Desenvolvimento de Sistemas para a Web I

Capítulo 8 - Layout com Estilos

Considerações ao Começar um Layout

- Sempre separe o seu conteúdo (HTML) da apresentação (CSS)
  - Torna a página mais fácil de manter e dá mais flexibilidade para funcionar em diferentes plataformas ou aparelhos
  - Faça sua página HTML e vincule com um arquivo CSS externo
- Teste o seu site em algum navegador (de preferência o Chrome)
- Se for necessário escrever regras CSS específicas para o IE6 e IE7, use comentários condicionais
  - https://www.quirksmode.org/css/condcom.html

Abordagens de Layout

- **Layout Fixo**: usa *pixels* para as larguras. A largura não muda quando vista de celulares, tablets ou quando uma janela do navegador for reduzida
- **Layout Fluido**: usa *porcentagens* para as larguras, permitindo que a página encolha ou se expanda dependendo das condições de visualização (web design responsivo)
- **Layout Elástico**: usa *ems* para largura ou qualquer outra propriedade relacionada a tamanho. Assim, a página se dimensiona de acordo com as configurações do tamanho da fonte

Estruturando a Página

- Divida as seções lógicas do seu documento em elementos `article`, `aside`, `nav`, `section`, `header`, `footer`, `main` e `div`
- Coloque o conteúdo em uma ordem que seria mais útil se o CSS não fosse usado. Exemplo: `<header>`, seguido por `<main>`, seguido por um ou mais `<aside>`, seguido por `<footer>`
- Use os cabeçalhos `<h1>` e `<h2>` de maneira consistente para identificar e priorizar a informação dentro das seções da sua página
- Use comentários `<!--` e `-->` se achar necessário

Tipos de Layout

- Layout de Fluxo (Flow Layout)
- Layout de Posicionamento (Positioning Layout)
- Layout de Múltiplas Colunas (Multi-column Layout)
- Layout Flexível (Flexbox Layout)
- Layout de Grade (Grid Layout)

Layout de Fluxo (Flow Layout)

- É o padrão do CSS
- Os elementos HTML são dispostos um abaixo do outro ou um ao lado do outro, seguindo a ordem em que aparecem no código HTML

Layout de Posicionamento (Positioning Layout)

- Envolve o uso das propriedades CSS `position`, `top`, `bottom`, `left` e `right` para posicionar elementos de maneira precisa na página
- Permite que você coloque elementos em lugares específicos, mesmo que eles não estejam na ordem natural do fluxo

Layout de Múltiplas Colunas (Multi-column Layout)

- Envolve o uso da propriedade CSS `column-count` para dividir o conteúdo em várias colunas
- É especialmente útil para conteúdo de texto, como artigos de notícias ou blogs

Layout Flexível (Flexbox Layout)

- Envolve o uso da propriedade CSS `display: flex` para criar um contêiner que pode ser dimensionado e posicionado de maneira flexível
- Os elementos dentro do contêiner podem ser organizados em uma única linha ou coluna, ou em várias linhas e colunas, dependendo da configuração

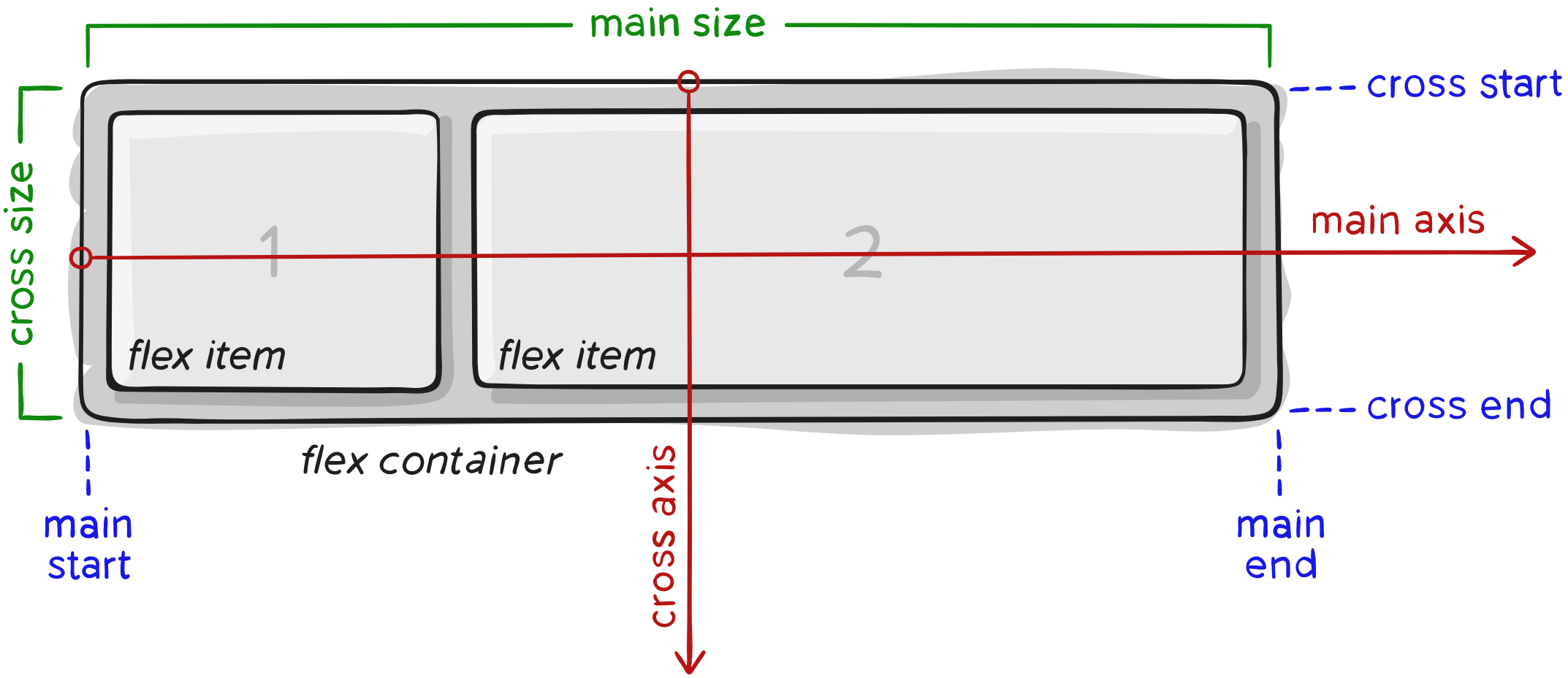
Layout de Grade (Grid Layout)

- Envolve o uso da propriedade CSS `display: grid` para criar um grid de células, onde os elementos HTML podem ser colocados em posições específicas
- Permite criar designs complexos com elementos dispostos em linhas e colunas

Flexbox (Layout Flexível)

- O Flexbox (de Flexible Box) é um método de layout unidimensional para organizar itens em linhas ou colunas. Os itens flexionam (expandem) para preencher espaço adicional ou encolhem para caber em espaços menores
- Possui propriedades que devem ser declaradas no contêiner (o elemento-pai, que chamamos de flex contêiner), enquanto outras devem ser declaradas nos elementos-filhos (flex itens)

Especificação do Layout Flexbox



Especificação do Layout Flexbox

- **main-axis** (Eixo principal): o eixo principal de um flex contêiner é o eixo primário e ao longo dele são inseridos os flex itens. **Cuidado**: O eixo principal não é necessariamente horizontal; vai depender da propriedade `flex-direction` (veja abaixo)
- **main-start** | **main-end**: os flex itens são inseridos dentro do contêiner começando pelo lado *start*, indo em direção ao lado *end*
- **main-size** (Tamanho principal): A largura ou altura de um flex item, dependendo da direção do contêiner, é o tamanho principal do item. A propriedade de tamanho principal de um flex item pode ser tanto *width* quanto *height*, dependendo de qual delas estiver na direção principal

Especificação do Layout Flexbox

- **cross-axis** (Eixo transversal): o eixo perpendicular ao eixo principal é chamado de eixo transversal. Sua direção depende da direção do eixo principal
- **cross-start** | **cross-end**: linhas flex são preenchidas com ítens e adicionadas ao contêiner, começando pelo lado cross start do flex container em direção ao lado cross end
- **cross size**: a largura ou altura de um flex item, dependendo do que estiver na dimensão transversal, é o cross size do item. A propriedade cross size pode ser tanto a largura quanto a altura do ítem, o que estiver na transversal

Propriedades para o Elemento-pai

- HTML

```
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
  </div>
```

- CSS

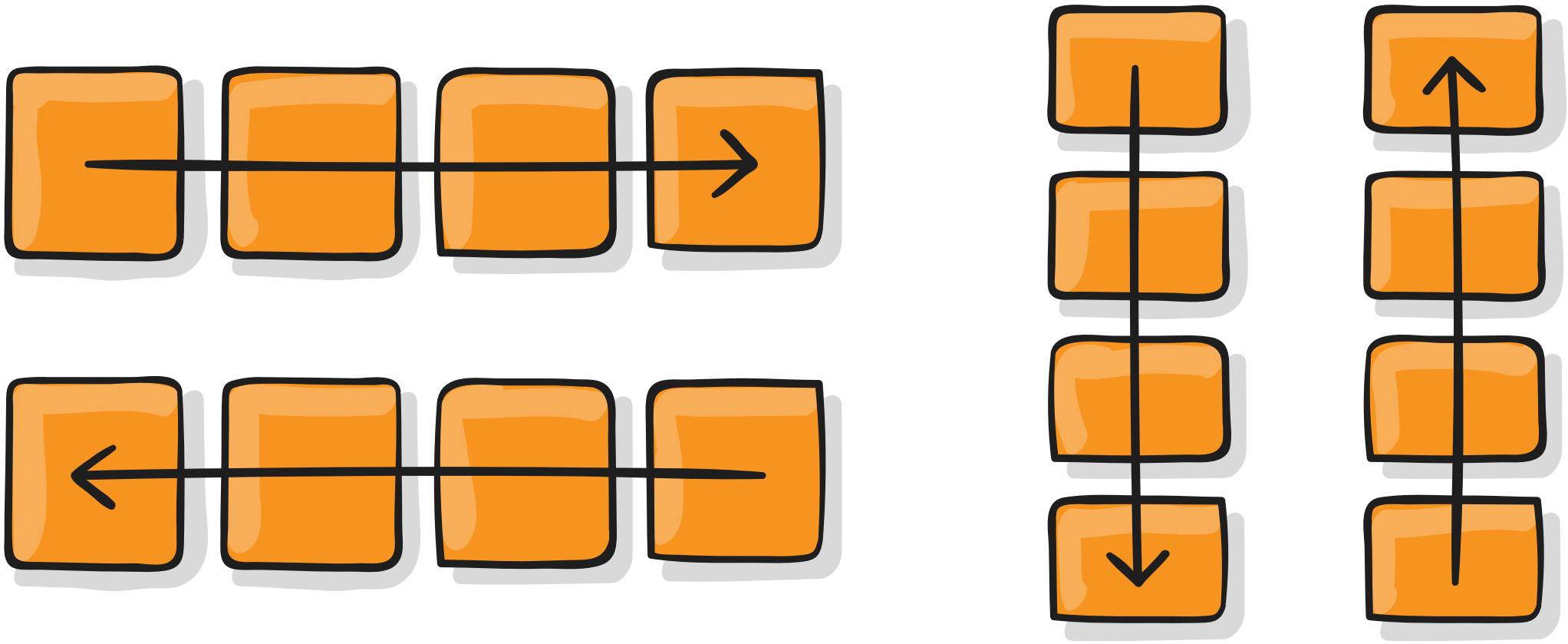
```
.flex-container {
  display: flex;
}
```

Display

- **display**: Esta propriedade define um flex contêiner; inline ou block dependendo dos valores passados. Coloca todos os elementos-filhos diretos num contexto Flex.

```
.container {
  display: flex; /* or inline-flex */
}
```

Flex-direction



Flex-direction

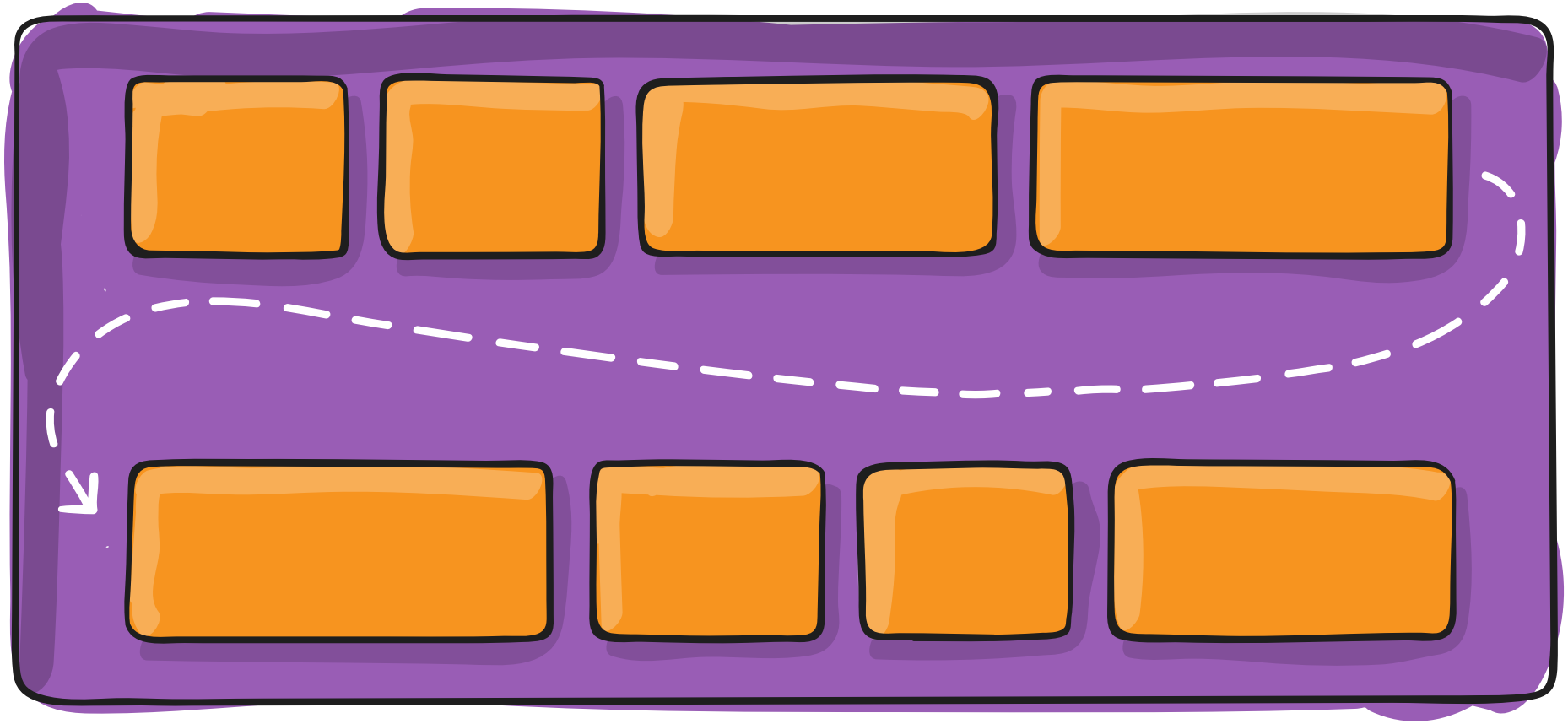
- Estabelece o eixo principal, definindo assim a direção em que os flex itens são alinhados no flex contêiner. O Flexbox é (com exceção de um wrapping opcional) um conceito de layout de uma só direção. Pense nos flex itens inicialmente posicionados ou em linhas horizontais ou em colunas verticais.

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

Flex-direction

- `row` (padrão): esquerda para a direita em `ltr` (left to right), direita para a esquerda em `rtl` (right to left)
- `row-reverse`: direita para a esquerda em ltr, esquerda para a direita em rtl
- `column`: mesmo que `row`, mas de cima para baixo
- `column-reverse`: mesmo que `row-reverse` mas de baixo para cima

Flex-wrap



Flex-wrap

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

```
.flex-container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

Flex-wrap

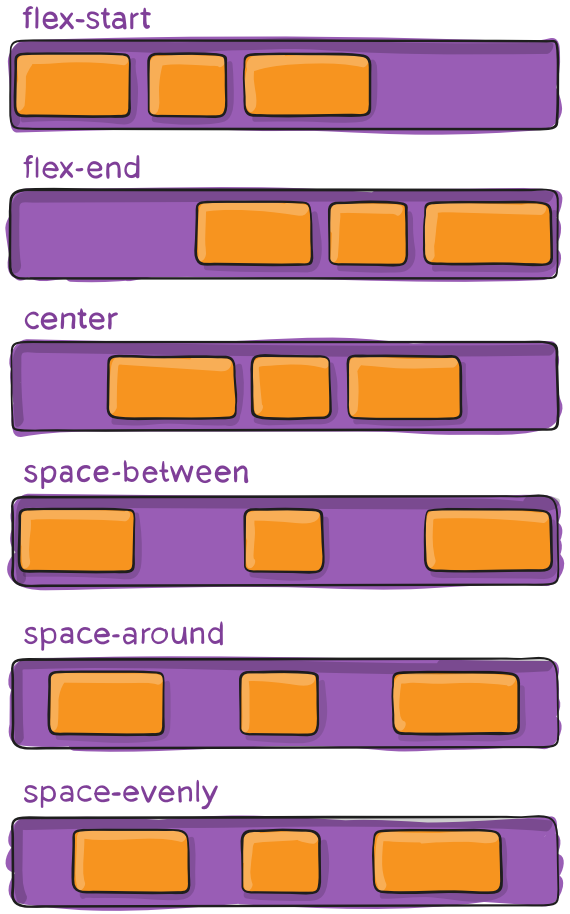
- `nowrap` (padrão): todos os flex itens ficarão em uma só linha
- `wrap`: os flex itens vão quebrar em múltiplas linhas, de cima para baixo
- `wrap-reverse`: os flex itens vão quebrar em múltiplas linhas de baixo para cima

Flex-flow

- A propriedade `flex-flow` é uma propriedade *shorthand* (uma mesma declaração inclui vários valores relacionados a mais de uma propriedade) que inclui `flex-direction` e `flex-wrap`. Determina quais serão os eixos principal e transversal do contêiner. O valor padrão é `row nowrap`.

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

Justify-content



Justify-content

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

```
.flex-container {
  justify-content: flex-start | flex-end | center |
                   space-between | space-around | space-evenly;
}
```

Justify-content

- `flex-start` (padrão): os itens são alinhados junto à borda de início (start) de acordo com qual for a `flex-direction` do contêiner
- `flex-end`: os itens são alinhados junto à borda final (end) de acordo com qual for a `flex-direction` do contêiner
- `start`: os itens são alinhados junto à borda de início da direção do writing-mode (modo de escrita).
- `end`: os itens são alinhados junto à borda final da direção do writing-mode (modo de escrita).

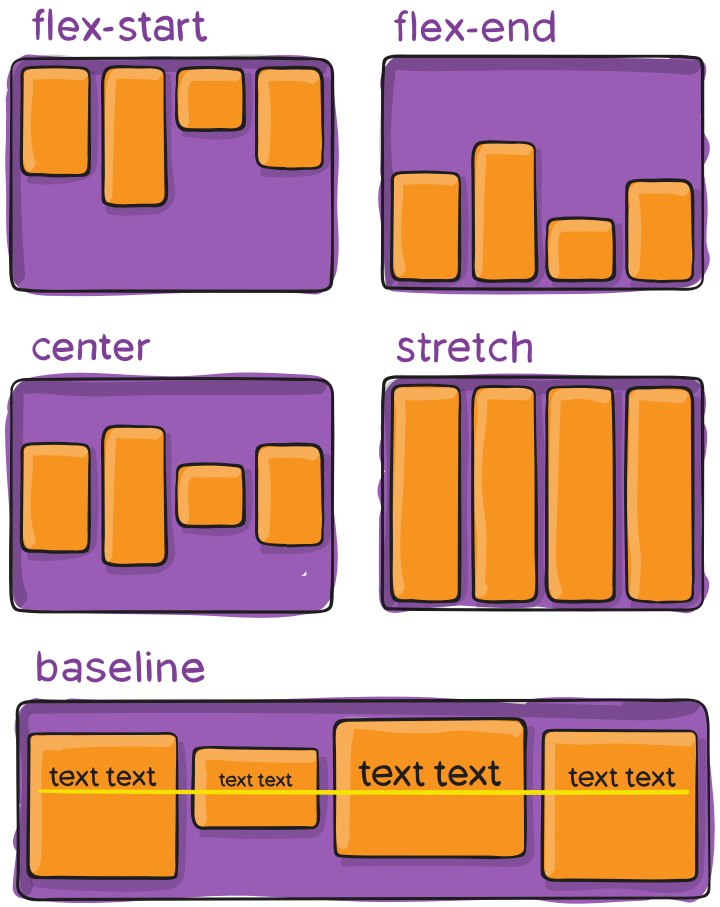
Justify-content

- `left`: os itens são alinhados junto à borda esquerda do contêiner, a não ser que isso não faça sentido com o `flex-direction` que estiver sendo utilizado. Nesse caso, se comporta como `start`
- `right`: os itens são alinhados junto à borda direita do contêiner, a não ser que isso não faça sentido com o `flex-direction` que estiver sendo utilizado. Nesse caso, se comporta como `start`
- `center`: os itens são centralizados na linha
- `space-between`: os itens são distribuídos de forma igual ao longo da linha; o primeiro item junto à borda inicial da linha, o último junto à borda final da linha

Justify-content

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

Align-items


flex-start
flex-end
center
stretch
baseline
text text
text text
text text
text text

Align-items

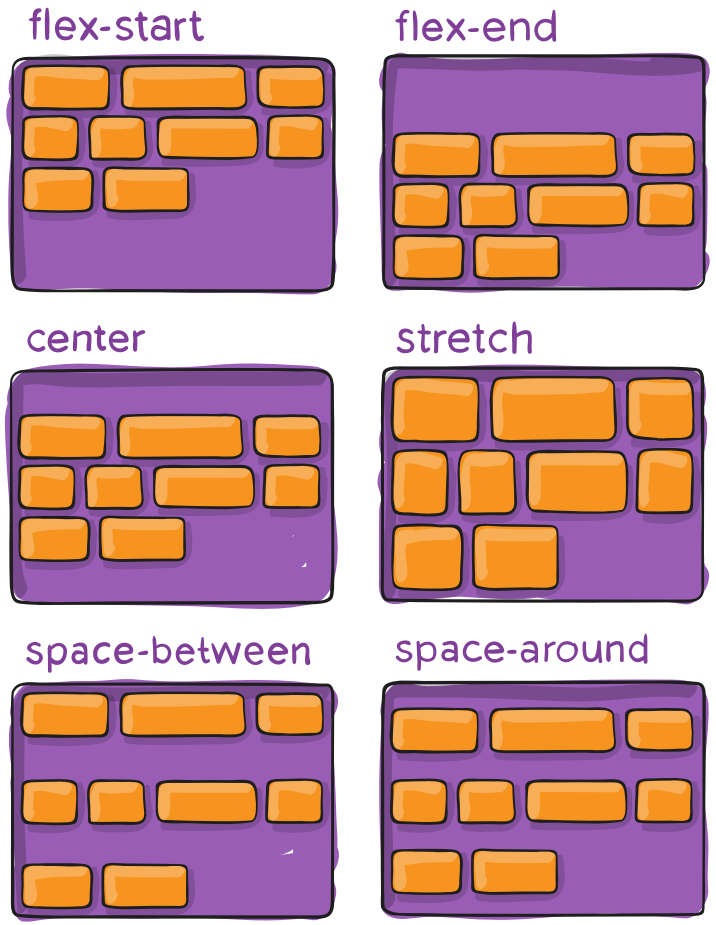
- Define o comportamento padrão de como flex itens são alinhados de acordo com o eixo transversal (cross axis). De certa forma, funciona de forma similar ao `justify-content`, porém no eixo transversal (perpendicular ao eixo principal).

```
.flex-container {
  align-items: stretch | flex-start | flex-end | center | baseline;
}
```

Align-items

- `stretch` (padrão): estica os itens para preencher o contêiner, respeitando o min-width/max-width)
- `flex-start` / `start` / `self-start`: itens são posicionados no início do eixo transversal. A diferença entre eles é sutil e diz respeito às regras de `flex-direction` ou writing-mode
- `center`: itens são centralizados no eixo transversal
- `baseline`: itens são alinhados de acordo com suas baselines

Align-content



Align-content

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

  Importante: Esta propriedade não tem efeito quando há somente uma linha de flex items no contêiner.

```
.flex-container {
  align-content: flex-start | flex-end | center |
                 space-between | space-around | stretch;
}
```

Align-content

- `flex-start` / `start`: itens alinhados com o início do contêiner. O valor (com maior suporte dos navegadores) `flex-start` se guia pela `flex-direction`, enquanto `start` se guia pela direção do writing-mode
- `flex-end` / `end`: itens alinhados com o final do contêiner. O valor (com maior suporte dos navegadores) `flex-end` se guia pela `flex-direction`, enquanto `end` se guia pela direção do writing-mode
- `center`: itens centralizados no container

Align-content

- `space-between`: itens distribuídos igualmente; a primeira linha junto ao início do contêiner e a última linha junto ao final do contêiner
- `space-around`: itens distribuídos igualmente com o mesmo espaçamento entre cada linha
- `space-evenly`: itens distribuídos igualmente com o mesmo espaçamento entre eles
- `stretch` (padrão): itens em cada linha esticam para ocupar o espaço remanescente entre elas

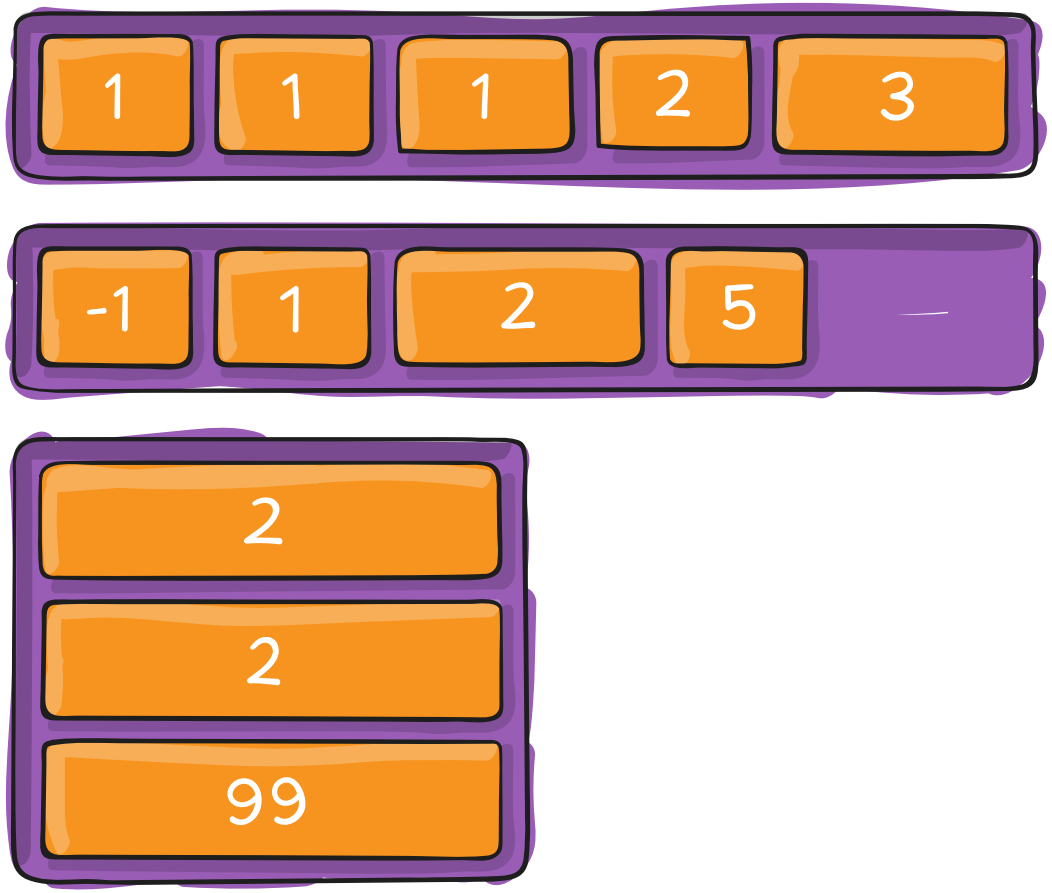
Propriedades para Elementos-filhos

- A seguir, veremos propriedades que devem ser declaradas tendo como seletor os elementos-filhos, ou seja:

```
<div class="flex-container">
  <div class="flex-item">1</div>
  <div class="flex-item">2</div>
  <div class="flex-item">3</div>
</div>
```

- Isso significa que, onde existe um elemento-pai com propriedade flex (o flex-container), é possível atribuir propriedades flex específicas também para as elementos-filhos (flex-item)

Order

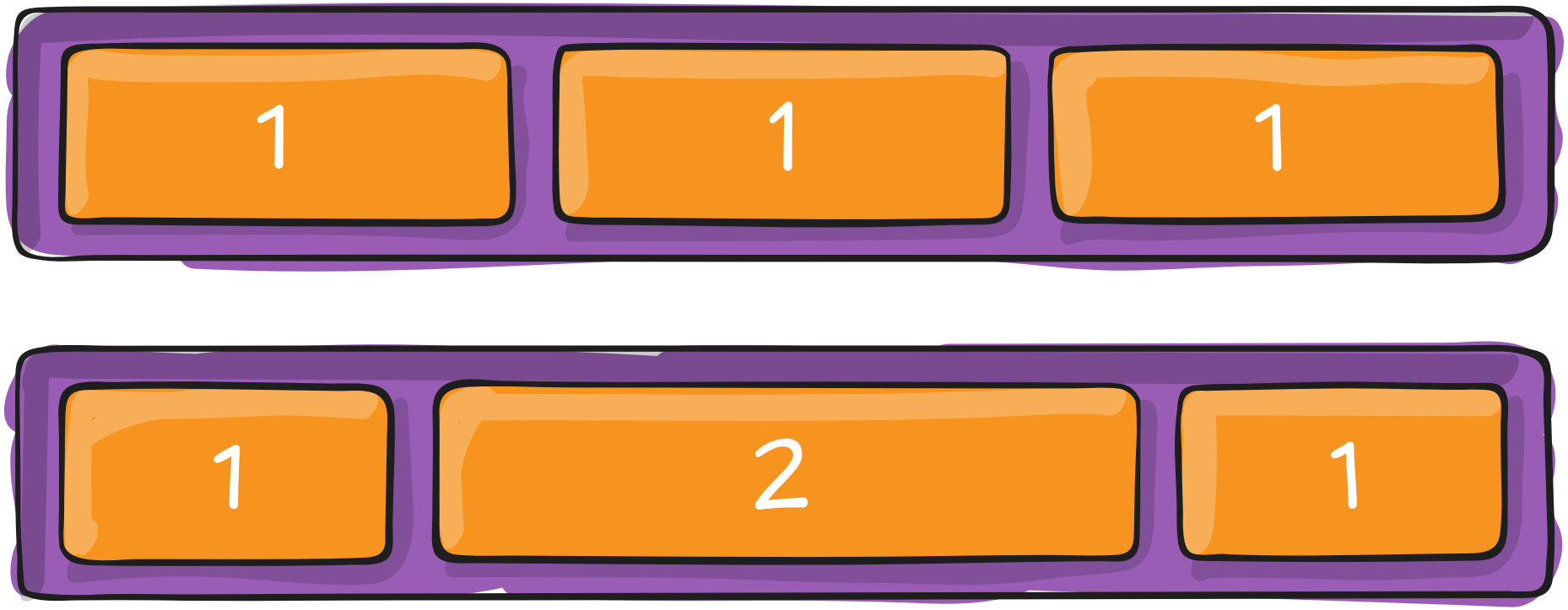


Order

- Determina a ordem em que os elementos aparecerão
- Por padrão os flex itens são dispostos na tela na ordem do código. Mas a propriedade order controla a ordem em que aparecerão no contêiner

```
.flex-item {
  order: <número>; /* o valor padrão é 0 */
}
```

Flex-grow



Flex-grow

- Define a habilidade de um flex item de crescer, caso necessário. O valor dessa propriedade é um valor numérico sem indicação de unidade, que serve para cálculo de proporção. Este valor dita a quantidade de espaço disponível no contêiner que será ocupado pelo item
- Se todos os ítens tiverem `flex-grow` definido em `1`, o espaço remanescente no container será distribuído de forma igual entre todos. Se um dos itens tem o valor de `2`, vai ocupar o dobro de espaço no container com relação aos outros (ou pelo menos vai tentar fazer isso)

Flex-grow

```
.flex-item {
  flex-grow: <numero>; /* o valor default(padrão) é 0 */
}
```

- Valores negativos não são aceitos pela propriedade

Flex-shrink

- Define a habilidade de um flex item de encolher, caso necessário

```
.flex-item {
  flex-shrink: <número>; /* o valor padrão é 0 */
}
```

- Valores negativos não são aceitos pela propriedade

Flex-basis

- Define o tamanho padrão para um elemento antes que o espaço remanescente do container seja distribuído. Pode ser um comprimento (por exemplo, 20%, 5rem, etc) ou uma palavra-chave.
- A palavra-chave `auto` significa "observe minhas propriedades de altura ou largura" (o que era feito pela palavra-chave `main-size`, que foi descontinuada). A palavra-chave `content` significa "estabeleça o tamanho com base no conteúdo interno do ítem" - essa palavra-chave ainda não tem muito suporte, então não é fácil de ser testada, assim como suas relacionadas: `max-content`, `min-content` e `fit-content`

Flex-basis

```
.flex-item {
  flex-basis: <número> | auto; /* o valor padrão é auto */
}
```

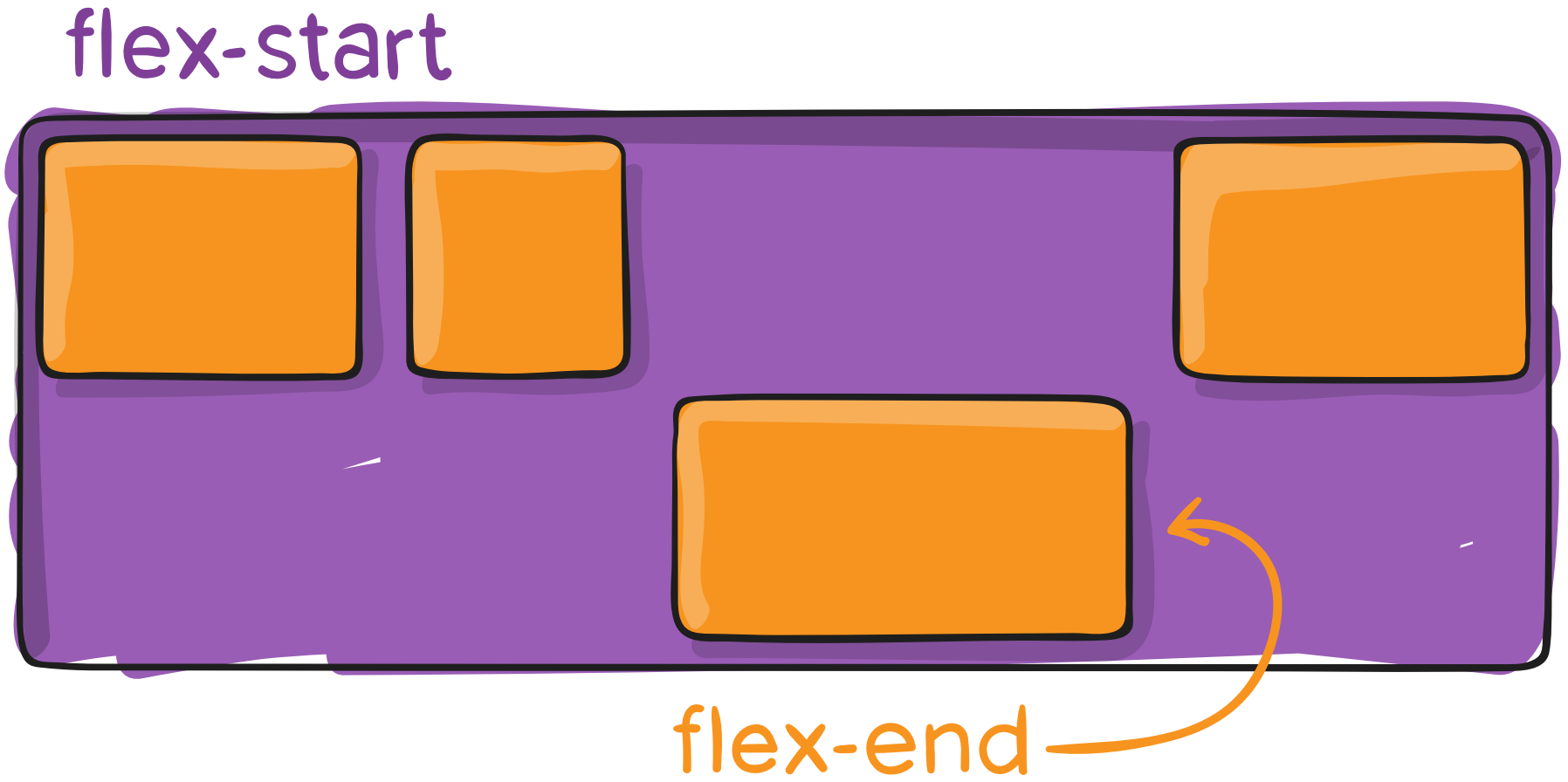
- Com o valor de `0`, o espaço extra ao redor do conteúdo não é considerado. Com o valor de `auto`, o espaço extra é distribuído com base no valor de `flex-grow` do item

Flex

- Esta é a propriedade *shorthand* para `flex-grow`, `flex-shrink` e `flex-basis`, combinadas. O segundo e terceiro parâmetros (`flex-shrink` e `flex-basis`) são opcionais. O padrão é `0 1 auto`, mas se você definir com apenas um número, é equivalente a `0 1`.

```
.flex-item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

Align-self



Align-self

- Permite que o alinhamento padrão (ou o que estiver definido por `align-items`) seja sobrescrito para itens individuais
- Veja a explicação da propriedade `align-items` para entender quais são os possíveis valores

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```

Para Praticar o Flexbox

- Flexbox Froggy: https://flexboxfroggy.com/

Referências Bibliográficas

- Castro, E. Hyslop, B. "HTML5 e CSS3 - Guia Prático & Visual", AltaBooks, 2013
- Amoasei, J. Site da Alura. "Flexbox CSS: Guia Completo, Elementos e Exemplos". https://www.alura.com.br/artigos/css-guia-do-flexbox, acessado em 29/05/2023
- MDN Web Docs. "Flexbox". https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox, acessado em 29/05/2023

Referências Bibliográficas

- CSS-Tricks. "A Complete Guide to Flexbox". https://css-tricks.com/snippets/css/a-guide-to-flexbox/, acessado em 29/05/2023