Conteúdo do dia!

Site: Geração Tech
Curso: Formação em Desenvolvedor Web - Online
Livro: Conteúdo do dia!

Impresso por: JOÃO VITOR DE MELO FREITAS
Data: quinta-feira, 18 jul. 2024, 22:33

Índice

1. Propriedades CSS Adicionais

Além das propriedades básicas que vimos anteriormente, o CSS oferece uma ampla gama de propriedades para estilizar e manipular o layout dos elementos HTML.

Aqui estão algumas propriedades CSS que vimos e mais algumas adicionais e suas funções:

- `display`: Controla como um elemento é exibido na página (por exemplo, `block`, `inline`, `flex`, `grid`).
- `position`: Define o método de posicionamento de um elemento (por exemplo, `static`, `relative`, `absolute`, `fixed`, `sticky`).
- `flex`: Utilizada para criar layouts flexíveis com o modelo de caixa flexível (Flexbox).
- `grid`: Utilizada para criar layouts em grade com o modelo de grade CSS.
- `align-items` **e** `justify-content`: Controlam o alinhamento de itens em contêineres flexíveis e de grade.
- `float` e `clear`: Controlam o posicionamento flutuante dos elementos.
- `z-index`: Controla a ordem de empilhamento de elementos posicionados.
- `overflow`: Controla o que acontece quando o conteúdo de um elemento é muito grande para caber em seu bloco (por exemplo, `visible`, `hidden`, `scroll`, `auto`).
- `box-shadow`: Adiciona sombras às caixas dos elementos.
- `text-shadow`: Adiciona sombras ao texto.

2. Aula 01

dia 08 video 01

3. Conceitos básicos de flexbox

O *Flexible Box Module*, geralmente chamado de *flexbox*, foi projetado tanto como um modelo de *layout* unidimensional quanto como um método capaz de organizar espacialmente os elementos em uma interface, além de possuir capacidades avançadas de alinhamento. Vamos aprender as principais funcionalidades do *flexbox*, as quais exploraremos com mais detalhes no restante deste guia.

Quando se descreve o flexbox como sendo unidimensional, enfatiza-se o fato de que ele lida com *o layout* em uma dimensão de cada vez - seja uma linha ou uma coluna. Isto pode ser comparado com o modelo bidimensional de CSS - Layout de Grade, que permite o controle simultâneo das colunas e linhas.

Ao se utilizar o *flexbox*, é preciso ter em mente que todas as operações realizadas relacionam-se a dois eixos: o eixo principal e o eixo transversal. O eixo principal é definido através da propriedade `flex-direction` e o eixo transversal encontra-se na direção perpendicular a ele. Como esses eixos são as engrenagens fundamentais do flexbox é necessário compreender minuciosamente o seu funcionamento.

3.1. Eixo principal

Conforme descrito acima, a propriedade `flex-direction` define a direção do eixo principal e pode tem quatro valores possíveis:

- `row`
- `row-reverse`
- `column`
- `column-reverse`

Se o valor escolhido for `row` (linha) ou `row-reverse` (linha reversa), seu eixo principal se moverá ao longo da linha — na **direção inline**.

Se o eixo principal for definido nas colunas, como `column` ou `column-reverse`, então o eixo transversal estará na direção das linhas, como `row` ou `row-reverse`.

Compreender a diferença entre os eixos principal e perpendicular é o que importa quando começamos a observar o alinhamento ou justificação dos itens flexíveis (flex items); o *flexbox* possui propriedades que alinham e justificam o conteúdo ao longo de um eixo ou de outro.

3.2. Sintaxe

A propriedade CSS `display` é especificada usando valores de palavra-chave.

```
/* valores pré-compostos */
display: block;
display: inline;
display: inline-block;
display: flex;
display: inline-flex;
display: grid;
display: inline-grid;
display: flow-root;

/* geração de caixas */
display: none;
display: contents;

/* sintaxe de dois valores */
display: block flow;
display: inline flow;
display: inline flow-root;
display: block flex;
display: inline flex;
display: block grid;
display: inline grid;
display: block flow-root;

/* outros valores */
display: table;
display: table-row; /* todos os elementos da tabela têm um valor de exibição CSS equivalente */
display: list-item;

/* Valores globais */
display: inherit;
display: initial;
display: revert;
display: revert-layer;
display: unset;
```

4. Flex Container

O `display: flex` funciona de uma maneira única comparado a outros tipos de display. Quando aplicamos essa propriedade a um elemento, ele se transforma em um **flex container**. A partir desse ponto, podemos manipular todos os elementos filhos desse container com novas propriedades específicas do Flexbox. Essas propriedades são aplicadas diretamente no elemento que é o flex container.

Por padrão, quando utilizamos `display: flex` em um elemento, todos os seus filhos são alinhados lado a lado, de maneira semelhante ao comportamento de elementos com `display: inline`.

4.1. Propriedades de um Flex Container

`justify-content`

Essa propriedade alinha os elementos filhos horizontalmente dentro do flex container.

- `flex-start`: Este é o valor padrão. Os elementos são alinhados à esquerda do flex container, um ao lado do outro.
- `flex-end`: Os elementos são alinhados à direita do flex container, um ao lado do outro.
- `center`: Os elementos são centralizados no flex container, um ao lado do outro.
- `space-between`: O primeiro elemento é alinhado à esquerda e o último à direita, com os elementos intermediários distribuídos com espaçamento igual entre eles.
- `space-around`: Cada elemento tem um espaçamento igual ao redor de si. Isso significa que o primeiro elemento terá um espaçamento maior à direita do que à esquerda, somando com o espaçamento à esquerda do segundo elemento.
- `space-evenly`: Resolve o "problema" do valor anterior. Os elementos têm um espaçamento igual em ambos os lados.

`align-items`

Essa propriedade alinha os elementos filhos verticalmente dentro do flex container.

- `stretch`: Este é o valor padrão. Os elementos são "esticados" para terem a mesma altura.
- `flex-start`: Os elementos são alinhados ao topo do flex container.
- `flex-end`: Os elementos são alinhados à base do flex container.
- `center`: Os elementos são centralizados verticalmente no flex container.
- `baseline`: Os elementos são alinhados com a linha base do conteúdo textual de cada um.

`flex-wrap`

Essa propriedade controla a quebra de linha dos elementos filhos dentro do flex container.

- `nowrap`: Este é o valor padrão. Os elementos são mantidos em uma única linha, mesmo que não haja espaço horizontal suficiente.
- `wrap`: Os elementos que não cabem mais na linha atual são movidos para a linha seguinte.
- `wrap-reverse`: Os elementos que não cabem mais na linha atual são movidos para a linha acima.

5. Funcionamento e construção

Para adicionar o Flexbox no seu projeto, é necessário definir a propriedade **display: flex** dentro de um seletor, que pode ser chamado de **container**. Este será o elemento pai e todos os seus elementos filhos serão itens flexíveis.

Basicamente, o código ficaria desta forma no seu arquivo CSS:

```
.container{

display: flex;

}
```

E da seguinte forma no seu arquivo HTML:

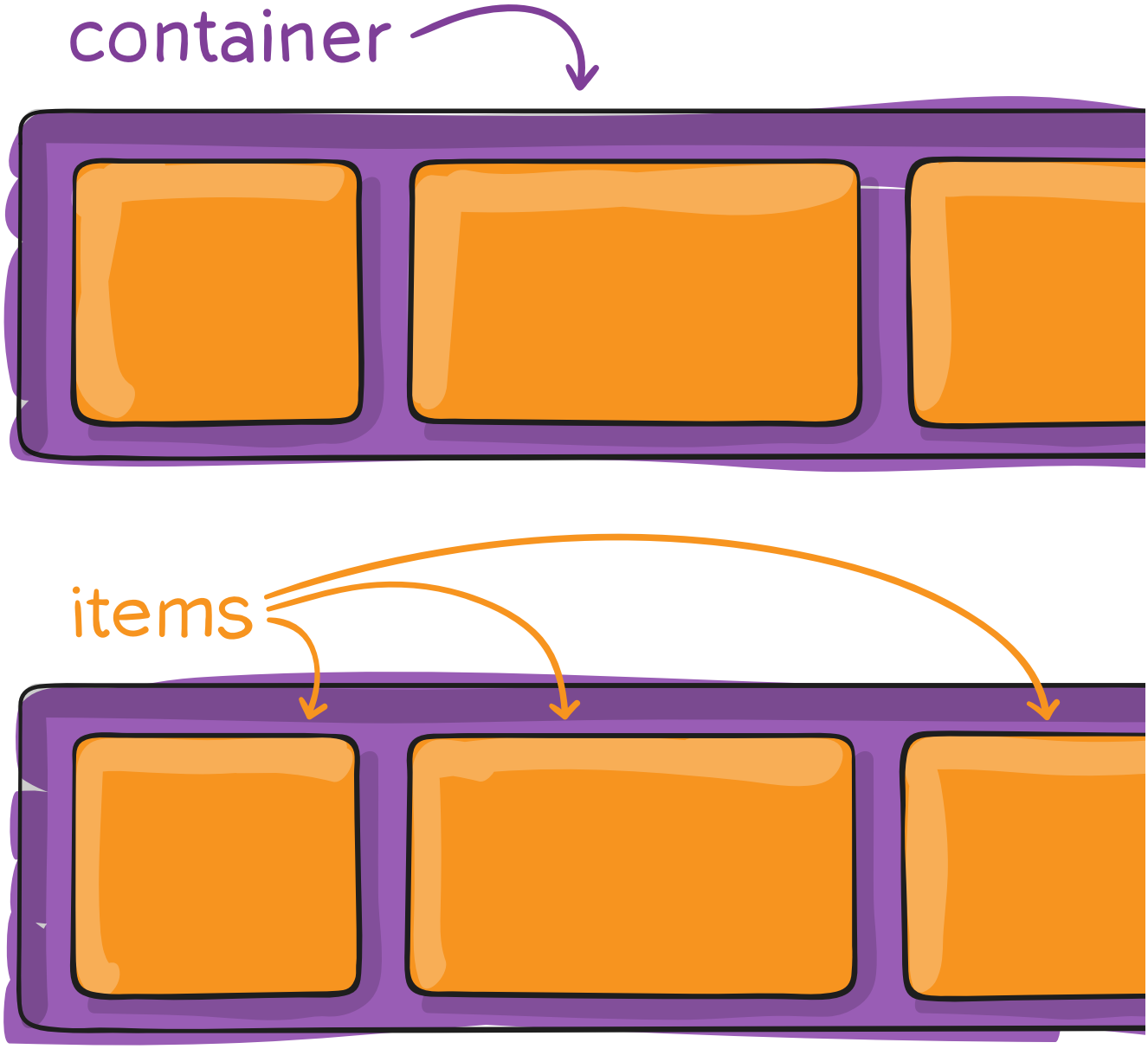
```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de Flexbox</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
        <div class="item">Item 5</div>
    </div>
</body>
</html>
```

Explicação do Código

- **HTML**: Criamos uma estrutura básica com uma `div` com a classe `container` que contém cinco `div` filhos com a classe `item`.
- **CSS**:
    - **body**: Definimos o `body` como um flex container para centralizar o `.container` tanto horizontalmente (`justify-content: center`) quanto verticalmente (`align-items: center`).
    - **.container**: Aplicamos `display: flex` ao container, permitindo que os elementos filhos se alinhem de acordo com as propriedades do Flexbox.

5.1. Continuação

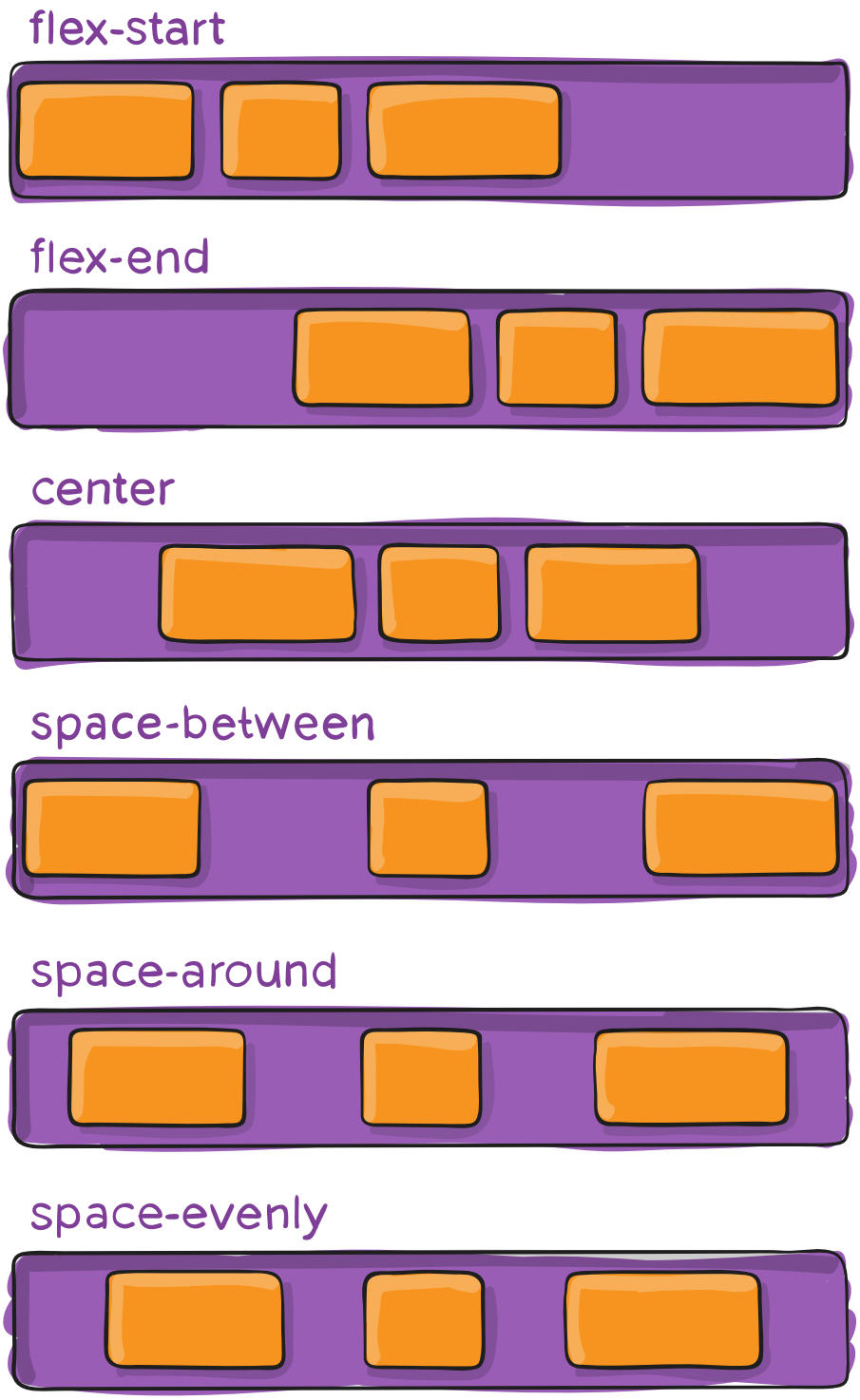
Temos que saber que teremos propriedades CSS para trabalhar com o elemento que possui nossos itens (container ou elemento pai) e propriedades para os nossos itens (elementos filhos).



5.2. Aula 02

6. Justify-content

O **justify-content** é uma propriedade que alinha os itens de um *container* com base no eixo principal. São possíveis posições: **flex-start**, **flex-end**, **center**, **space-between**, **space-around** e **space-evenly**.



6.1. Valores da Propriedade justify-content

Aqui estão os valores possíveis para a propriedade `justify-content` e uma explicação detalhada de cada um:

1. `flex-start`:
   - **Descrição**: Este é o valor padrão.
   - **Comportamento**: Os itens são alinhados no início do contêiner. Se o contêiner é horizontal, os itens serão empurrados para a esquerda. Não há espaço entre os itens e a borda esquerda do contêiner.
   - **Uso**: Ideal quando se deseja que os itens se alinhem no início, sem espaço inicial.
2. `flex-end`:
   - **Descrição**: Os itens são alinhados no final do contêiner.
   - **Comportamento**: Os itens são empurrados para a direita, com qualquer espaço extra colocado antes dos itens.
   - **Uso**: Utilizado quando se quer que os itens se acumulem no final do contêiner.
3. `center`:
   - **Descrição**: Os itens são centralizados no contêiner.
   - **Comportamento**: Os itens são alinhados no centro, com espaço igual antes e depois deles.
   - **Uso**: Útil para centralizar visualmente os itens no contêiner.
4. `space-between`:
   - **Descrição**: Os itens são distribuídos com o máximo de espaço possível entre eles.
   - **Comportamento**: O primeiro item é alinhado no início do contêiner e o último item no final, com espaço igual entre os itens intermediários.
   - **Uso**: Ideal para layouts onde se deseja um espaçamento uniforme entre os itens, com os itens das extremidades colados nas bordas do contêiner.
5. `space-around`:
   - **Descrição**: Os itens são distribuídos com espaço igual ao redor deles.
   - **Comportamento**: Cada item tem um espaçamento igual ao seu redor. Isso resulta em um espaço maior nas extremidades do contêiner porque o primeiro item terá espaço à sua esquerda e o último item terá espaço à sua direita.
   - **Uso**: Bom para quando se quer um espaçamento uniforme, mas com alguma margem nas bordas do contêiner.
6. `space-evenly`:
   - **Descrição**: Os itens são distribuídos com espaço igual entre eles, incluindo nas extremidades do contêiner.
   - **Comportamento**: Todos os espaços entre os itens, bem como antes do primeiro item e depois do último item, são iguais.
   - **Uso**: Excelente para garantir um espaçamento absolutamente uniforme em todo o contêiner.

6.2. Exemplos Visuais

Para facilitar a compreensão, vejamos exemplos visuais de como `justify-content` afeta o layout dos itens em um contêiner flexível.

HTML de Base

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de Justify Content</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container justify-start">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
    </div>
</body>
</html>
```

CSS para Demonstração

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    font-family: Arial, sans-serif;
}

.container {
    display: flex;
    width: 80%;
    border: 1px solid #ccc;
    padding: 10px;
    margin-bottom: 20px;
}

/* Justify Content Examples */
.justify-start {
    justify-content: flex-start;
}

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

.justify-center {
    justify-content: center;
}

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

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

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

/* Item Styles */
.item {
    background-color: #4CAF50;
    color: white;
    padding: 15px;
    text-align: center;
    margin: 5px;
    border-radius: 5px;
}
```

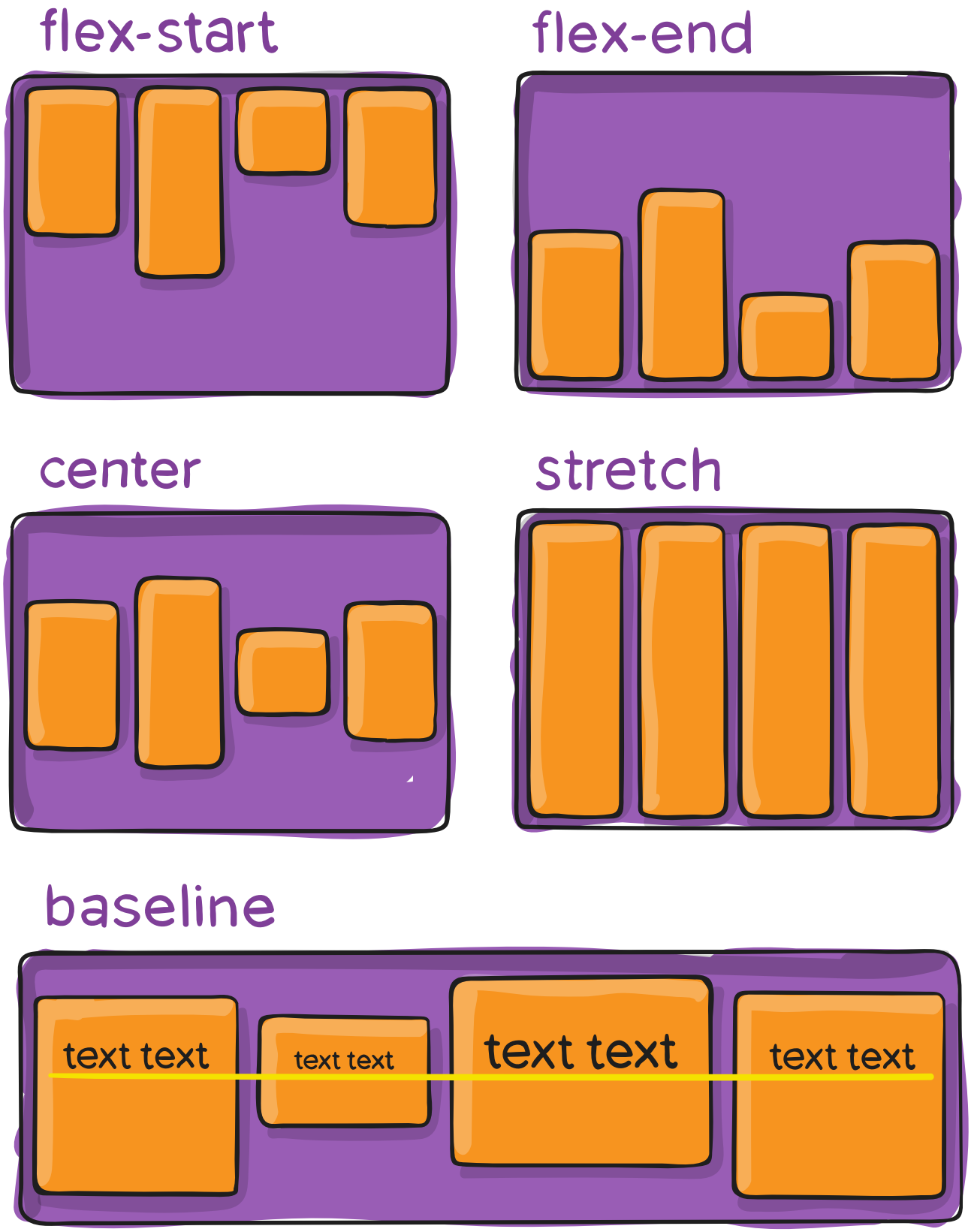
Modificações no HTML para Demonstração de Cada Valor

Para ver os diferentes comportamentos, você pode alterar a classe do contêiner para `justify-start`, `justify-end`, `justify-center`, `justify-space-between`, `justify-space-around`, ou `justify-space-evenly`.

A propriedade `justify-content` é fundamental para controlar o espaçamento e alinhamento dos itens dentro de um contêiner flexível ao longo do eixo principal. Compreender como cada valor funciona permite criar layouts flexíveis e responsivos que se adaptam às necessidades de design e usabilidade do projeto.

7. Align-items

O **align-items** funciona semelhante ao **justify-content**, porém define a disposição dos itens flexíveis ao longo do eixo transversal, em vez do eixo principal, como o **justify content**. Os valores possíveis são: **stretch**, **flex-end**, **flex-start**, **center** e **baseline**.



7.1. Valores

Aqui estão os valores possíveis para a propriedade `align-items` e uma explicação detalhada de cada um:

1. `stretch`:
   - **Descrição**: Este é o valor padrão.
   - **Comportamento**: Os itens são esticados para preencher o contêiner, de modo que cada item tenha a mesma altura que o contêiner. Isso é possível quando os itens têm uma altura definida ou podem ser esticados.
   - **Uso**: Ideal para garantir que todos os itens tenham a mesma altura, preenchendo todo o contêiner.
2. `flex-start`:
   - **Descrição**: Os itens são alinhados ao início do contêiner.
   - **Comportamento**: Os itens são alinhados na parte superior do contêiner. Se o contêiner é horizontal, os itens são alinhados ao topo.
   - **Uso**: Utilizado quando se deseja que os itens se alinhem no topo do contêiner.
3. `flex-end`:
   - **Descrição**: Os itens são alinhados ao final do contêiner.
   - **Comportamento**: Os itens são alinhados na parte inferior do contêiner. Se o contêiner é horizontal, os itens são alinhados à base.
   - **Uso**: Útil quando se quer que os itens se acumulem na base do contêiner.
4. `center`:
   - **Descrição**: Os itens são centralizados no contêiner.
   - **Comportamento**: Os itens são alinhados no centro do contêiner, verticalmente.
   - **Uso**: Ideal para centralizar visualmente os itens no contêiner verticalmente.
5. `baseline`:
   - **Descrição**: Os itens são alinhados com suas linhas de base.
   - **Comportamento**: Os itens são alinhados de forma que suas linhas de base (baseline) se alinhem. Isso é particularmente útil quando os itens contêm texto de diferentes tamanhos.
   - **Uso**: Excelente para garantir que o texto dentro dos itens esteja alinhado.

7.2. Exemplos

Para facilitar a compreensão, vejamos exemplos visuais de como `align-items` afeta o layout dos itens em um contêiner flexível.

HTML de Base

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de Align Items</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container align-start">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
        <div class="item">Item 3</div>
    </div>
</body>
</html>
```

CSS para Demonstração

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    font-family: Arial, sans-serif;
}

.container {
    display: flex;
    flex-direction: column; /* Para visualizar a propriedade align-items melhor */
    width: 50%;
    height: 300px;
    border: 1px solid #ccc;
    padding: 10px;
    margin-bottom: 20px;
}

/* Align Items Examples */
.align-start {
    align-items: flex-start;
}

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

.align-center {
    align-items: center;
}

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

.align-baseline {
    align-items: baseline;
}

/* Item Styles */
.item {
    background-color: #4CAF50;
    color: white;
    padding: 15px;
    text-align: center;
    margin: 5px;
    border-radius: 5px;
}

.item:nth-child(2) {
    font-size: 24px; /* Alterar a altura do texto para visualizar align-baseline */
}
```

Modificações no HTML para Demonstração de Cada Valor

Para ver os diferentes comportamentos, você pode alterar a classe do contêiner para `align-start`, `align-end`, `align-center`, `align-stretch`, ou `align-baseline`.

Explicação dos Exemplos

- **`.container`**: A configuração `flex-direction: column` é usada para alinhar os itens ao longo do eixo vertical. O contêiner é dado uma altura fixa para demonstrar o efeito de `align-items`.
- **`.align-start`**: Os itens são alinhados ao topo do contêiner.
- **`.align-end`**: Os itens são alinhados à base do contêiner.
- **`.align-center`**: Os itens são centralizados verticalmente no contêiner.
- **`.align-stretch`**: Os itens são esticados para preencher a altura do contêiner, mantendo a altura igual.
- **`.align-baseline`**: Os itens são alinhados com base nas suas linhas de base de texto, útil para alinhar textos de diferentes tamanhos.

A propriedade `align-items` é crucial para controlar o alinhamento vertical dos itens dentro de um contêiner flexível ao longo do eixo transversal. Compreender como cada valor funciona permite criar layouts mais sofisticados e bem organizados, garantindo que os itens dentro de um contêiner flexível se comportem de maneira previsível e esteticamente agradável.

7.3. Aula 03

dia 08 video 03