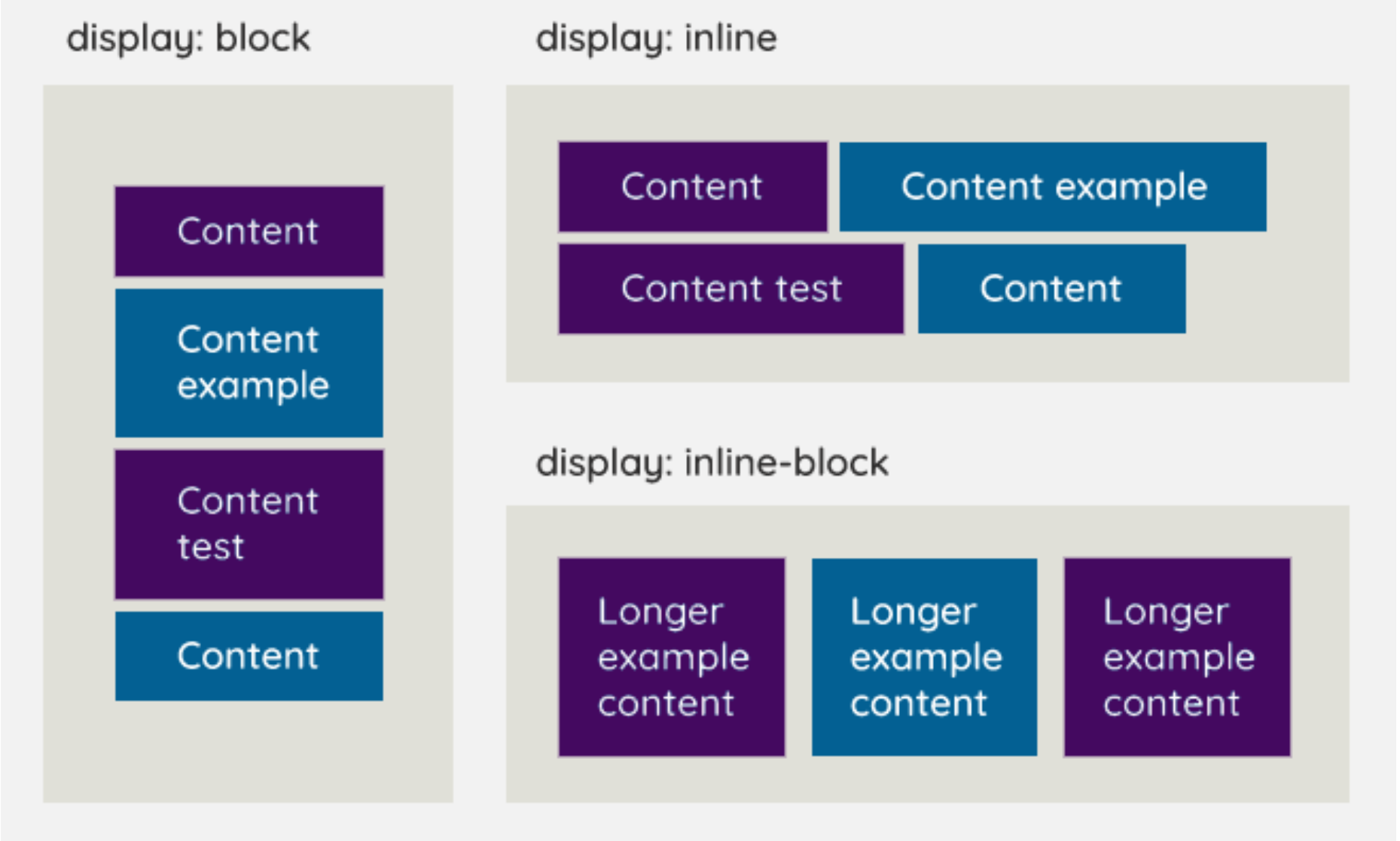
Prof.: Paulo Júnior

O Display

- **é responsável pelo comportamento dos elementos**
- **esse comportamento está ligado ao tipo de caixa de renderização usada por cada elemento**
- **cada elemento tem seu comportamento padrão ou está inserido em uma caixa de renderização especifica**
- **a grande maioria dos elementos do HTML tem seu comportamento padrão como BLOCK**

O Display



Porque o Flex?

- **por um longo tempo, as únicas ferramentas compatíveis entre browsers disponíveis para criação de layouts CSS eram coisas como floats e positions**

- **elas são boas e funcionam, mas em alguns casos também são limitadas e frustrantes**

- **existem vários requisitos de layouts que elas são difíceis de gerar ou impossíveis de se conseguir posicionar**

Porque o Flex?

- **centralizar um bloco de conteúdo verticalmente dentro de seu pai**
- **fazer com que os filhos de um pai ocupe uma quantidade igual de largura/altura disponível, independente da quantidade de largura/altura disponível**
- **Fazer todas as colunas de um layout com múltiplas colunas adotem a mesma altura, mesmo que contenham uma quantidade diferente de conteúdo**

O Flex

- **o Flex é um conceito do CSS3**
- **Traz no seu comportamento uma organização dos elementos de uma página HTML dentro de seus pais de forma dinâmica**
- **Essa organização ignora as dimensões, ele sempre vai manter o comportamento do layout flexível e dentro de seu pai, reorganizando-se de acordo com a necessidade**

O inicio!

- **o flex possui um conjunto de propriedades**
- **elas tem por objetivo organizar itens dentro de um elemento pai, que pode ser chamado de container(semântica)**
- **com isso para utilizar todo o poder do flex é necessário ter no seu HTML um elemento container(pai) contendo os itens(filhos)**

O inicio!

- **para entender o funcionamento do flex você precisa ter a visão da distribuição dos elementos no container(pai)**

- **O principal conceito é que existe um eixo principal e eixo transversal, esses eixos estão diretamente ligados à propriedade flex-direction**

- **com o valor row ou row-reverse os elementos se organizam em linha e o eixo principal do container(pai) fica na horizontal**

- **já se o valor for column ou column-reverse os elementos se organizam em coluna e o eixo principal do container(pai) fica na vertical**

As propriedades

- display
- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-content
- align-items
- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

O display

- **O começo de tudo para o utilização do flex é definir a propriedade display do container(pai) com o valor flex**

- **Isso é necessário para que as demais propriedades apresentem o resultado esperado**

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

O flex-direction

- o flex-direction deve ser aplicado ao container(pai)
- como dito anteriormente ele define o eixo(fluxo) de exibição dos elementos
- se você não aplicar o valor padrão é row

```
.container {
    display: flex;
    flex-direction: [row / row-reverse / column / column-reverse];
}
```

O flex-wrap

- o padrão de comportamentos dos itens no container(pai) é de se ajustarem em uma única linha

- com a propriedade flex-wrap o container(pai) perde o comportamento padrão e faz a “quebra de linha” dos itens

```
.container {
    display: flex;
    flex-wrap: [nowrap / wrap / wrap-reverse];
}
```

O flex-flow

- esta propriedade abrevia a esrita do flex
- nela ficam contidas as propriedades flex-direction e flex-wrap
- seguindo está ordem

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

O justify-content

- **a propriedade justify-content define o alinhamento dos itens tendo como base o eixo principal do container(pai)**

```
.container {
    display: flex;
    justify-content: [flex-start/flex-end/center/space-between/space-around];
}
```

O align-content

- para utilizar o poder dessa propriedade o flex-wrap:wrap precisa ter sido definido
- essa propriedade trabalha o comportamento das linhas no eixo transversal
- só tem efeito se o layout possuir mais de uma linha

```
.container {
    display: flex;
    align-content: [stretch/flex-start/flex-end/center/space-between/space-around];
}
```

O align-items

- **essa propriedade define como os itens são distribuídos ao longo do eixo transversal do container(pai)**

```
.container {
  display: flex;
  align-items: [stretch/flex-start/flex-end/center/baseline];
}
```

O order

- **O padrão de distribuição do flex é o mesmo de como os elementos são distribuídos no seu HTML**
- **Essa propriedade pode alterar a ordem de distribuição**
- **Tipo se você colocar o numero 2, leva para segunda posição no eixo**

```
.item2 {
  order: [número];
}
```

O flex-grow

- esta propriedade define a proporção com que um item deve crescer caso seja necessário
- o padrão de crescimento é 0, o que indica que o item não deve crescer
- só são aceitos valores numéricos positivos

```
.item2 {
  flex-grow: [número];
}
```

O flex-shrink

- esta propriedade define a proporção com que um item deve encolher caso seja necessário
- o padrão é 1
- só são aceitos valores numéricos positivos

```
.item2 {
  flex-shrink: [número];
}
```

O flex-basis

- ela define o tamanho inicial que um item deve ter antes que o espaço ao seu redor seja distribuído
- Ela tem relação direta com a direção do eixo principal (horizontal ou vertical)
- ela define a largura ou altura mínima do elemento antes que ele seja redimensionado por outras propriedades
- O valor atribuído a essa propriedade pode ser em percentual, em pixels, ou a palavra auto

```
.item2 {
  flex-basis: [número];
}
```

O flex

- esta propriedade abrevia a esrita
- nela ficam contidas as propriedades flex-grow, flex-shrink e flex-basis
- seguindo está ordem

```
.item2 {
  flex: [flex-grow] [flex-shrink] [flex-basis];
}
```

O align-self

- **esta propriedade permite sobrescrever no item o comportamento que foi definido pela propriedade align-items**

```
.item2 {
    align-self: [auto/stretch/flex-start/flex-end/center/baseline];
 }
```