Professores:
Celso Giusti
Daniel Manoel Filho
Marlon Palata Fanger Rodrigues

Diagramação e Posicionamento (Layout CSS)

SENAI

O Que é Diagramação (Layout)?

Até agora, aprendemos a criar elementos (HTML Semântico) e a estilizá-los individualmente (CSS Box Model, Cores, Fontes).

Diagramação (ou Layout) é a arte de **posicionar** esses elementos na página. É como decidimos onde o cabeçalho fica, se haverá uma barra lateral, ou como dois blocos de texto ficarão lado a lado.

Tudo o que aprendemos até agora (Box Model, seletores, semântica) será usado para construir nossos layouts.

Setup para a Aula Prática

Vamos usar um arquivo HTML simples para testar todos os conceitos de posicionamento.

Estrutura:

```
/aula16-layout/
|-- index.html
|-- style.css
```

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Aula 16 - Layout CSS</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <div class="caixa" id="caixa1">1 (static)</div>
    <div class="caixa" id="caixa2">2 (relative)</div>
    <div class="caixa" id="caixa3">3 (absolute)</div>
    <div class="caixa" id="caixa4">4 (fixed)</div>
  </div>
  <div class="footer-clear">Footer</div>
</body>
</html>
```

Revisão Vital: O Box Model

Não podemos posicionar nada sem dominar o Box Model. Lembre-se que todo elemento é uma caixa.

- **width / height**: Define o tamanho do **conteúdo**.
- **padding**: Espaçamento **interno** (entre o conteúdo e a borda).
- **border**: A borda.
- **margin**: Espaçamento **externo** (entre a borda e outros elementos).

CSS (style.css - Estilo Base):

```css
/* Estilo base para todas as caixas */
.caixa {
    width: 200px;
    height: 100px;
    padding: 10px;
    border: 3px solid #333;
    margin: 10px;
}

/* Dando cores para identificar */
#caixa1 { background-color: #e67e22; }
#caixa2 { background-color: #2ecc71; }
#caixa3 { background-color: #3498db; }
#caixa4 { background-color: #9b59b6; }
```

A Propriedade Mais Importante do Box Model

O cálculo de **width** padrão é confuso: **largura_total = width + padding + border.**

Para facilitar o layout, **sempre** usaremos **box-sizing: border-box;**. Isso força o **padding** e a **border** a ficarem *para dentro* da largura (**width**) que definimos.

CSS (style.css - Adicione no Topo):

```css
/* Reset universal do Box Model */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
body {
    font-family: sans-serif;
    padding-top: 20px; /* Espaço para ver os exemplos */
}
/* Container para os exemplos */
.container {
    border: 2px dashed #ccc;
    padding: 10px;
    margin-bottom: 20px;
}
```

HTML E CSS - Diagramação com CSS

display: O Controle do Fluxo

A propriedade **display** é a mais fundamental do layout. Ela define como um elemento se comporta no fluxo da página.

- **display: block;** (Padrão de ** , <p>, <section>,** etc.)
  - Ocupa 100% da largura disponível.
  - Começa sempre em uma nova linha.
  - Respeita **width, height, margin** e **padding.**
  - É o "tijolo" do layout.
- **display: inline;** (Padrão de **<a>, , ,** etc.)
  - Ocupa apenas a largura do seu conteúdo.
  - Flui junto com o texto, não quebra linha.
  - **Ignora width, height, margin-top e margin-bottom**.
  - É a "palavra" no meio de uma frase.

display: inline-block; e display: none;

display: inline-block;

- O "melhor dos dois mundos".
- Flui como uma "palavra" (**inline**), não quebra a linha.
- **Mas** permite definir **width, height, margin** e **padding**, como um "tijolo" (**block**).
- Muito usado para criar itens de menu ou botões lado a lado.

display: none;

- Remove o elemento completamente da página. Ele não é apenas escondido; ele "deixa de existir" para o layout, e os outros elementos ocupam seu espaço.
- (Diferente de **visibility**: **hidden**;, que esconde mas preserva o espaço.)

O "Fluxo Normal" vs. Posicionamento

Por padrão, os elementos (**block** e **inline**) são organizados no **Fluxo Normal**. Um **div** (**block**) empilha sobre o outro.

A propriedade **position** nos permite quebrar esse fluxo normal e posicionar elementos com total precisão.

Valores da Propriedade position:

- **static** (O padrão)
- **relative** (O ponto de partida)
- **absolute** (A ferramenta de precisão)
- **fixed** (O "adesivo" da tela)

position: static e position: relative

position: static;

- É o valor padrão. O elemento segue o **Fluxo Normal**. As propriedades **top**, **right**, **bottom**, **left** e **z-index não funcionam**.

position: relative;

- O elemento **ainda ocupa seu espaço** no Fluxo Normal.
- Agora, as propriedades **top**, **right**, **bottom**, **left** funcionam. Elas movem o elemento **relativo à sua própria posição original**.
- **Uso Principal: position: relative** é usado para transformar um elemento em um **container de posicionamento** para seus filhos **absolute**.

CSS **(style.css):**

```
#caixa2 {
    position: relative;
    top: 30px; /* Desce 30px */
    left: 50px; /* Vai 50px para a direita */

    /* O espaço original da Caixa 2
(verde) ainda está lá,
       mas a caixa 3 (azul) vai se
sobrepor a ela! */
}
```

position: absolute

Este é o mais poderoso e o mais confuso.

- O elemento é **completamente removido** do Fluxo Normal.
- Os outros elementos agem como se ele não existisse (eles ocupam o espaço dele).
- Ele é posicionado usando **top, right, bottom, left** em relação ao seu **ancestral posicionado mais próximo**.
- "Ancestral posicionado" = um elemento pai com **position** diferente de **static** (ou seja, **relative**, **absolute** ou **fixed**).
- Se não houver ancestral posicionado, ele se posiciona em relação à tag **<body>**.

CSS **(style.css):**

```
#caixa3 {
    position: absolute;
    top: 5px;  /* 5px do topo do .container
*/
    right: 5px; /* 5px da direita do
.container */
    width: 100px; /* Reduzimos para
caber */
}
```

position: fixed

- O elemento é **completamente removido** do Fluxo Normal.
- Ele é posicionado usando **top, right, bottom, left** em relação à **Viewport** (a janela do navegador).
- Ele **não rola** com a página. Fica "fixo" na tela.

CSS **(style.css):**

```
#caixa4 {
    position: fixed;
    bottom: 10px; /* 10px do final da janela */
    right: 10px;  /* 10px da direita da janela */
    width: 250px;
    /* Esta caixa vai ficar no canto da tela mesmo se você rolar a página */
}
```

z-index: Empilhando Elementos

Quando elementos saem do Fluxo Normal (com **position**), eles podem se sobrepor. O **z-index** controla a ordem de empilhamento (qual fica na frente).

- **Só funciona em elementos posicionados** (**relative, absolute, fixed**).
- É um valor numérico (sem unidade).
- O valor padrão é **auto** (ou **0**).
- Um número **maior** fica **na frente** de um número menor.

CSS **(style.css):**

```css
#caixa2 { /* Relative */
    /* ... */
    z-index: 10; /* Fica na frente da caixa 3 */
}

#caixa3 { /* Absolute */
    /* ... */
    z-index: 5; /* Fica atrás da caixa 2 */
}
```

O Que Fazer com float? (Uso Moderno)

Antigamente, **float** era usado para criar layouts de colunas (como **main** e **aside** lado a lado). **NÃO FAZEMOS MAIS ISSO.** É um método antigo, confuso e que "quebra" o layout, exigindo **clear: both**; para consertar.

Hoje, usamos **float** apenas para seu propósito original e semântico: fazer o texto **fluir ao redor** de um elemento (geralmente uma imagem).

Exemplo de Uso Correto:

html:

```
<img src="ferrari.jpg"
class="avatar-flutuante" />
  <p>
    Este texto irá fluir ao redor da imagem...
  </p>
```

css:

```
.avatar-flutuante {
  float: left;
  width: 300px;
  margin-right: 15px; /* Espaço entre a
imagem e o texto */
}
```

Qual Método de Posicionamento Usar?

- **Fluxo Normal (display):** Para 90% da sua página (títulos, parágrafos, seções empilhadas).
- **position: relative/absolute:** Para elementos de UI que se sobrepõem (menus dropdown, ícones sobre imagens, pop-ups).
- **position: fixed:** Para elementos que devem ficar fixos na tela (menus de topo, banners de cookie, icones de chatbot).
- **float: APENAS** para fazer texto fluir ao redor de imagens.
- **Layout de Colunas (Lado a Lado):** Para isso, usaremos **Flexbox** e **Grid** (próxima aula), que são as ferramentas modernas e corretas.

HTML E CSS

Resumo:

SENAI

✅ **Layout** é sobre organizar as caixas do Box Model.

✅ **box-sizing: border-box** é seu melhor amigo. Use sempre.

✅ **display** controla o comportamento de fluxo da caixa (**block, inline, inline-block**).

✅ **position: relative** cria um "ponto de âncora" para seus filhos.

✅ **position: absolute** remove o elemento do fluxo e o posiciona em relação ao seu "ancoradouro" (**relative**).

✅ **position: fixed** remove o elemento do fluxo e o posiciona em relação à janela do navegador.

✅ **z-index** controla o empilhamento (camadas) de elementos posicionados.

✅ **float** NÃO é para layout. É para texto ao redor de imagens.

ATIVIDADE

O "Santo Graal" do Layout

1

Objetivo: Criar o layout mais clássico da web: um site com cabeçalho, menu de navegação, barra lateral, conteúdo principal e rodapé.

Instruções:

1. Crie um novo projeto **aula16-layout/exercicio** com **index.html** e **style.css**.
2. Copie o HTML base do próximo slide.
3. Sua missão: Usar o que aprendemos (especialmente **position**) para fazer o **aside** e **main** ficarem lado a lado.
4. O **footer** deve ficar sempre por último, abaixo de tudo.

PÁGINA INDEX.HTML:

https://github.com/marlon-palata/meu_site/blob/main/aula16-layout/exercicios/index.html

Referências

MOZILLA. **Pseudo-classes**. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-classes. Acesso em: 11 out. 2025.

MOZILLA. **Pseudo-elements**. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-elements. Acesso em: 11 out. 2025.

MOZILLA. **Using CSS transitions**. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions. Acesso em: 11 out. 2025.

MOZILLA. **transform**. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS/transform. Acesso em: 11 out. 2025.

MOZILLA. **Using CSS animations**. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Animations/Using_CSS_animations. Acesso em: 11 out. 2025.

W3SCHOOLS. **CSS Gradients**. Disponível em: https://www.w3schools.com/css/css3_gradients.asp. Acesso em: 11 out. 2025.