SOFTEX
PERNAMBUCO

Softex

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

Aula 12 | Módulo: Lógica de Programação com JavaScript (continuação) e HTML + CSS / SASS (continuação)

- ➔ Modularização de código com funções e boas práticas
- ➔ Box model: content, padding, border, margin

Abrindo editor de código

Vamos agora abrir o VSCode e criar o index.html

- Com o programa aberto, clique em File > Open Folder... (Arquivo > Abrir Pasta...).
- Escolha um local para criar a sua pasta, crie uma nova pasta e dê o nome de **seunome_aula_12**. Depois dê dois clique nessa pasta criada e clique em **Selecionar pasta**. O VSCode reabrirá dentro dessa pasta que foi criada.
- Agora vamos criar o arquivo HTML e CSS:
- Dê o nome de **index.html** , **scripts.js** e **style.css**

Lógica de Programação com JavaScript (continuação)

index.html

```
index.html > ...
2   <html lang="pt-BR">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <link rel="stylesheet" href="style.css">
7       <title>Título do meu site!</title>
8   </head>
9   <body>
10      <h1>Meu site!</h1>
11      <script src="scripts.js"></script>
12  </body>
13  </html>
14
```

Lógica de Programação com JavaScript (continuação)

Revisão rápida

- Funções servem para:
  - Guardar um conjunto de instruções em um “pacote” com nome.
  - Reutilizar esse pacote sempre que precisar.
  - Organizar o código em partes menores e mais fáceis de entender.

scripts.js > ...

```
1   function dizerOla() {
2       console.log("Olá, mundo!");
3   }
4
5   dizerOla();
6   dizerOla();
7   dizerOla();
8   dizerOla();
```

Lógica de Programação com JavaScript (continuação)

Por que modularizar?

- Quebra problemas grandes em partes pequenas.
- Reutilização: evita copiar/colar (princípio DRY - Don’t Repeat Yourself).
- Leitura e manutenção: funções pequenas são fáceis de testar e trocar.
- Colaboração: cada pessoa cuida de uma parte sem conflitar.

Exemplo visual:

- Antes: 1 arquivo com 200 linhas num único script.
- Depois: 10 funções de 10~20 linhas, cada uma com uma responsabilidade.

Lógica de Programação com JavaScript (continuação)

Exercícios para praticar

1. Crie uma função chamada dobrar que receba um número como parâmetro e mostre o dobro desse número usando console.log.
2. Crie uma função chamada saudacao que receba um nome como parâmetro e mostre no console a frase: "Olá, [nome]!". Se nenhum nome for passado, deve mostrar "Olá, Visitante!".
3. Crie uma função chamada ehPar que receba um número e mostre **true** se ele for par e **false** se for ímpar.
4. Crie uma função chamada maiorNumero que receba dois números e mostre no console qual deles é o maior.
5. Crie uma função chamada contarCaracteres que receba uma palavra e mostre no console quantos caracteres ela tem. *(use **variavel.length** para descobrir o tamanho)*

HTML + CSS / SASS (continuação)

Box Model: content, padding, border, margin

- Entender a anatomia de uma “caixa” no CSS (content, padding, border, margin).
- Saber calcular tamanhos finais de elementos e evitar “quebras” de layout.
- Praticar shorthands (padding, margin, border) e box-sizing.
- Aplicar Box Model em componentes simples (cards, botões) e criar utilitários com SASS.

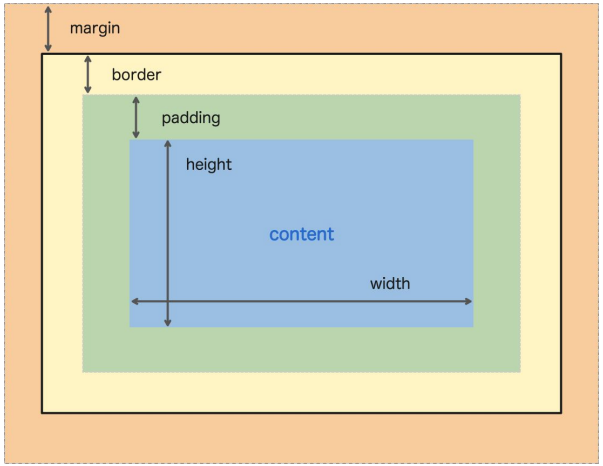
O que é o Box Model?

- Tudo é caixa no CSS (parágrafos, divs, imagens, inputs).
- Uma caixa é composta por content > padding > border > margin.
- O navegador usa o Box Model para calcular o tamanho real e o espaço ocupado na página.

HTML + CSS / SASS (continuação)

Anatomia: content, padding, border, margin

- **content**: onde ficam texto/imagens; controlado por width/height.
- **padding**: espaço interno entre o content e a borda.
- **border**: contorno visível da caixa (espessura + estilo + cor).
- **margin**: espaço externo até as outras caixas.



HTML + CSS / SASS (continuação)

Exemplo básico em código

index.html | style.css

index.html > ...

```
1   <!DOCTYPE html>
2   <html lang="pt-BR">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <link rel="stylesheet" href="style.css">
7       <title>Document</title>
8   </head>
9   <body>
10      <div class="caixa">Sou uma caixa</div>
11      <script src="script.js"></script>
12  </body>
13  </html>
14
```

HTML + CSS / SASS (continuação)

Exemplo básico em código

index.html | style.css

style.css > ...

```css
.caixa {
    width: 200px;  /* content width */
    padding: 16px;  /* espaço interno */
    border: 2px solid #555;  /* moldura */
    margin: 20px;  /* espaço externo */
    background: #f5f5f5;  /* para ajudar a visualizar o content */
}

```

HTML + CSS / SASS (continuação)

Shorthands (atalhos) úteis

- padding: 10px; (todos os lados)
- padding: 10px 20px; (top/bottom, left/right)
- padding: 10px 20px 5px; (top, left/right, bottom)
- padding: 10px 8px 6px 4px; (top, right, bottom, left)
- Mesmo padrão serve para margin.

Converta os estilos abaixo para um shorthands:

- padding-top: 8px;
- padding-right: 16px;
- padding-bottom: 8px;
- padding-left: 16px;

HTML + CSS / SASS (continuação)

Border: estilo, espessura e cor

- border: 2px solid #333;
- Pode separar: border-width, border-style, border-color.
- Cantos arredondados: border-radius: 8px;

```html
<body>
    <div class="card">Cartão</div>
    <script src="script.js"></script>
</body>
```

```css
.card {
    border: 2px solid #000000;
    border-radius: 12px;
}
```

HTML + CSS / SASS (continuação)

Margin: espaçamento externo

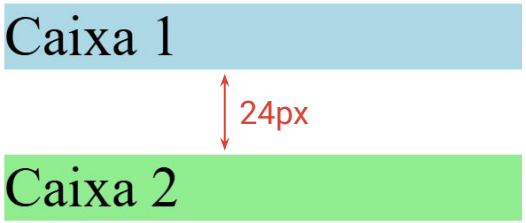
- Empurra a caixa para fora em relação às outras.
- margin: 0 auto; centraliza blocos com largura fixa.
- Colapso de margens: margens verticais adjacentes podem se fundir.

O que muita gente imagina:

Distância entre as caixas: 24px + 16px = 40px.

O que acontece de verdade:

Distância entre as caixas: 24px (apenas a maior margem).



HTML + CSS / SASS (continuação)

Margin: espaçamento externo

```html
<body>
    <div class="box1">Caixa 1</div>
    <div class="box2">Caixa 2</div>
    <script src="script.js"></script>
</body>
```

```css
.box1 {
  margin-bottom: 24px; /* margem inferior */
  background: lightblue;
}
.box2 {
  margin-top: 16px; /* margem superior */
  background: lightgreen;
}
```

Colapso de margens

- Ocorre entre irmãos verticais e entre elemento e seu pai (em certos casos) quando não há padding/overflow que “quebre”.
- Para evitar colapso: adicione **padding-top: 20px;** ou **overflow: auto;** no contêiner.

HTML + CSS / SASS (continuação)

Colapso de margens

```html
<section class="container">
    <h1>Título</h1>
</section>
```

```css
.container {
    background: #D66;
    /* Tente adicionar isso depois */
    /* padding-top: 20px; */
    /* ou */
    /* Tente adicionar isso depois */
    /* overflow: auto; */
}

.container h1 {
    margin-top: 50px;
}
/* A margem do h1 pode "subir" e empurrar o container, parecendo margem do container. */
```

HTML + CSS / SASS (continuação)

box-sizing: qual a diferença?

O **box-sizing** define como o navegador calcula a largura e a altura de um elemento.

1. **content-box (padrão)**
   a. O **width** e o **height** controlam só o conteúdo.
   b. O padding e a border são adicionados em cima disso.

```html
<div class="caixa"></div>
```

```css
.caixa {
    width: 200px;              /* só o conteúdo */
    padding: 20px;             /* aumenta 40px (20 esquerda + 20 direita) */
    border: 5px solid black; /* aumenta +10px */
}
```

Tamanho total real: 200 + 40 + 10 = **250px**

box-sizing: qual a diferença?

2. **border-box**
   a. O **width** e o **height** já incluem conteúdo + padding + border.
   b. Fica mais previsível e evita “quebras” no layout.

```html
<div class="caixa"></div>
```

```css
.caixa {
    box-sizing: border-box;
    width: 200px; /* inclui conteúdo, padding e border */
    padding: 20px;
    border: 5px solid black;
}
```

Tamanho total real: 200px fixo (o conteúdo diminui para caber junto com o padding e border).

HTML + CSS / SASS (continuação)

Comparação rápida

- **content-box**: tamanho cresce com padding e border**.**
- **border-box**: tamanho fica fixo, mais fácil de controlar.

Boa prática:

```
style.css > ...
1   * {
2       box-sizing: border-box;
3   }
4
```

Usado na maioria dos projetos modernos para simplificar cálculos.

HTML + CSS / SASS (continuação)

Overflow (quando o conteúdo não cabe)

- **overflow** controla o que acontece se o conteúdo for maior que a caixa:
  - **visible** (padrão): o conteúdo “vaza” para fora.
  - **hidden**: corta o conteúdo que ultrapassa.
  - **scroll**: adiciona barras de rolagem sempre.
  - **auto**: adiciona barra de rolagem só quando necessário.

```html
<div class="texto">aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</div>
```

style.css > ...

```css
.texto {
    width: 150px;
    height: 50px;
    overflow:auto;
    background-color: blue;
}
```

Outline x Border

- Border: faz parte da caixa, aumenta o tamanho.
- Outline: é como um marcador em volta, não altera o tamanho.

```html
<div class="box">TEXTO</div>
```

style.css > ...

```css
.box {
    border: 3px solid black;
    outline: 3px solid green;
}
```

O **outline** é ótimo para mostrar quando um campo está em foco (formulários).

HTML + CSS / SASS (continuação)

Background-clip

- Define até onde o fundo (background) será pintado:
  - **border-box**: fundo vai até a borda.
  - **padding-box**: fundo vai só até o padding.
  - **content-box**: fundo só no conteúdo.

```html
<div class="alerta">TEXTO</div>
```

style.css > ...

```css
.alerta {
    border: 10px dashed red;
    background: yellow;
    padding: 10px;
    background-clip: border-box;  /* tente também padding-box e content-box */
}
```

Lógica de Programação com JavaScript (continuação)

Exercícios para praticar (gabarito)

JS script.js > ...

```
1   // 1 - Crie uma função chamada dobrar que receba um número
    como parâmetro e mostre o dobro desse número usando console.
    log.
2   function dobrar(n) {
3       console.log(n * 2);
4   }
5   dobrar(5);
6
```

Lógica de Programação com JavaScript (continuação)

Exercícios para praticar (gabarito)

JS script.js > ...

```
1   // 3 - Crie uma função chamada saudacao que receba um nome
    como parâmetro e mostre no console a frase: "Olá, [nome]!".
    Se nenhum nome for passado, deve mostrar "Olá, Visitante!".
2   function saudacao(nome = "Visitante") {
3       console.log("Olá, " + nome + "!");
4   }
5   saudacao();
6   saudacao("Ana");
7
```

Lógica de Programação com JavaScript (continuação)

Exercícios para praticar (gabarito)

JS script.js > ...

```
1   // 4 - Crie uma função chamada ehPar que receba um número e
    mostre true se ele for par e false se for ímpar.
2   function ehPar(numero) {
3       console.log(numero % 2 === 0);
4   }
5   ehPar(6);
6   ehPar(7);
7
```

Lógica de Programação com JavaScript (continuação)

Exercícios para praticar (gabarito)

JS script.js > ...

```
1   // 5 - Crie uma função chamada maiorNumero que receba dois
    números e mostre no console qual deles é o maior.
2   function maiorNumero(a, b) {
3       if (a > b) {
4           console.log(a);
5       } else {
6           console.log(b);
7       }
8   }
9   maiorNumero(10, 7);
10
```

Lógica de Programação com JavaScript (continuação)

Exercícios para praticar (gabarito)

JS script.js > ...

```
1   // 6 - Crie uma função chamada contarCaracteres que receba
    uma palavra e mostre no console quantos caracteres ela tem.
    (use variavel.length para descobrir o tamanho).
2   function contarCaracteres(palavra) {
3       console.log(palavra.length);
4   }
5   contarCaracteres("React");
6
```

ATÉ A PRÓXIMA AULA!

Front-end - Design. Integração. Experiência.

Professor: Hygor Rasec
https://www.linkedin.com/in/hygorrasec
https://github.com/hygorrasec