens autodimensionados serão igualmente ampliados para preencher o container.
- ***flex-start***: realiza o alinhamento dos itens flexíveis de acordo com o início principal do flex container.
- ***flex-end***: realiza o alinhamento dos itens flexíveis de acordo com a extremidade final do flex container.
- ***center***: os itens são centralizados na linha do eixo cruzado.
- ***baseline***: os itens passam a ser alinhados a partir da base da primeira linha de texto de cada item.

PRATICANDO: abra o arquivo “**ex-flex-align-items.html**” e visualize o resultado via Live Server. Faça alterações nas propriedades em vermelho para visualizar novas possibilidades de layout.

css	html
<pre>.stretch { align-items: stretch;</pre>	<pre><!DOCTYPE html> <html lang="pt-br"></pre>

```

}

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

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

.center {
    align-items: center;
}

.baseline {
    align-items: baseline;
}

.container {
    max-width: 500px;
    margin: 0 auto;
    display: flex;
    border: 1px solid;
}

.column {
    flex-direction: column;
}

.item {
    flex: 1;
    margin: 5px;
    padding: 0 10px;
    background: rgb(8, 155, 96);
    color: beige;
    text-align: center;
    font-size: 1.5em;
}

.centralizado {
    height: 400px;
    justify-content: center;
    align-items: center;
}

.centralizado .item {
    flex: 0;
    padding: 10px;
}

h1 {
    text-align: center;
    margin: 20px 0 0 0;
    font-size: 1.25em;
}

body {

```

```

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Exemplos - CSS - Aula 3</title>
    <link rel="stylesheet" href="css/style.css" />
</head>
<body>

    <h1>align-items: stretch;</h1>
    <section class="container stretch">
        <div class="item">ODS 1</div>
        <div class="item">ODS 2 - Fome zero e agricultura
sustentável</div>
        <div class="item">ODS 3 - Saúde e bem estar</div>
        <div class="item">ODS 4</div>
    </section>

    <h1>align-items: flex-start;</h1>
    <section class="container flex-start">
        <div class="item">ODS 1</div>
        <div class="item">ODS 2 - Fome zero e agricultura
sustentável</div>
        <div class="item">ODS 3 - Saúde e bem estar</div>
        <div class="item">ODS 4</div>
    </section>

    <h1>align-items: flex-end;</h1>
    <section class="container flex-end">
        <div class="item">ODS 1</div>
        <div class="item">ODS 2 - Fome zero e agricultura
sustentável</div>
        <div class="item">ODS 3 - Saúde e bem estar</div>
        <div class="item">ODS 4</div>
    </section>

    <h1>align-items: center;</h1>
    <section class="container center">
        <div class="item">ODS 1</div>
        <div class="item">ODS 2 - Fome zero e agricultura
sustentável</div>
        <div class="item">ODS 3 - Saúde e bem estar</div>
        <div class="item">ODS 4</div>
    </section>

    <h1>align-items: baseline;</h1>
    <section class="container baseline">
        <div class="item">ODS 1</div>
        <div class="item">ODS 2 - Fome zero e agricultura
sustentável</div>
        <div class="item">ODS 3 - Saúde e bem estar</div>
        <div class="item">ODS 4</div>
    </section>

    <h1>align-items: flex-stretch; flex-direction:
column;</h1>
    <section class="container stretch column">
        <div class="item">ODS 1</div>

```


`align-items: stretch;`

ODS 1	ODS 2 - Fome zero e agricultura sustentável	ODS 3 - Saúde e bem estar	ODS 4
-------	--	------------------------------------	-------

`align-items: flex-start;`

ODS 1	ODS 2 - Fome zero e agricultura sustentável	ODS 3 - Saúde e bem estar	ODS 4
-------	--	------------------------------------	-------

`align-items: flex-end;`

	ODS 2 - Fome zero e agricultura sustentável	ODS 3 - Saúde e bem estar	
ODS 1			ODS 4

`align-items: center;`

	ODS 2 - Fome zero e agricultura sustentável	ODS 3 - Saúde e bem estar	
ODS 1			ODS 4

`align-items: baseline;`

ODS 1	ODS 2 - Fome zero e agricultura sustentável	ODS 3 - Saúde e bem estar	ODS 4
-------	--	------------------------------------	-------

`align-items: flex-stretch; flex-direction: column;`

ODS 1
ODS 2 - Fome zero e agricultura sustentável
ODS 3 - Saúde e bem estar
ODS 4

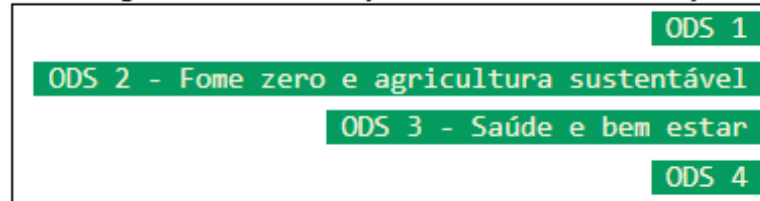
`align-items: flex-start; flex-direction: column;`

ODS 1
ODS 2 - Fome zero e agricultura sustentável
ODS 3 - Saúde e bem estar
ODS 4

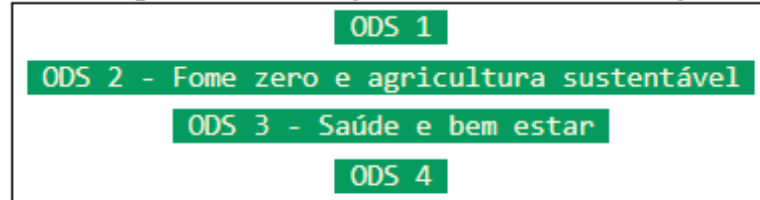
`align-items: flex-end; flex-direction: column;`

	ODS 1
ODS 2 - Fome zero e agricultura sustentável	
	ODS 3 - Saúde e bem estar
	ODS 4

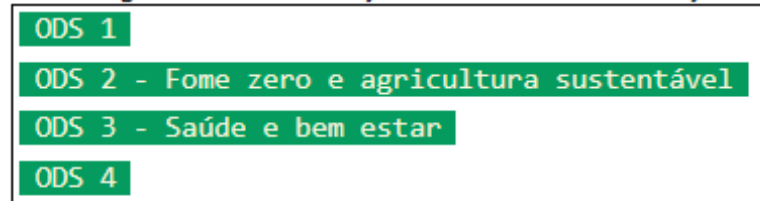
`align-items: flex-end; flex-direction: column;`



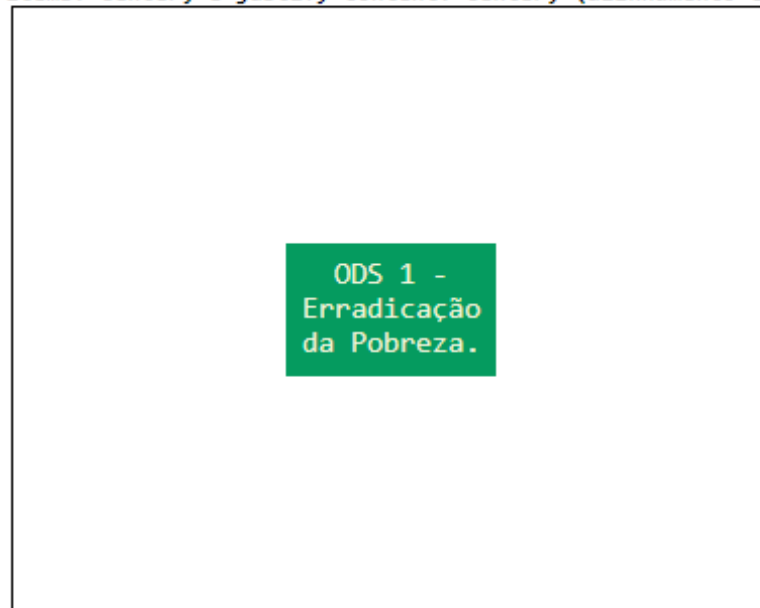
`align-items: center; flex-direction: column;`



`align-items: baseline; flex-direction: column;`

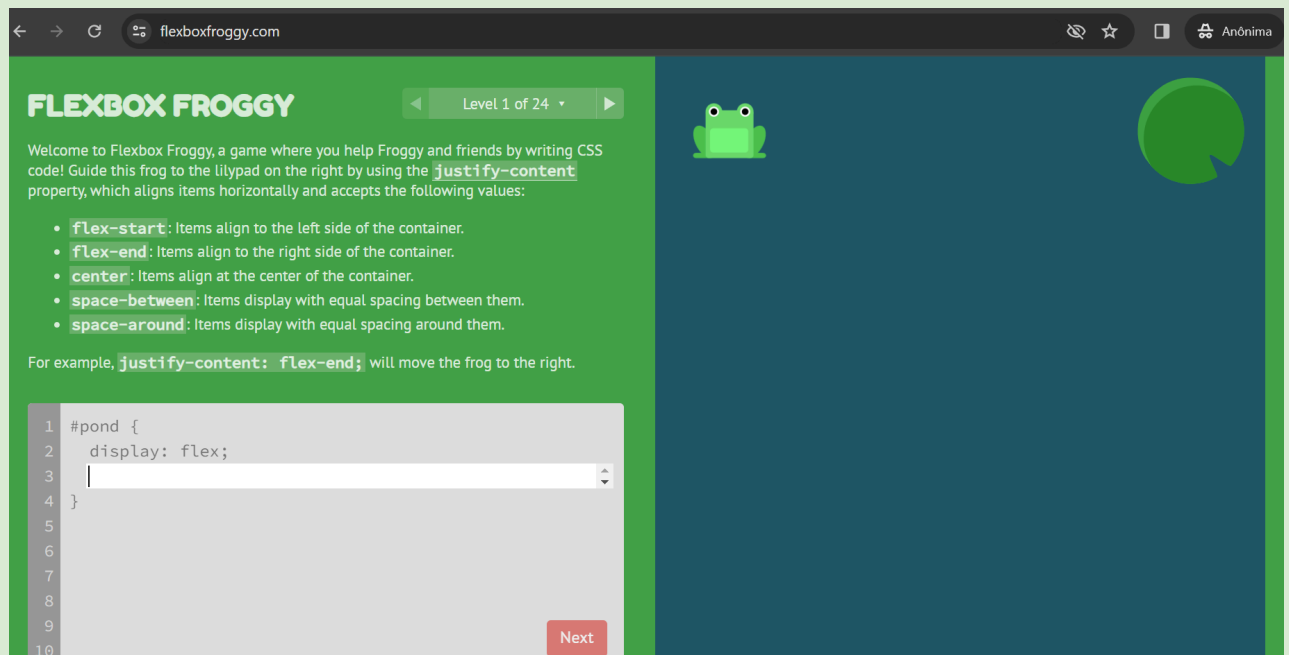


`align-items: center; e justify-content: center; (alinhamento central)`



Vamos jogar?

[Flexbox Froggy](#) é um game criado para te ajudar a aprender os conceitos de flexbox e CSS. São 24 níveis, no qual em cada nível você precisa indicar o comando correto em CSS para que o sapo fique posicionado no local adequado.



Aplicando os conceitos vistos até o momento, tente passar até o nível 10.

Bora lá?

Passo 3 - Aprender as propriedades dos elementos filhos (*flex items*)

Conforme já mencionado, os elementos *flex items* são os filhos do elemento pai *flex container*.

Em relação a esses elementos, podemos inserir as seguintes propriedades:

- align-self
- flex-grow
- flex-basis
- flex-shrink
- order

align-self

Assim como temos o align-items, que especifica o alinhamento de todos os elementos de um flex container, podemos utilizar também a propriedade align-self, que possibilita especificar o alinhamento de um elemento específico. Os valores possíveis são similares aos da propriedade align-items:

- **stretch**: o item será ampliado para preencher o container.
- **flex-start**: realiza o posicionamento do item no início do container.
- **flex-end**: realiza o posicionamento do item no fim do container
- **center**: o item é centralizado no container.
- **baseline**: o item passa a ser alinhado a partir da base da primeira linha do container.

flex-grow

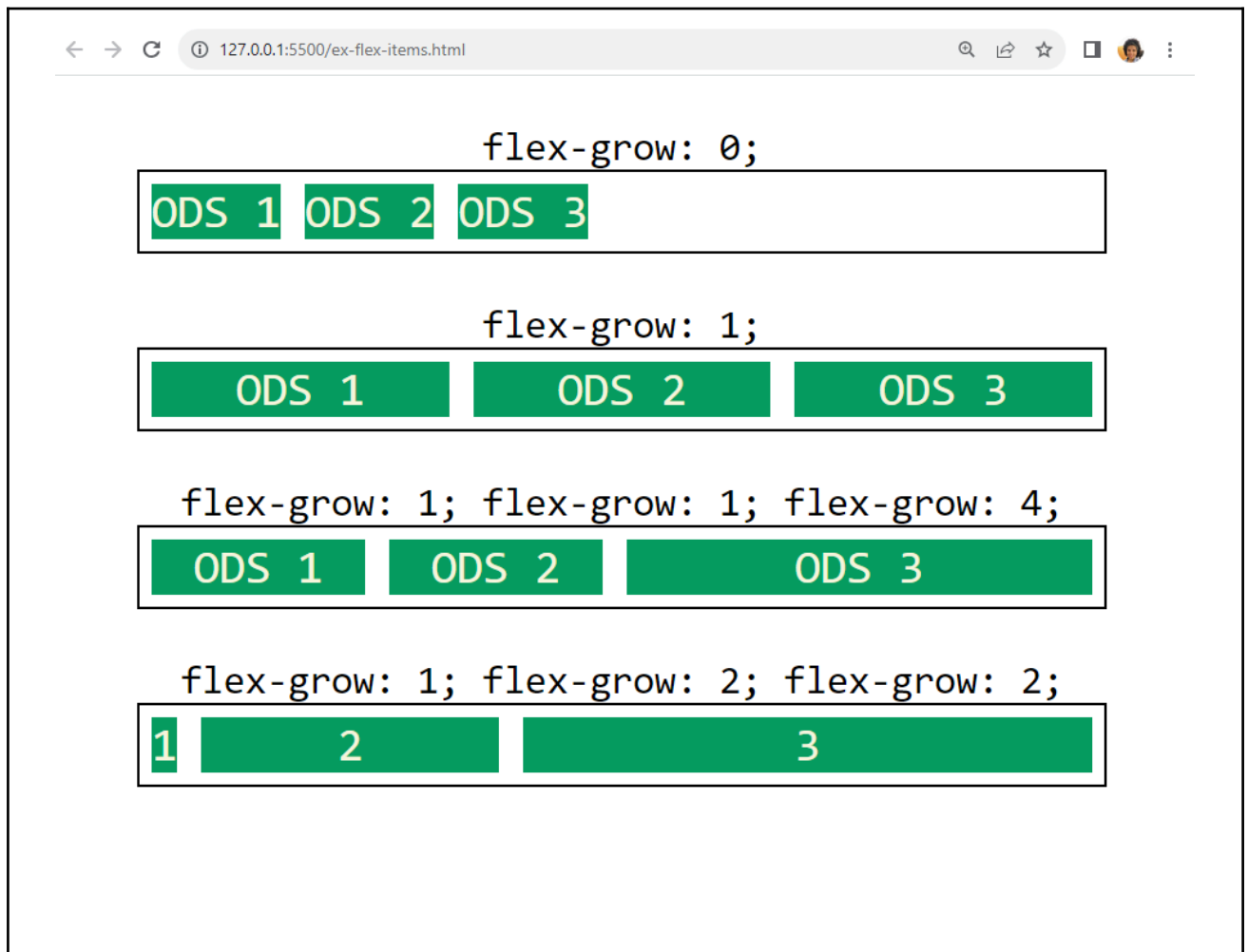
Essa propriedade é utilizada para definir a habilidade de um *flex item* crescer. Por padrão o valor dessa propriedade é zero. Caso seja definido 1 para todos os elementos filhos, eles tentarão ter a mesma largura e vão ocupar 100% do container. Porém, caso tenha uma linha contendo quatro elementos, dos quais três possuem a propriedade “flex-grow:1” e um outro com a propriedade “flex-grow: 2”, o “flex-grow: 2” tentará ocupar 2 vezes mais espaço extra do que os outros elementos.

PRATICANDO:

- Abra o arquivo “**ex-flex-grow.html**” e visualize o resultado via Live Server.
- Faça alterações nas propriedades em vermelho para visualizar novas possibilidades de layout.

css	html
<pre>.grow0 { flex-grow: 0; } .grow1 {</pre>	<pre><!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width,</pre>

<pre> flex-grow: 1; } .grow2 { flex-grow: 2; } .grow4 { flex-grow: 4; } .item { margin: 5px; background: rgb(8, 155, 96); color: beige; text-align: center; font-size: 1.5em; } .container { max-width: 400px; margin: 0 auto; display: flex; border: 1px solid; } h1 { text-align: center; margin: 20px 0 0 0; font-size: 1.25em; font-weight: normal; } body { font-family: monospace; } </pre>	<pre> initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h1>flex-grow: 0;</h1> <section class="container"> <div class="item grow0">ODS 1</div> <div class="item grow0">ODS 2</div> <div class="item grow0">ODS 3</div> </section> <h1>flex-grow: 1;</h1> <section class="container"> <div class="item grow1">ODS 1</div> <div class="item grow1">ODS 2</div> <div class="item grow1">ODS 3</div> </section> <h1>flex-grow: 1; flex-grow: 1; flex-grow: 4;</h1> <section class="container column-reverse"> <div class="item grow1">ODS 1</div> <div class="item grow1">ODS 2</div> <div class="item grow4">ODS 3</div> </section> <h1>flex-grow: 1; flex-grow: 2; flex-grow: 2;</h1> <section class="container column-reverse"> <div class="item grow0">1</div> <div class="item grow1">2</div> <div class="item grow2">3</div> </section> </body> </html> </pre>
<div>resultado</div>	



flex-basis

A propriedade flex-basis é utilizada para definir o tamanho inicial de um elemento flex item, antes de ele ser reduzido ou ampliado. Ou seja, caso seja definido o valor “flex-grow: 1”; e seja definido o valor “auto” na propriedade flex-basis, o valor restante para ocupar o container é distribuído ao redor do conteúdo do elemento flex-item.

Os seguintes valores são aceitos:

- auto: faz com que a largura da base seja igual a do item. Se o item não tiver tamanho especificado, o tamanho será de acordo com o conteúdo. Valor padrão.
- valor em unidade (% , em, px e etc).

flex-shrink

Essa propriedade é muito útil também para a responsividade do site, pois ela é capaz de definir a capacidade de redução de tamanho do item.

PRATICANDO: abra o arquivo “**ex-flex-shrink.html**” e visualize o resultado via Live Server. Faça alterações nas propriedades em vermelho para visualizar novas possibilidades de layout.

css	html
<pre>.shrink-1 { flex-grow: 1;</pre>	<pre><!DOCTYPE html> <html lang="pt-br"></pre>

<pre> flex-shrink: 1; } .shrink-0-basis { flex-grow: 1; flex-shrink: 0; flex-basis: 150px; } .flex-wrap { flex-wrap: wrap; } .item { margin: 5px; background: rgb(8, 155, 96); color: beige; text-align: center; font-size: 1.5em; } .container { width: 400px; margin: 0 auto; display: flex; border: 1px solid; } h1 { text-align: center; margin: 20px 0 0 0; font-size: 1.25em; } body { font-family: monospace; } </pre>	<pre> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h1>flex-shrink: 1;</h1> <section class="container"> <div class="item shrink-1">ODS 1</div> <div class="item shrink-1">ODS 2</div> <div class="item shrink-1">ODS 3</div> <div class="item shrink-1">ODS 4</div> </section> <h1>flex-shrink: 1; (com conteúdo diferente)</h1> <section class="container"> <div class="item shrink-1">ODS 1</div> <div class="item shrink-1">ODS 2</div> <div class="item shrink-1">ODS 3 - Saúde e bem estar</div> <div class="item shrink-1">ODS 4</div> </section> <h1>flex-shrink: 0; flex-basis: 150px; flex-wrap: wrap;</h1> <section class="container flex-wrap"> <div class="item shrink-0-basis">ODS 1</div> <div class="item shrink-0-basis">ODS 2</div> <div class="item shrink-0-basis">ODS 3 - Saúde e bem estar</div> <div class="item shrink-0-basis">ODS 4 - Educação de qualidade</div> </section> </body> </html> </pre>
<div>resultado</div>	

← → ↻ 127.0.0.1:5500/ex-flex-shrink.html 🔍 📄 ☆ 🏠 👤 ⋮

flex-shrink: 1;

ODS 1	ODS 2	ODS 3	ODS 4
-------	-------	-------	-------

flex-shrink: 1; (com conteúdo diferente)

ODS 1	ODS 2	ODS 3 - Saúde e bem estar	ODS 4
-------	-------	---------------------------	-------

flex-shrink: 0; flex-basis: 150px; flex-wrap: wrap;

ODS 1	ODS 2
ODS 3 - Saúde e bem estar	ODS 4 - Educação de qualidade

Cusioridade

As propriedades *flex-grow*, *flex-shrink*, e *flex-basis* podem ser combinadas por meio da propriedade abreviada *flex*.

A abreviatura *flex* permite definir os três valores na seguinte ordem: *flex-grow*, *flex-shrink*, *flex-basis*. Veja um exemplo:

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

Para saber mais!

Além dessas propriedades, pode ocorrer de você querer definir o tamanho mínimo e máximo de um elemento, e ainda assim querer que ele aumente e diminua de forma dinâmica.

Para esse cenário, podemos utilizar as propriedades **min-width** e **max-width** para a largura mínima e máxima, e a propriedade **min-height** e **max-height** para a altura mínima e máxima, respectivamente.

order

Essa propriedade é utilizada para modificar a ordem dos flex itens.

PRATICANDO: abra o arquivo “**ex-flex-order.html**” e visualize o resultado via Live Server. Faça alterações nas propriedades em vermelho para visualizar novas possibilidades de layout.

css	html
<pre>.order1 { order: 1; } .order2 { order: 2; } .order3 { order: 3; } .order4 { order: 4; } .column { flex-direction: column; } .item { margin: 5px; padding: 0 5px; background: rgb(8, 155, 96); color: beige; text-align: center; font-size: 1.5em; } .container { max-width: 400px; margin: 0 auto; display: flex; border: 1px solid; } h1 { text-align: center; margin: 20px 0 0 0; font-size: 1.25em; } body { font-family: monospace; }</pre>	<pre><!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Exemplos - CSS - Aula 3</title> <link rel="stylesheet" href="css/style.css" /> </head> <body> <h1>(ordens misturadas)</h1> <section class="container"> <div class="item order2">ODS 1</div> <div class="item order4">ODS 2</div> <div class="item order1">ODS 3</div> <div class="item order3">ODS 4</div> </section> <h1>(ordens misturadas) flex-direction: column;</h1> <section class="container column"> <div class="item order2">ODS 1</div> <div class="item order1">ODS 2</div> <div class="item order3">ODS 3</div> <div class="item order4">ODS 4</div> </section> </body> </html></pre>
resultado	

← → ↻ 127.0.0.1:5500/ex-flex-shrink.html 🔍 📄 👤 ⋮

flex-shrink: 1;

ODS 1
ODS 2
ODS 3
ODS 4

flex-shrink: 1; (com conteúdo diferente)

ODS 1
ODS 2
ODS 3 - Saúde e bem estar
ODS 4

flex-shrink: 0; flex-basis: 150px; flex-wrap: wrap;

ODS 1

ODS 2

ODS 3 - Saúde e bem estar

ODS 4 - Educação de qualidade

A partir dessas propriedades, você passa a ter recursos para construir um produto digital ou página web com flexibilidade e de forma responsiva. Além desse conteúdo, você pode continuar a praticar o uso dessas propriedades a partir dos recursos disponíveis no site flexbox.help, que apresenta um meio dinâmico para visualizar os diferentes atributos, conforme o exemplo a seguir.

← → ↻ flexbox.help ☆ 📄 👤 ⋮

TEST CSS FLEXBOX RULES

CHILD COUNT

8

12345

678

FLEX-DIRECTION Details <input checked="" type="radio"/> row (default) <input type="radio"/> row-reverse <input type="radio"/> column <input type="radio"/> column-reverse	FLEX-WRAP <i>whether items wrap to the next row (only applies if combined width of items is greater than container's)</i> Details <input type="radio"/> nowrap (default) <input checked="" type="radio"/> wrap <input type="radio"/> wrap-reverse	JUSTIFY-CONTENT <i>alignment along the x axis</i> Details <input checked="" type="radio"/> flex-start (default) <input type="radio"/> flex-end <input type="radio"/> center <input type="radio"/> space-around <input type="radio"/> space-evenly <input type="radio"/> space-between	ALIGN-ITEMS <i>alignment along the y axis</i> Details <input type="radio"/> stretch (default) <input type="radio"/> baseline <input type="radio"/> center <input checked="" type="radio"/> flex-start <input type="radio"/> flex-end	ALIGN-CONTENT <i>only applies if there is more than one row of items</i> Details <input type="radio"/> stretch (default) <input checked="" type="radio"/> center <input type="radio"/> flex-start <input type="radio"/> flex-end <input type="radio"/> space-around <input type="radio"/> space-evenly <input type="radio"/> space-between
---	---	--	--	--

Vamos jogar?

Agora que pode aprender os demais conceitos de flexbox, volte ao [Flexbox Froggy](#) e tente chegar até o último nível!

Considerações finais

Caso tenha chegado até aqui, você conseguiu completar o conteúdo do terceiro tutorial sobre CSS. A partir desses recursos, você passa a ser capaz de tornar suas páginas mais bonitas e agradáveis. Mas há muito mais para ser aprendido sobre CSS ainda. Continue explorando as opções disponíveis, utilizando sua criatividade.

Bom estudo!