# Propriedades

# Cor de texto

- a propriedade `color` define a cor de um texto;
- a cor padrão de todos os textos é definida no `body`;
- exemplo: `body { color: #444444 }`

# Cor de fundo

- a propriedade `background-color` define a cor de fundo de um elemento;
- a cor de fundo da página pode ser definida no `body`;
- exemplo: `body { background-color: #f3f3f3 }`

# Imagem de fundo

- a propriedade `background-image` define uma imagem de fundo para um elemento;
- por padrão, a imagem se repete para cobrir toda área do elemento;
- utilize imagens que não causem distorções nos textos que sobrepõe;
- também utilizado para gradientes;
- exemplo:

```
div { background-image: url("paper.gif") }
div { background-image: linear-gradient(to bottom right, red, yellow) }
```

# Imagem de fundo - Repetição

- a propriedade `background-repeat` controla o comportamento da imagem de fundo;
- valores: **`repeat`**, `repeat-x`, `repeat-y`, `no-repeat`, `space`, `round`;
- exemplo: `div { background-repeat: no-repeat }`

# Imagem de fundo - Anexo

- a propriedade `background-attachment` define se a imagem de fundo deverá ficar fixa ou rolar junto com a página;
- valores: **`scroll`**, `fixed`, `local`;
- exemplo: `div { background-attachment: fixed }`

# Imagem de fundo - Posição

- a propriedade `background-position` define onde a imagem de fundo deve ser apresentada;
- valores: [`left`, `right`, `right`] [`top`, `center`, `bottom`], `x% y%`, `xpos ypos`;
- exemplo: `div { background-position: right top }`

# Imagem de fundo - Tamanho

- a propriedade `background-size` define o tamanho da imagem de fundo;
- possui três sintaxes:
  - palavra-chave: `auto`, `cover`, `contain`;
  - largura (altura será `auto`): `length`;
  - largura e altura: `x-length y-length`;
- exemplo: `div { background-size: 200px 300px }`

# Fundo (abreviação)

- a propriedade `background` permite definir através de uma única sintaxe várias propriedades do fundo;
- valores: `color image position/size repeat origin clip attachment`;
- exemplo: `div { background: #ffffff url("dog.png") no-repeat right top/100px }`

# Bordas - Espessura

- a propriedade `border-width` define a espessura da borda do elemento;
- possui quatro sintaxes:
  - todos lados: `width`;
  - topo+inferior e direita+esquerda: `top-bottom right-left`;
  - topo, direita+esquerda e inferior: `top right-left bottom`;
  - lados individuais: `top right bottom left`;
- exemplo: `div { border-width: 2px }`

# Bordas - Estilo

- a propriedade `border-style` define o estilo da borda;
- valores: **`solid`**, `dotted`, `dashed`, `double`, `none`, `hidden`;
- exemplo: `div { border-style: dashed }`

# Bordas - Cor

- a propriedade `border-color` define a cor da borda;
- possui quatro sintaxes:
  - todos lados: `color`;
  - topo+inferior e direita+esquerda: `top-bottom right-left`;
  - topo, direita+esquerda e inferior: `top right-left bottom`;
  - lados individuais: `top right bottom left`;
- exemplo: `div { border-color: red green blue yellow }`

# Borda (abreviação)

- a propriedade `border` permite definir a borda através de uma única sintaxe;
- valores: `width style color`;
- é possível especificar individualmente os quatro lados;
- exemplo:

```
div { border: 5px solid red }
div { border-bottom: 2px solid blue }
```

# Bordas arredondadas

- a propriedade `border-radius` permite definir a borda através de uma única sintaxe;
- possui quatro sintaxes:
  - todos cantos: `radius`;
  - topo-esquerdo+inferior-direito e topo-direito+inferior-esquerdo: `top-left/bottom-right top-right/bottom-left`;
  - topo-esquerdo, topo-direito+inferior-esquerdo e inferior-direito: `top-left top-right/bottom-left bottom-right`;
  - lados individuais: `top-left top-right bottom-right bottom-left`;
- é possível especificar individualmente os quatro cantos;
- exemplo: `div { border-radius: 10px }`

  `div { border-top-left-radius: 20px }`

# Margens

- a propriedade `margin` é utilizada para criar um espaço entre os elementos, para fora das bordas;
- possui quatro sintaxes (segue a mesma regra das bordas);
- é possível especificar individualmente os quatro lados;
- pode possuir valores negativos;
- o valor `auto` é utilizado para centralizar o conteúdo horizontalmente;
- margens superiores e inferiores podem colapsar em uma única margem;
- exemplo:

```
div { margin: 40px 10px }
div { margin-top: 20px }
```

# Padding

- a propriedade `padding` é utilizada para criar um espaço para o conteúdo de um elemento, para dentro das bordas;
- possui quatro sintaxes (segue a mesma regra das bordas);
- é possível especificar individualmente os quatro lados;
- **não** pode possuir valores negativos;
- exemplo:

```
div { padding: 40px 30px 20px 10px }
div { padding-bottom: 20px }
```

# Altura e Largura

- as propriedades `height` e `width` são utilizadas para definir uma altura e largura para um elemento, respectivamente;
- o valor pode ser definido em qualquer unidade de medida;
- `auto` (valor padrão) é utilizado para calcular automaticamente o tamanho;
- por padrão, são utilizados para definir o tamanho do conteúdo (não incluem *padding*, bordas e margens);
- exemplo:

```
div { width: 50% }
div { height: 200px }
```

# Altura e Largura - Mínimo

- as propriedades `min-height` e `min-width` são utilizadas para definir uma altura e largura mínima;
- o elemento terá pelo menos esse tamanho, mas poderá ser maior;
- geralmente utilizado em conjunto com `height` e `width`;
- exemplo: `div { min-height: 100px }`

# Altura e Largura - Máximo

- as propriedades `max-height` e `max-width` são utilizadas para definir uma altura e largura máxima;
- o elemento terá no máximo esse tamanho, mas poderá ser menor;
- geralmente utilizado em conjunto com `height` e `width`;
- exemplo: `div { max-width: 400px }`

# Tamanho do Box

- é possível alterar o comportamento de tamanho da caixa através da propriedade `box-sizing`;
- valores: **`content-box`**, `border-box`;
- exemplo: `div { box-sizing: border-box }`

# Opacidade/Transparência

- a propriedade `opacity` define o nível de opacidade de um elemento;
- 0 = totalmente transparente, 1.0 ou 100% = totalmente opaco;
- exemplo: `div { opacity: 0.5 }`

# Alinhamento de texto

- a propriedade `text-align` é utilizada para definir o alinhamento horizontal de um texto;
- valores: **`left`**, `center`, `right`, `justify`;
- exemplo: `div { text-align: center }`

# Alinhamento vertical

- a propriedade `vertical-align` é utilizada para definir o alinhamento vertical de um elemento;
- valores: **`baseline`**, `top`, `middle`, `bottom`, `super`, `sub`;
- exemplo: `div { vertical-align: middle }`

# Família da Fonte

- a propriedade `font-family` define qual fonte de texto será utilizada;
- devem ser definidas várias fontes como fallback para garantir que a página será exibida da forma esperada;
  - inicie com a fonte específica e termine com uma fonte genérica;
- os nomes das fontes devem ser separados por “,” (vírgula);
- se o nome da fonte possuir mais de uma palavra, deve ser utilizado entre aspas: `“Times New Roman”`;
- exemplo: `body { font-family: Arial, Helvetica, sans-serif }`

# Estilo da Fonte

- a propriedade `font-style` define o estilo de fonte que será utilizado;
- valores: **`normal`**, `italic`, `oblique`;
- exemplo: `div { font-style: italic }`

# Peso da Fonte

- a propriedade `font-weight` define o peso de fonte que será utilizado;
- valores: **`normal`**, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400` (`normal`), `500`, `600`, `700` (`bold`), `800`, `900`;
- exemplo: `div { font-weight: bold }`

# Tamanho da Fonte

- a propriedade `font-size` define o tamanho de fonte que será utilizado;
- o valor pode ser definido em qualquer unidade de medida;
- exemplo: `div { font-size: 18px }`

# Fonte (abreviação)

- a propriedade `font` permite definir várias propriedades da fonte através de uma única sintaxe;
- pode ser declarado de várias formas;
- os únicos atributos obrigatórios são `font-size` e `font-family`;
- valores: `font-style font-variant font-weight font-size/line-height font-family`;
- exemplo:

```
div { font: italic small-caps bold 12px/30px Georgia, serif }
div { font: 20px Arial, sans-serif }
```

# Capitalização do texto

- a propriedade `text-transform` é utilizada para alterar a forma de escrita do texto;
- valores: **none**, `capitalize`, `uppercase`, `lowercase`;
- exemplo: `div { text-transform: uppercase }`

# Decoração do texto - Linha

- a propriedade `text-decoration-line` define o tipo de decoração que será utilizado no texto;
- valores: **none**, `underline`, `overline`, `line-through`;
- exemplo: `div { text-decoration-line: underline }`

# Decoração do texto - Cor

- a propriedade `text-decoration-color` define a cor da decoração que será utilizado no texto;
- valor: `color`;
- exemplo: `div { text-decoration-color: red }`

# Decoração do texto - Estilo

- a propriedade `text-decoration-style` define o estilo da decoração que será utilizado no texto;
- valores: **`solid`**, `double`, `dotted`, `dashed`, `wavy`;
- exemplo: `div { text-decoration-style: wavy }`

# Decoração do texto - Espessura

- a propriedade `text-decoration-thickness` define a espessura da decoração que será utilizado no texto;
- valores: **`auto`**, `from-font`, `length`;
- exemplo: `div { text-decoration-thickness: 4px }`

# Decoração do texto (abreviação)

- a propriedade `text-decoration` permite definir várias propriedades de decoração através de uma única sintaxe;
- valores: `line color style thickness`;
- exemplo: `div { text-decoration: underline blue wavy 5px }`

# Espaçamento entre linhas

- a propriedade `line-height` define a altura de uma linha;
- o valor pode ser definido em qualquer unidade de medida, ou simplesmente um número (sem unidade) que será multiplicado pelo tamanho da fonte atual;
- exemplo: `div { line-height: 1.5 }`

# Espaçamento entre caracteres

- a propriedade `letter-spacing` define o espaçamento entre os caracteres;
- o valor pode ser definido em qualquer unidade de medida;
- pode possuir valor negativo;
- exemplo: `div { letter-spacing: -0.3pt }`

# Espaço em branco

- a propriedade `white-space` define como deve funcionar os espaços em branco (espaço ou tab) dentro de um elemento;
- valores: **`normal`**, `nowrap`, `pre`, `pre-line`, `pre-wrap`;
- exemplo: `div { white-space: pre }`

# Display

- a propriedade `display` especifica o comportamento de *display* de um elemento;
- valores comuns: `block`, `inline`, `inline-block`, `flex`, `inline-flex`, `grid`, `inline-grid`, `none`, `hidden`;
- exemplo: `div { display: flex }`

# Posicionamento - Position

- a propriedade `position` especifica o método de posicionamento que o elemento irá utilizar;
- valores comuns:
  - **`static`**: valor padrão, o elemento é renderizado e posicionado conforme o fluxo do documento;
  - `absolute`: é posicionado de forma relativa ao seu primeiro ancestral não-estático;
  - `fixed`: é posicionado relativo a janela do navegador;
  - `relative`: é posicionado de forma relativa à sua posição original;
  - `sticky`: é posicionado de acordo com a posição de rolagem da página;
- exemplo: `div { position: fixed }`

# Posicionamento - Top, Right, Bottom, Left

- elementos posicionados de alguma forma podem ser reposicionados na tela através das propriedades `top`, `right`, `bottom` e `left`;
- o valor pode ser definido em qualquer unidade de medida;
- exemplo: `div { top: 20px }`

# Z-Index

- quando elementos estão posicionados de alguma forma, eles podem sobrepor outros elementos;
- a propriedade `z-index` especifica a ordem de empilhamento dos elementos (qual elemento deve ficar na frente, qual deve ficar atrás);
- o valor deve ser um número inteiro;
- pode ser negativo;
- exemplo: `div { z-index: 1000 }`

# Overflow

- a propriedade `overflow` define como o conteúdo deve transbordar o *box* ancestral, adicionando ou removendo barras de rolagem;
- valores: **`visible`**, `auto`, `scroll`, `hidden`, `clip`;
- só funciona para elementos com display *block* com um tamanho especificado;
- pode ser declarado individualmente através do `overflow-x` e `overflow-y`
- exemplo:

```
div { overflow: scroll }
div { overflow-x: hidden }
```

# `Float`

- a propriedade `float` especifica como um elemento deve flutuar;
- utilizado para posicionar e formatar o conteúdo, como por exemplo, deixar uma imagem flutuar à esquerda de um texto em um *container*;
- valores: **`none`**, `left`, `right`;
- exemplo: `div { float: left }`

# Clear

- a propriedade `clear` é geralmente utilizada no elemento subsequente ao elemento `float`, e especifica o que deve acontecer com o elemento que está flutuando;
- valores: **`none`**, `left`, `right`, `both`;
- exemplo: `div { clear: both }`

# Flexbox

- o *Flexible Box Layout* facilita através do `display: flex` e `display: inline-flex` o design flexível e responsivo sem o uso do `float` ou `position`;

# Flex Container - Direção

- a propriedade `flex-direction` especifica a direção dos itens flexíveis;
- só funcionam em conjunto com o `display: flex`;
- valores: **`row`**, `row-reverse`, `column`, `column-reverse`;
- exemplo: `div { flex-direction: column }`

# Flex Container - Quebra

- a propriedade `flex-wrap` especifica se os itens flexíveis devem ser totalmente envolvidos ou não;
- só funcionam em conjunto com o `display: flex`;
- valores: **`nowrap`**, `wrap`, `wrap-reverse`;
- exemplo: `div { flex-wrap: wrap }`

# Flex Container - Fluxo (abreviação)

- a propriedade `flex-flow` é um atalho para o `flex-direction` e `flex-wrap`;
- só funcionam em conjunto com o `display: flex`;
- valores: `flex-direction flex-wrap`;
- exemplo: `div { flex-flow: row-reverse wrap }`

# Flex Container - Alinhamento horizontal

- a propriedade `justify-content` é utilizada para alinhar os itens flexíveis de forma horizontal quando eles não utilizam todo o espaço disponível;
- valores: **`flex-start`**, `flex-end`, `center`, `space-between`, `space-around`, `space-evenly`;
- exemplo: `div { justify-content: center }`

# Flex Container - Alinhamento vertical

- a propriedade `align-items` é utilizada para alinhar os itens flexíveis de forma vertical quando eles não utilizam todo o espaço disponível;
- valores: **`stretch`**, `center`, `flex-start`, `flex-end`, `baseline`;
- exemplo: `div { align-items: center }`

# Flex Container - Alinhamento de linhas

- a propriedade `align-content` é parecida com o `align-items`, porém este é utilizado para alinhar as linhas flexíveis quando há espaço sobrando no *container*;
- precisa haver mais de uma linha flexível para essa propriedade funcionar;
- valores: **`stretch`**, `center`, `flex-start`, `flex-end`, `space-between`, `space-around`;
- exemplo: `div { align-content: space-between }`

# Flex Item - Ordem

- a propriedade `order` especifica a ordem dos itens flexíveis;
- o primeiro elemento no HTML não precisa ser o primeiro a ser renderizado;
- o valor deve ser um número inteiro;
- exemplo: `div { order: 1 }`

# Flex Item - Grow

- a propriedade `flex-grow` especifica quanto um item pode crescer em relação ao resto dos itens flexíveis dentro do mesmo *container*;
- só funciona dentro de um elemento com `display: flex`;
- o valor deve ser um número;
- exemplo: `div { flex-grow: 5 }`

# Flex Item - Shrink

- a propriedade `flex-shrink` especifica quanto um item pode encolher em relação ao resto dos itens flexíveis dentro do mesmo *container*;
- só funciona dentro de um elemento com `display: flex`;
- o valor deve ser um número;
- exemplo: `div { flex-shrink: 5 }`

# Flex Item - Basis

- a propriedade `flex-basis` especifica o tamanho inicial de um item flexível;
- só funciona dentro de um elemento com `display: flex`;
- o valor pode ser definido em qualquer unidade de medida;
- exemplo: `div { flex-basis: 200px }`

# Flex Item - Flex (abreviação)

- a propriedade `flex` é um atalho que combina o `flex-grow`, `flex-shrink` e `flex-basis`;
- só funciona dentro de um elemento com `display: flex`;
- valores: `flex-grow flex-shrink flex-basis`;
- exemplo: `div { flex-basis: 1 }`

# Flex Item - Alinhamento vertical

- a propriedade `align-self` possui o mesmo objetivo da `align-items`, porém é aplicada diretamente no ítem flexível;
- irá sobrescrever o valor do `align-items`;
- valores: **`auto`**, `stretch`, `center`, `flex-start`, `flex-end`, `baseline`;
- exemplo: `div { align-self: flex-end }`

# Links úteis

- https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Selectors
- https://www.w3schools.com/css/css_pseudo_classes.asp
- https://www.w3schools.com/css/css_pseudo_elements.asp
- https://www.w3schools.com/Css/default.asp
- https://www.w3schools.com/cssref/default.asp