**CSS – Sintaxe**

**Introdução**

No CSS trabalhando com seletores, propriedades e seus valores.

Uma sentença CSS é iniciada com { e concluída com }

Seletor {

Propriedade: valor;

Exemplo de CSS funcional:

H1 {

Font-size: 24px;

}

**Estilo no HTML**

Podemos adicionar o CSS a nossa págia HTML de três formas:

1. Diretamente ns tags, através do atributo style-CSS inline. Exemplo:  
   <h1 style=”fonte=size: 24px” > Olá </h1>
2. Na head, escrevendo o CSS dentro da tagstyle. Exemplo:  
   <head>  
   <style>  
   h1 {  
   font-size: 22px;  
   }  
   </style>  
   <head>
3. Importando o arquivo CSS. Exemplo  
   <head>  
   <link rel=”stylesheet” href=”meu\_arquivo.css” />

**Classes**

No CSS podemos estilizar os elementos através do nome da tag, do nome da classe ou pelo seu ID.

Para adicionarmos uma classe no elemento HTML usamos o atributo class:

<h1 class=”title”> Olá CSS </h1>

**Elementos**

Para selecionar os elementos que queremos, podemos usar os seletores, são eles:

Nome da tag, exemplo: h1 {...}

ID da tag, representado por #: #meuid

Classe da tag, usamos o ponto: MinhaClasse{}

Podemos selecionar mais de um elemento por vez: .title, subtitle{...}

Podemos navegar entre os elementos: .header .title{...}

Nesse exemplo, estilizamos o elemento **.title** que está dentro do elemento **.header**

**Pseudo-classes**

Podemos estilizar além dos elementos, seu estado. Fazemos isso com as pseudo-classes. Por exemplo, quando passamos o mouse sobre um link e ele muda de cor, essa mudança é feita através da estilização de pseudo-classes.  
  
seletor:estado{...}

Ou quando passamos o mouse sobre as tags h1 o texto fica na cor vermelha:  
h1:hover{color:red;}

**Confira a lista das principais pseudo-classes:**

1) :first-child – use para selecionar a primeira ocorrência do elemento.

2) :last-child – use para selecionar a última ocorrência do elemento.

3) :hover – use para estilizar o elemento quando passarmos o mouse por cima.

4) :focus – use para estilizar um elemento quando o foco estiver nele.

5) :checked – use para estilizar inputs do tipo checkbox e radio, quando estes estiverem marcados.

6) :required – use para estilizar inputs que sejam obrigatórios.

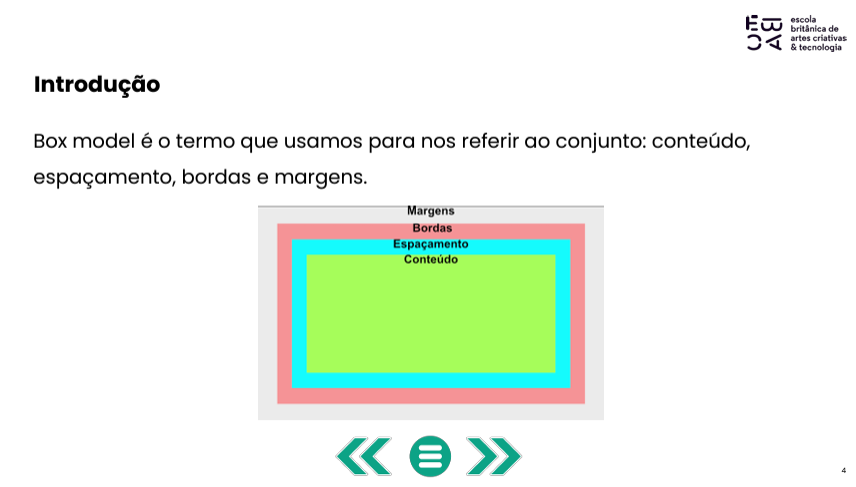
7) :optional – use para estilizar inputs que sejam opcionais.

Confira alguns exemplos de uso de pseudo-classes:

Selecionando o primeiro item de uma lista:   
ulli:first-child{...}

Selecionado um campo quando ele está com o foco:   
input:focus{...}

Selecionando um link quando passamos o mouse:   
a:hover {...}



**Propriedade margin**

Com essa propriedade aplicamos **margem** aos elementos.

Podemos adicionar margem a cada **lado** do elemento, usando uma propriedade específica:

margin-left (ou right, top, bottom)

Ou aplicar a margem em apenas uma linha de CSS:

margin: top rightbottomleft;

**Exemplos:**

section{

margin-top: 16px;

margin-bottom: 16px;

}

section{

margin: 16px 0 16px;

}

**Propriedade border**

Com essa propriedade aplicamos uma borda aos elementos. A forma mais simples de fazer essa ação é através da propriedade border:

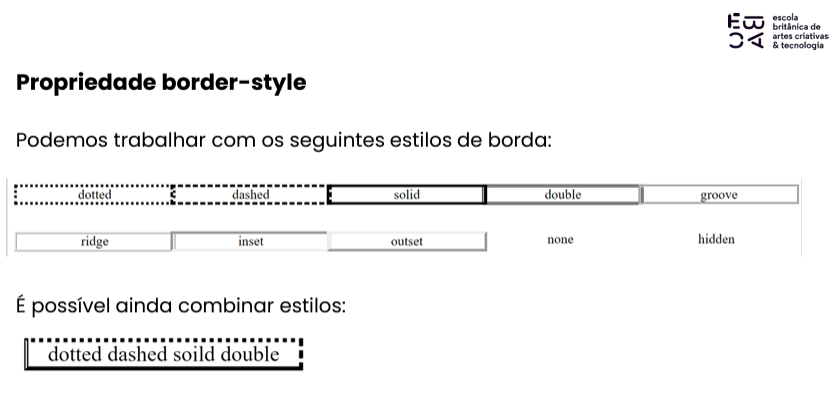
border: tamanho estilo cor;

Podemos configurar o tamanho, estilo e cor da borda de forma individual, respectivamente:

border-width: 1px;

border-style: solid;

border-color: black;



**Propriedade padding**

Com essa propriedade aplicamos espaçamento interno aos elementos. Podemos adicionar espaçamento a cada lado do elemento, usando uma propriedade específica:

Padding-left (ou right, top, bottom)

Ou aplicar o espaçamento em apenas uma linhade CSS:

padding: top right bottom left;

**Exemplos:**

section{

padding-top: 16px;

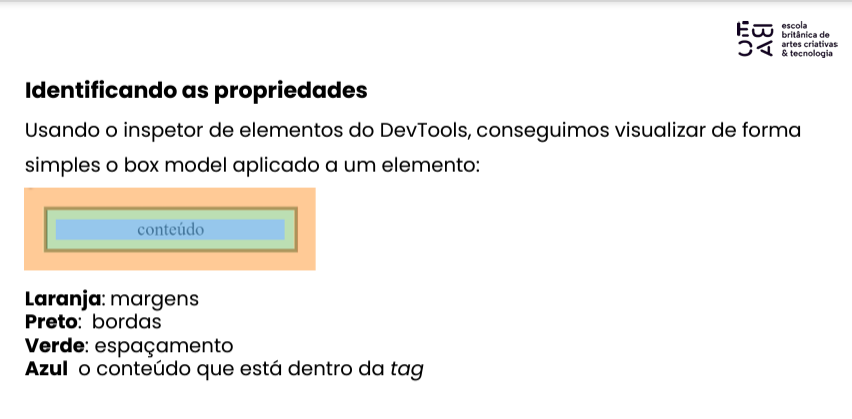
padding-bottom: 16px;

}

section{

padding: 16px 0 16px;

}



**Dimensão dos elementos**

O box model é importante para nos ajudar a calcular a dimensão de um elemento. Para isso, considere o seguinte CSS:

div{

width: 200px;

padding: 8px;

margin: 16px;

border: 4px solidblack;

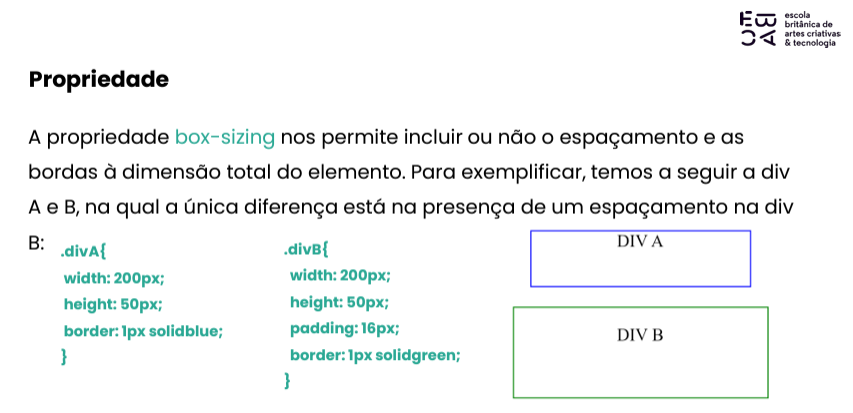
}

A largura do elemento div não será 200px, é necessário considerar as propriedades padding, margin e border neste valor. O que resultará em uma div com 224px de largura, sendo:

200 (largura inicial) + 8 (padding right) + 8 (paddin gleft) + 4 (border right) + 4 (border left)

A margem por ter um efeito externo ao elemento não afeta sua dimensão.

**Configure a página com box sizing**



Apesar de atribuirmos a largura e a altura dos elementos com width: 200px, height: 50px, a diferença entre a div A e div B é muito grande, isso por causa do efeito do espaçamento na composição do box-model. Adicionando a propriedade box-sizing com o valor border-box podemos informar que o espaçamento deve ser considerado nas medidas informadas pela largura e altura.

Os valores possíveis para a propriedade box-sizing são:

1) content-box: é o valor padrão da propriedade, a largura e altura, incluindo o conteúdo, sem considerar espaçamento e bordas.

2) border-box: é a largura e altura incluindo o conteúdo, espaçamento e bordas.

3) initial: valor padrão da propriedade.

4) inherit: o valor propriedade do elemento-pai.

**Seletor**

A propriedade box-sizing é uma das propriedades que iremos querer atribuir para todas as tags, podemos fazer isso com o seletor \*:

\* {

box-sizing: border-box;

}

**Defina o display**

**Propriedade**

Com a propriedade display informamos ao navegador como e se um elemento deve ser exibido. Os principais valores que iremos explorar são:

**1)Display: none**

Com o valor none o elemento não será exibido.

**2) Display: block**

O elemento será exibido como um bloco, não tendo nenhum outro elemento a seu lado, como se houvesse uma quebra de linha antes e depois do elemento.

**3) Display: inline**

O elemento será exibido na mesma linha, como se fosse uma tag, span, por exemplo as listas possuem seus itens separados por linhas, com o display inline sendo a aplicado na tag li veremos os itens lado a lado.

**Flexbox**

O flexbox não é uma propriedade CSS e sim um **módulo.**

É composto por algumas propriedades que exploraremos, como:

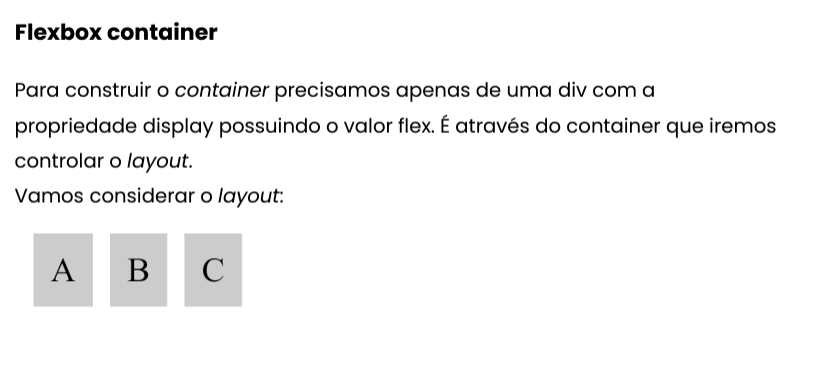
• flex-direction;

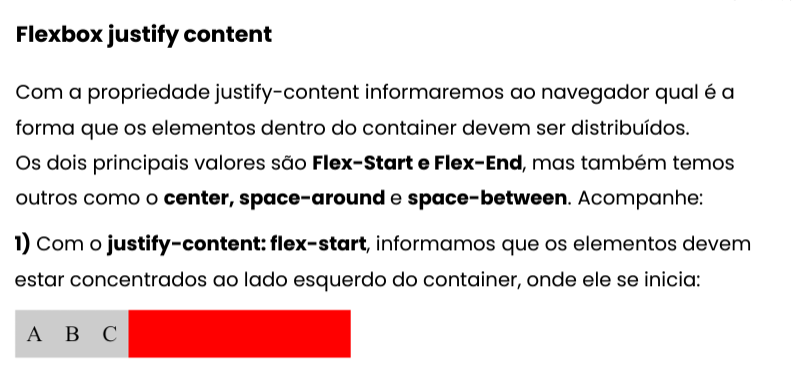
• flex-wrap;

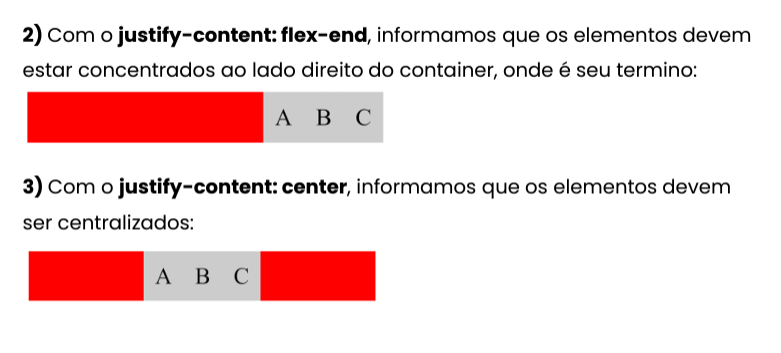
• align-items;

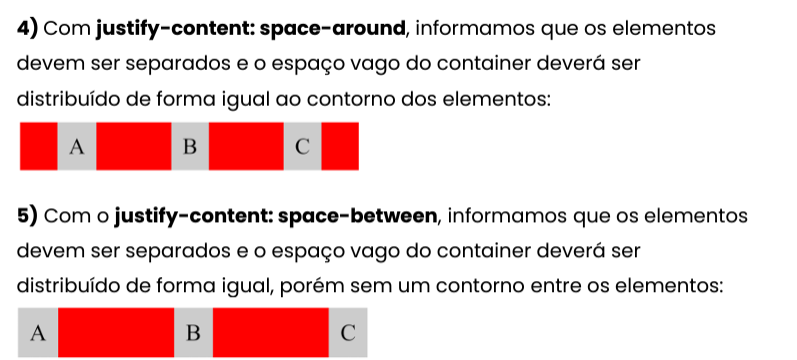
• align-content.

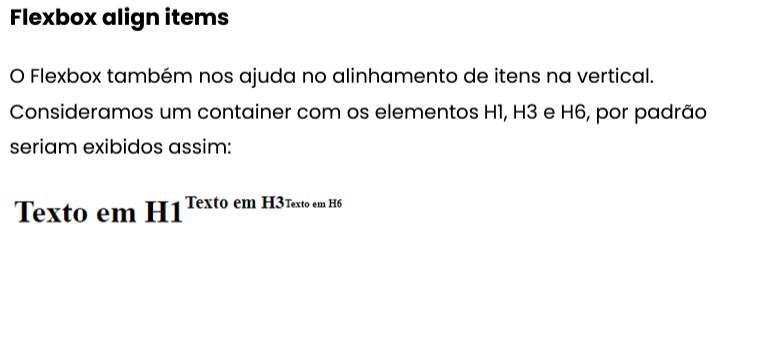
Para começarmos a trabalhar com o Flexbox temos um requisito que é a construção de um container que possua o display configurado com o valor flex.

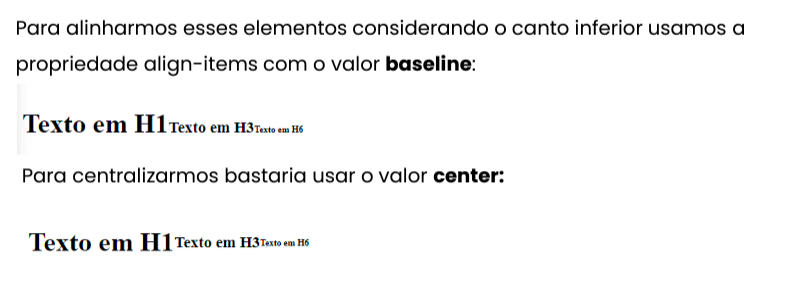


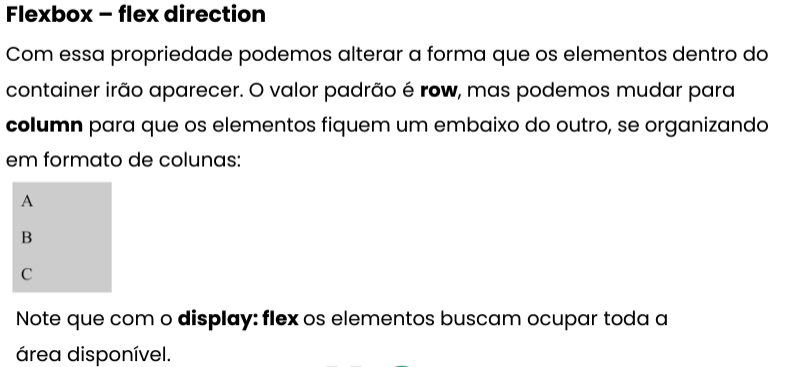












**Altere a posição (Position)**

Propriedade Para posicionar os elementos no CSS iremos usar a propriedade position, através dela diremos de qual forma queremos posicionar o elemento, se é de forma absoluta, relativa ou fixa. Além dessa propriedade devemos configurar as propriedades: top, right, left e bottom.

**Posição absoluta**

Com a posição absoluta, o elemento será posicionado de forma absoluta ao elemento que ele pertence, desde que esse elemento-pai possua a propriedade position: relative. Apesar de sua posição estática o elemento com a posição absoluta não será visível quando houver rolagem de página.

**Posição relativa**

Com a posição relativa iremos posicionar um elemento de forma relativa a sua posição inicial. Esse tipo de abordagem é bastante usada quando precisamos mover um item sem que isso desloque os elementos que estão a sua volta.

**Posição fixa**

A posição fixa irá posicionar o elemento na mesma **posição** independente do tamanho da tela e da rolagem da página. Exemplo:

.alert{

position: fixed;

bottom: 0;

right: 0;

}

O elemento está sempre no canto inferior direito da tela, independente do tamanho da tela.

**Trabalhe com cores**

**Formatos de cores**

No CSS podemos trabalhar com os formatos de cores:

• hexadecimal;

• hexadecimal com transparência;

• rgb;

• rgba;

• hsl;

• hsla.

**Aplicação de cores**

Para aplicar uma cor a um texto fazemos o uso da propriedade color:

p {

color: #333333;

}

Para aplicar uma cor de fundo ao **elemento** usamos a propriedade **background-color** ou apenas **background:**

.header {

background-color: red;

}

**Altere o texto na página**

**Alterar fonte**

Podemos alterar a fonte da nossa página utilizando a propriedade **font-family**:

p {

font-family: “Segoe UI”, arial, sans-serif;

}

Colocamos o nome da fonte entre aspas quando contém espaço.

Para alterar o tamanho da fonte, usamos a propriedade **font-size**:

p {

font-size: 16px;

}

**Alinhar textos**

Para fazer o alinhamento de texto usamos a propriedade text-align, por padrão seu valor é **left**:

p {

text-align: left OU center OU justify OU right;

}

**Aplicar estilos**

Podemos aplicar alguns estilos aos textos usando CSS, como itálico, negrito (mudança de peso da fonte), linhas ao texto, entre outros. Acompanhe!

1) **Texto em itálico**: para aplicar o itálico a um texto usamos a propriedade font-style com o valor italic.

2) **Para mudar o peso de uma fonte**: podemos mudar o peso de uma fonte usando a propriedade **font-weight**, os principais valores que temos para essa propriedade são: **lighter, normal, bold, bolder**, e de 100 ... 900 (mais eleve até o mais pesado, acrescentando-se 100).

3) **Adicionando linhas ao texto:** podemos adicionar algumas linhas para efeito aos textos, como o **underline**, usamos a propriedade **text-decoration**, onde os principais valores são **underline, line-through e overline**.

4) **Manipular o tipo de caixa dos textos:** podemos mudar a forma que o texto se apresenta, se é em CAIXA ALTA ou caixa baixa, ou capitalizado. Fazemos isso através da propriedade **text-tranform** com os respectivos valores **uppercase**, **lowercase** e capitalize.