Notas fundação Bradesco:

áreas de atuação

front-end- angular, react

back-end: java, c#

banco de dados: sql

Mobile: swift, kotlin/ flutter(dart)

jogos:

segurança

devops:

IA

fullstack:

Alert() - para passar a mensagem

Mantenha o padrão : ‘’ ou “”

variavel em = let

nomes de funções

<script src="app. " defer></script> If the defer attribute is set, it specifies that the script is downloaded in parallel to parsing the page, and executed after the page has finished parsing.

quando o atributo defer está definido, significa que o atributo está fazendo o download em paralelo da página e executado quando a página terminar de carregar

quando o atributo async está definido, significa que o download é em paralelo,e é executado antes que a página carregue totalmente

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Título da Página</title>

</head>

<body>

<h1>Bem-vindo ao Mundo do HTML!</h1>

<p>Este é um parágrafo de exemplo.</p>

</body>

</html>

### **O que é HTML?**

HTML (Hypertext Markup Language) é uma linguagem de marcação usada para estruturar e organizar o conteúdo de uma página web. Ao contrário das linguagens de programação, HTML não é usado para lógica de controle ou cálculos, mas para criar uma estrutura semântica, com a qual o conteúdo da página pode ser compreendido tanto pelos navegadores quanto pelos motores de busca e outras tecnologias assistivas.

* **Marcação**: HTML usa *tags* (ou elementos) para marcar diferentes partes do conteúdo. Essas tags não são vistas diretamente pelo usuário, mas são interpretadas pelo navegador para renderizar a página de forma adequada.  
    
   **Exemplo**:  
  + <h1>Olá, Mundo!</h1>: Define um título de nível 1.
  + <p>Este é um parágrafo</p>: Define um parágrafo de texto.

### **2. Estrutura Básica de um Documento HTML**

A estrutura básica de um documento HTML é um esqueleto que é sempre utilizado, e pode ser expandido conforme necessário. Vamos explicar detalhadamente cada parte:

#### **<!DOCTYPE html>**

Esta linha deve sempre aparecer no início de um arquivo HTML. Ela informa ao navegador que o arquivo está utilizando o HTML5, que é a versão mais moderna da linguagem. A ausência do DOCTYPE pode causar comportamentos inesperados no navegador.

#### **<html lang="pt-br">**

A tag <html> envolve todo o conteúdo da página HTML. Atributos podem ser usados nela para configurar comportamentos específicos:

* lang="pt-br": Define o idioma da página como português do Brasil. Isso é importante para acessibilidade, pois motores de busca e leitores de tela podem usar essa informação para melhor interpretação do conteúdo.

#### **<head>**

A tag <head> contém metadados sobre a página, como informações de codificação de caracteres, links para arquivos externos (como CSS e JavaScript), título da página e outros dados.

* **<meta charset="UTF-8">**: Define o tipo de codificação de caracteres. UTF-8 é o mais comum, pois cobre a maioria dos caracteres internacionais.
* **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: Essa tag é fundamental para garantir que a página seja responsiva e que se ajuste corretamente a diferentes tamanhos de tela, especialmente em dispositivos móveis.
* **<title>**: Define o título da página, que aparece na aba do navegador. O título deve ser conciso e descritivo, pois é um dos elementos mais importantes para SEO (Search Engine Optimization).

#### **<body>**

A tag <body> contém todo o conteúdo visível da página, como texto, imagens, links, formulários, etc. É a parte da página que será renderizada pelo navegador.

### **3. Elementos HTML Básicos**

Agora, vamos mergulhar nos elementos mais importantes do HTML. Cada tipo de conteúdo no seu site será representado por um tipo específico de elemento.

#### **3.1. Títulos (<h1>, <h2>, etc.)**

O HTML possui seis níveis de cabeçalhos, de <h1> a <h6>. Eles são usados para organizar o conteúdo hierarquicamente, com o <h1> sendo o título principal (o mais importante) e o <h6> o título de nível mais baixo. O uso correto desses títulos melhora a acessibilidade e a indexação do conteúdo por motores de busca.

html

Copiar

<h1>Título Principal</h1>

<h2>Subtítulo</h2>

<h3>Subtítulo Menor</h3>

**Importância**:

* O <h1> deve ser utilizado apenas uma vez por página, para o título principal, pois ele ajuda o Google a entender do que se trata a página.
* Use <h2> a <h6> para organizar subtítulos de forma hierárquica.

#### **3.2. Parágrafos (<p>)**

O parágrafo é representado pela tag <p>. Essa tag é usada para agrupar um bloco de texto.

html

Copiar

<p>Este é um parágrafo de texto. Ele pode conter informações sobre o assunto principal da página.</p>

**Importância**:

* Sempre use <p> para dividir blocos de texto. Isso torna a leitura do conteúdo mais fácil e melhora a acessibilidade.
* Evite usar outras tags como <div> para criar parágrafos, pois elas são destinadas a agrupar blocos de conteúdo e não têm um significado semântico.

#### **3.3. Links (<a>)**

A tag <a> é usada para criar links. Ela deve sempre ter o atributo href, que define o destino do link.

html

Copiar

<a href="https://www.exemplo.com">Visite o Exemplo</a>

**Atributos comuns**:

* **href**: Define o destino do link.
* **target="\_blank"**: Faz com que o link abra em uma nova aba ou janela.

html

Copiar

<a href="https://www.exemplo.com" target="\_blank">Abrir em nova aba</a>

#### **3.4. Imagens (<img>)**

A tag <img> é usada para inserir imagens. Ela é uma tag auto-fechante, ou seja, não precisa de uma tag de fechamento.

html

Copiar

<img src="imagem.jpg" alt="Descrição da imagem">

* **src**: Define o caminho da imagem.
* **alt**: Fornece um texto alternativo que será exibido caso a imagem não possa ser carregada. Também é importante para acessibilidade, já que leitores de tela utilizam esse texto para descrever a imagem.

#### **3.5. Listas (<ul>, <ol>, <li>)**

Existem dois tipos principais de listas:

* **Listas não ordenadas** (<ul>) são usadas quando a ordem dos itens não é importante.
* **Listas ordenadas** (<ol>) são usadas quando a ordem dos itens é relevante.

html

Copiar

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

<ol>

<li>Primeiro item</li>

<li>Segundo item</li>

</ol>

#### **3.6. Tabelas (<table>, <tr>, <th>, <td>)**

As tabelas são usadas para exibir dados em formato tabular.

html

Copiar

<table>

<tr>

<th>Nome</th>

<th>Idade</th>

</tr>

<tr>

<td>João</td>

<td>30</td>

</tr>

</table>

* **<table>**: Define a tabela.
* **<tr>**: Define uma linha de dados na tabela.
* **<th>**: Define uma célula de título, que é sempre em negrito.
* **<td>**: Define uma célula de dados.

#### **3.7. Formulários (<form>, <input>, <textarea>, <button>)**

Formulários são usados para coletar dados do usuário. Eles podem incluir diferentes tipos de entradas, como campos de texto, botões e áreas de texto.

html

Copiar

<form action="/enviar" method="post">

<label for="nome">Nome:</label>

<input type="text" id="nome" name="nome">

<label for="email">Email:</label>

<input type="email" id="email" name="email">

<button type="submit">Enviar</button>

</form>

* **<label>**: Associado a um campo de entrada, melhora a acessibilidade.
* **<input>**: Define um campo de entrada. O tipo (type) pode ser variado, como text, password, email, submit, etc.
* **<textarea>**: Define uma área de texto multilinha.
* **<button>**: Define um botão, geralmente usado para submeter formulários.

### **4. Atributos HTML**

Atributos HTML são usados para fornecer informações adicionais sobre um elemento. Alguns dos mais comuns incluem:

* **id**: Define um identificador único para um elemento. Usado para estilização e manipulação via JavaScript.
* **class**: Define uma ou mais classes para um elemento. Permite aplicar estilos a múltiplos elementos.
* **href**: Usado em links para definir o destino.
* **src**: Usado em imagens e scripts para definir o caminho para a imagem ou arquivo de script.
* **alt**: Fornece uma descrição alternativa de uma imagem.

### **5. Elementos de Bloco vs. Elementos Inline**

* **Elementos de bloco**: Ocupam a largura total disponível e começam em uma nova linha, como <div>, <p>, <h1>.
* **Elementos inline**: Não quebram a linha e ocupam apenas o espaço necessário, como <span>, <a>, <img>.

### **6. Estilizando com CSS**

CSS (Cascading Style Sheets) é usado para estilizar os elementos HTML. Existem três maneiras de aplicar CSS a um documento HTML:

1. **Interno**: Usando a tag <style> dentro do <head>.
2. **Externo**: Usando a tag <link> para importar um arquivo .css.
3. **Inline**: Usando o atributo style diretamente nos elementos.

html

Copiar

<p style="color: red;">Este texto é vermelho.</p>

### **7. Práticas Recomendadas**

* **Usar tags semânticas** como <article>, <section>, <header>, <footer> para organizar o conteúdo de forma lógica e acessível.
* **Evitar usar <div> para tudo**: Utilize as tags apropriadas para manter a semântica da página.

8. Tag iframe

<iframe src =”” width= “” height=”” title = “meu iframe”></iframe>

### **Exercício 1: Estrutura Básica de um Documento HTML**

**Objetivo**: Criar a estrutura básica de um arquivo HTML.

**Instruções**: Peça para os alunos criarem a estrutura básica de um documento HTML, com as tags obrigatórias, como <!DOCTYPE html>, <html>, <head>, <meta>, <body>, <h1>, <p>.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Meu Primeiro HTML</title>

</head>

<body>

<h1>Bem-vindo à Aula de HTML!</h1>

<p>Este é um parágrafo explicando o básico de HTML.</p>

</body>

</html>

### **Exercício 2: Adicionando Títulos e Parágrafos**

**Objetivo**: Usar as tags de título e parágrafo para estruturar um conteúdo simples.

**Instruções**: Peça para os alunos criarem um arquivo HTML com diferentes níveis de títulos e pelo menos dois parágrafos explicando o conteúdo que eles escolherem.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Títulos e Parágrafos</title>

</head>

<body>

<h1>Introdução ao HTML</h1>

<h2>O que é HTML?</h2>

<p>HTML é uma linguagem de marcação usada para criar páginas web.</p>

<h2>Estrutura de um documento HTML</h2>

<p>Todo documento HTML começa com uma declaração de tipo e inclui tags como <html>, <head> e <body>.</p>

</body>

</html>

### **Exercício 3: Links e Imagens**

**Objetivo**: Incluir links e imagens em uma página.

**Instruções**: Peça para os alunos criarem um arquivo HTML que inclua:

1. Um link para um site externo.
2. Uma imagem com um texto alternativo.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Links e Imagens</title>

</head>

<body>

<h1>Links e Imagens em HTML</h1>

<p>Visite o <a href="https://www.google.com" target="\_blank">Google</a>.</p>

<img src="imagem.jpg" alt="Imagem de exemplo">

</body>

</html>

### **Exercício 4: Listas Ordenadas e Não Ordenadas**

**Objetivo**: Criar listas ordenadas e não ordenadas.

**Instruções**: Peça para os alunos criarem uma página HTML com uma lista não ordenada de itens que eles gostam de fazer, seguida por uma lista ordenada das etapas para aprender HTML.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Listas em HTML</title>

</head>

<body>

<h1>Minhas Atividades Favoritas</h1>

<ul>

<li>Ler livros</li>

<li>Correr no parque</li>

<li>Assistir filmes</li>

</ul>

<h2>Como Aprender HTML</h2>

<ol>

<li>Estudar a estrutura básica</li>

<li>Praticar criando páginas simples</li>

<li>Adicionar imagens, links e listas</li>

</ol>

</body>

</html>

### **Exercício 5: Criando um Formulário**

**Objetivo**: Criar um formulário simples de contato.

**Instruções**: Peça para os alunos criarem um formulário HTML com os seguintes campos:

* Nome (campo de texto)
* E-mail (campo de texto)
* Mensagem (campo de área de texto)
* Um botão para enviar

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Formulário de Contato</title>

</head>

<body>

<h1>Formulário de Contato</h1>

<form action="/enviar" method="post">

<label for="nome">Nome:</label>

<input type="text" id="nome" name="nome" required><br><br>

<label for="email">E-mail:</label>

<input type="email" id="email" name="email" required><br><br>

<label for="mensagem">Mensagem:</label><br>

<textarea id="mensagem" name="mensagem" rows="4" cols="50" required></textarea><br><br>

<button type="submit">Enviar</button>

</form>

</body>

</html>

### **Exercício 6: Tabela com Dados**

**Objetivo**: Criar uma tabela HTML com dados de uma lista de alunos.

**Instruções**: Peça para os alunos criarem uma tabela HTML com as seguintes colunas:

* Nome
* Idade
* Curso

E preenchê-la com 3 ou 4 alunos fictícios.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Tabela de Alunos</title>

</head>

<body>

<h1>Lista de Alunos</h1>

<table border="1">

<tr>

<th>Nome</th>

<th>Idade</th>

<th>Curso</th>

</tr>

<tr>

<td>João Silva</td>

<td>20</td>

<td>Engenharia</td>

</tr>

<tr>

<td>Maria Oliveira</td>

<td>22</td>

<td>Design Gráfico</td>

</tr>

<tr>

<td>Pedro Santos</td>

<td>19</td>

<td>Ciência da Computação</td>

</tr>

</table>

</body>

</html>

### **Exercício 7: Criando uma Página de Portfólio**

**Objetivo**: Criar uma página de portfólio simples, com imagem e descrição de projetos.

**Instruções**: Peça para os alunos criarem uma página HTML de portfólio que contenha:

1. Uma imagem de perfil.
2. Uma lista de projetos.
3. Descrição dos projetos.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Meu Portfólio</title>

</head>

<body>

<h1>Olá, sou João!</h1>

<img src="foto.jpg" alt="Foto de João" width="200">

<h2>Meus Projetos</h2>

<ul>

<li><strong>Projeto 1:</strong> Site Pessoal - Um site para compartilhar minhas experiências.</li>

<li><strong>Projeto 2:</strong> Blog de Tecnologia - Um blog sobre as últimas tendências em tecnologia.</li>

<li><strong>Projeto 3:</strong> E-commerce - Uma plataforma de vendas online.</li>

</ul>

</body>

</html>

### **Exercício 8: Introdução ao CSS (Estilizando Títulos e Parágrafos)**

**Objetivo**: Adicionar estilo básico a títulos e parágrafos.

**Instruções**: Peça para os alunos estilizar os títulos e parágrafos utilizando CSS. Eles devem mudar a cor do título e o tamanho da fonte dos parágrafos.

**Exemplo de Solução**:

html

Copiar

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CSS no HTML</title>

<style>

h1 {

color: blue;

}

p {

font-size: 18px;

}

</style>

</head>

<body>

<h1>Olá, Mundo!</h1>

<p>Este é um exemplo de como estilizar com CSS.</p>

</body>

</html>

### **Estrutura Básica**

* **<!DOCTYPE html>**: Declara que o documento é um arquivo HTML5.
* **<html>**: Tag raiz do documento HTML. Engloba todo o conteúdo da página.
* **<head>**: Contém informações não visíveis para o usuário, como metadados e links para arquivos externos (CSS, ).
* **<meta>**: Usado para definir metadados, como a codificação de caracteres e configurações de responsividade.  
  + **<meta charset="UTF-8">**: Define a codificação de caracteres para o documento.
  + **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: Torna a página responsiva, adaptando-se ao tamanho da tela do dispositivo.
* **<title>**: Define o título da página que aparece na aba do navegador.
* **<body>**: Contém o conteúdo visível da página, como texto, imagens, links, formulários, etc.

### **2. Títulos e Parágrafos**

* **<h1>, <h2>, <h3>, <h4>, <h5>, <h6>**: Definem títulos de diferentes níveis. O <h1> é o mais importante e o <h6> o de menor importância.
* **<p>**: Define um parágrafo de texto.

### **3. Links e Imagens**

* **<a>**: Define um link (âncora). O atributo href define o destino do link.  
  + **<a href="url">Texto do Link</a>**: Exemplo de link básico.
* **<img>**: Define uma imagem na página. O atributo src especifica a URL da imagem e alt fornece um texto alternativo para a imagem.  
  + **<img src="imagem.jpg" alt="Descrição da imagem">**: Exemplo de tag de imagem.

### **4. Listas**

* **<ul>**: Define uma lista não ordenada (com marcadores).
* **<ol>**: Define uma lista ordenada (numerada).
* **<li>**: Define um item de lista, tanto para listas ordenadas quanto não ordenadas.

### **5. Tabelas**

* **<table>**: Define uma tabela.
* **<tr>**: Define uma linha dentro de uma tabela.
* **<th>**: Define uma célula de cabeçalho de tabela (geralmente em negrito).
* **<td>**: Define uma célula de dados em uma tabela.

### **6. Formulários**

* **<form>**: Define um formulário. O atributo action define para onde os dados do formulário serão enviados, e o atributo method define o método HTTP (geralmente GET ou POST).
* **<input>**: Define um campo de entrada de dados. O atributo type pode ser usado para definir o tipo de campo, como text, email, password, etc.  
  + **<input type="text">**: Campo de texto.
  + **<input type="email">**: Campo de e-mail.
  + **<input type="submit">**: Botão de envio de formulário.
* **<textarea>**: Define uma área de texto multilinha, ideal para mensagens ou descrições mais longas.
* **<button>**: Define um botão dentro de um formulário.

### **Resumo das Tags Usadas nos Exercícios**

1. **Estrutura de Página**: <!DOCTYPE html>, <html>, <head>, <meta>, <title>, <body>
2. **Texto e Estrutura**: <h1>, <h2>, <p>, <span>, <div>, <br>, <hr>,<strong>
3. **Links e Navegação**: <a>, href
4. **Listas**: <ul>, <ol>, <li>
5. **Tabelas**: <table>, <tr>, <th>, <td>
6. **Formulários**: <form>, <input>, <textarea>, <button>

# **AULA COMPLETA SOBRE CSS**

## **📌 O que é CSS?**

**CSS (Cascading Style Sheets)** é uma linguagem de estilo utilizada para definir a aparência visual das páginas HTML. Enquanto o HTML define a estrutura, o CSS é responsável por formatar os elementos, como cores, fontes, espaçamentos e layouts.

**Vantagens do uso do CSS:**

* Separa o conteúdo visual do estrutural;
* Facilita a manutenção;
* Permite personalização e responsividade;
* Otimiza o desenvolvimento web.

## **📌 Como utilizar CSS?**

Existem três formas principais de incluir CSS numa página HTML:

### **✅ 1. Inline (diretamente no elemento HTML)**

Útil para alterações pontuais, mas menos recomendado devido à manutenção mais difícil.

**Sintaxe:**

html

<p style="color: red; font-size: 20px;">Texto estilizado inline</p>

### **✅ 2. Interno (CSS incorporado no HTML)**

Ideal para páginas únicas ou rápidas estilizações.

**Sintaxe:**

html

<head>

<style>

p {

color: blue;

font-size: 18px;

}

</style>

</head>

<body>

<p>Texto estilizado internamente</p>

</body>

### **✅ 3. Externo (arquivo CSS separado)**

Recomendado, facilita manutenção e organização.

**Exemplo**:

**index.html**

html

<head>

<link rel="stylesheet" href="estilos.css">

</head>

<body>

<p>Texto estilizado externamente</p>

</body>

**estilos.css**

css

p {

color: green;

font-size: 22px;

}

## **📌 Seletores no CSS**

Os seletores definem quais elementos serão estilizados:

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Exemplo** | **Descrição** |
| **Elemento** | p {} | Seleciona todos elementos <p> |
| **Classe** | .classe {} | Seleciona elementos com classe específica |
| **ID** | #id {} | Seleciona um único elemento com ID específico |
| **Universal** | \* {} | Seleciona todos os elementos da página |
| **Combinação** | div p {} | Seleciona <p> dentro de <div> |

## **📌 Principais Propriedades CSS**

**Cores:** css  
   
color: red;

background-color: #333;

**Fontes:** css  
   
font-family: Arial, sans-serif;

font-size: 16px;

font-weight: bold;

**Espaçamentos e Layout:** css  
   
margin: 10px;

padding: 15px;

width: 100px;

height: 200px;

**Bordas:** css  
   
border: 1px solid black;

border-radius: 10px;

**Alinhamento:** css  
   
text-align: center;

## **📌 Exemplos Completos**

### **Exemplo de estilização inline:**

html

<p style="color: purple; font-weight: bold;">

Este parágrafo está estilizado inline.

</p>

### **Exemplo de estilização interna:**

html

<!DOCTYPE html>

<html>

<head>

<style>

h1 { color: blue; }

.destaque { background-color: yellow; }

</style>

</head>

<body>

<h1>Estilo Interno</h1>

<p class="destaque">Parágrafo destacado.</p>

</body>

</html>

### **Exemplo de estilização externa:**

**HTML**

html

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h2 id="titulo">Estilo Externo</h2>

<p class="texto">Parágrafo estilizado.</p>

</body>

</html>

**CSS (style.css)**:

css

#titulo {

color: darkgreen;

text-align: center;

}

.texto {

color: black;

font-size: 18px;

}

## **📌 Atividades para a Sala de Aula**

Aqui estão atividades práticas para fixar os conceitos ensinados:

### **📝 Atividade 1 – Inline CSS**

* Crie uma página HTML com três parágrafos.
* Use estilos inline para deixar cada parágrafo de uma cor diferente.

**Correção (exemplo)**:

html

<p style="color:red;">Parágrafo vermelho</p>

<p style="color:blue;">Parágrafo azul</p>

<p style="color:green;">Parágrafo verde</p>

### **📝 Atividade 2 – CSS Interno**

* Crie uma página HTML contendo:  
  + Um título principal (<h1>).
  + Um subtítulo (<h2>).
  + Dois parágrafos.
* Utilize CSS interno para estilizar os títulos e parágrafos.

**Correção (exemplo)**:

html

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<style>

h1 { color: orange; }

h2 { color: gray; }

p { font-size: 18px; }

</style>

</head>

<body>

<h1>Título Principal</h1>

<h2>Subtítulo Secundário</h2>

<p>Primeiro parágrafo estilizado.</p>

<p>Segundo parágrafo estilizado.</p>

</body>

</html>

### **📝 Atividade 3 – CSS Externo**

* Crie uma página HTML com:  
  + Uma lista (<ul>) de 3 itens;
  + Um formulário simples com 2 campos (nome e email);
  + Um botão.
* Faça um arquivo CSS separado que estilize:  
  + A lista com cor de fundo e espaçamento;
  + Os inputs com bordas arredondadas;
  + O botão com cor de fundo e cor do texto personalizada.

**Correção (exemplo):**

**HTML:**

html

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<link rel="stylesheet" href="estilos.css">

</head>

<body>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

<form>

Nome:<br>

<input type="text" name="nome"><br>

Email:<br>

<input type="email" name="email"><br><br>

<button type="submit">Enviar</button>

</form>

</body>

</html>

**CSS (estilos.css)**:

css

ul {

background-color: #f2f2f2;

padding: 10px;

}

input {

border-radius: 5px;

padding: 5px;

margin-bottom: 10px;

}

button {

background-color: darkcyan;

color: white;

padding: 8px;

border: none;

border-radius: 5px;

}

## **📌 Resumo da Aula**

* Aprendemos o que é CSS e sua importância;
* Conhecemos três formas de utilização: Inline, Interna e Externa;
* Trabalhamos com os principais seletores e propriedades do CSS;
* Realizamos exercícios práticos para consolidar o aprendizado.

**🖥️ Sugestões extras:**

* Incentive alunos a experimentar propriedades adicionais.
* Utilize ferramentas online como o [CodePen](https://codepen.io/) para praticar rapidamente.

Se precisar de mais material ou exercícios adicionais, me avise! Estou aqui para ajudar.

##### **Você disse:**

aprofunda mais, faça com que essa aula vire um curso, me explica sobre as class e id, como utilizar e fazer exercícios

###### **O ChatGPT disse:**

Claro, Diego! Vamos transformar esta aula em um **Curso Completo de CSS**, incluindo uma explicação detalhada sobre **Classes e IDs**, abordagens práticas, exemplos claros e exercícios específicos.

# **🎯 CURSO COMPLETO DE CSS**

**Sumário:**

* Introdução aprofundada ao CSS
* Modos de utilização do CSS (Inline, Interno e Externo)
* Seletores avançados
* Classes e IDs (conceitos, diferenças e usos práticos)
* Exercícios detalhados sobre Classes e IDs

## **🚀 1. INTRODUÇÃO APROFUNDADA AO CSS**

CSS (*Cascading Style Sheets*) é uma linguagem usada para controlar a aparência visual dos sites e aplicações web. Com o CSS você pode controlar cores, fontes, layouts, animações e muito mais.

**Por que usar CSS?**

* Facilita a padronização visual.
* Melhora a experiência do usuário.
* Torna páginas responsivas e adaptáveis.

## **🚀 2. MODOS DE UTILIZAÇÃO DO CSS**

### **🔸 Inline CSS**

Aplicado diretamente no elemento HTML através do atributo style.

html

<p style="color: blue; font-size:16px;">

Texto com estilo inline

</p>

### **🔸 Interno (Embedded CSS)**

Aplicado dentro da tag <style> no <head> do HTML.

html

<head>

<style>

p { color: green; font-size: 18px; }

</style>

</head>

<body>

<p>Texto com estilo interno</p>

</body>

### **🔸 Externo (Separate CSS)**

Utiliza arquivos separados .css vinculados via <link>.

html

<head>

<link rel="stylesheet" href="estilo.css">

</head>

<body>

<p>Texto com estilo externo</p>

</body>

estilo.css:

css

p {

color: purple;

font-size: 20px;

}

## **🚀 3. SELETORES AVANÇADOS**

Os seletores são essenciais para escolher quais elementos HTML estilizar.

**Principais seletores:**

* **Elemento**: p {} seleciona todos os parágrafos.
* **Classe**: .nomeClasse {} seleciona elementos que possuem determinada classe.
* **ID**: #nomeId {} seleciona o único elemento com determinado ID.
* **Descendente**: div p {} seleciona todo <p> dentro de <div>.

## **🚀 4. CLASSES E IDs: APROFUNDAMENTO**

### **🔖 4.1. Classes**

Uma classe (class) permite aplicar o mesmo estilo em diversos elementos.

**Características**:

* Podem ser usadas múltiplas vezes.
* São indicadas com um ponto (.) no CSS.

**Exemplo prático:**

**HTML**

html

<h2 class="titulo-verde">Título 1</h2>

<h3 class="titulo-verde">Título 2</h3>

<p class="texto">Parágrafo 1</p>

<p class="texto">Parágrafo 2</p>

**CSS**

css

.titulo-verde {

color: green;

}

.texto {

font-size: 16px;

font-family: Arial;

}

**Vantagem:** Reutilização eficiente de código.

### **🔖 4.2. IDs**

Um ID (id) deve ser único em toda a página e usado para identificar um único elemento.

**Características**:

* Deve aparecer uma única vez no HTML.
* É indicado com cerquilha (#) no CSS.

**Exemplo prático:**

**HTML**

html

<h1 id="titulo-principal">Título Principal</h1>

<p id="primeiro-paragrafo">Introdução do artigo.</p>

**CSS**

css

#titulo-principal {

color: blue;

text-align: center;

}

#primeiro-paragrafo {

font-size: 20px;

}

**Vantagem:** Controle individualizado de elementos específicos.

### **🔖 4.3. Diferença Prática entre Classe e ID**

|  |  |  |
| --- | --- | --- |
| **Critério** | **Classe** | **ID** |
| Uso | Pode ser repetida | Único elemento |
| Seleção no CSS | .nomeDaClasse | #nomeDoId |
| Aplicação comum | Reutilização em grupo | Elemento único (logo, título principal) |

## **🚀 5. EXERCÍCIOS PRÁTICOS SOBRE CLASSES E IDs**

Realize esses exercícios com os alunos, peça para explicarem suas escolhas:

### **📌 Exercício 1 – Classes**

**Instruções:**

* Crie uma página HTML com cinco parágrafos.
* Utilize uma classe CSS para estilizar três desses parágrafos com a fonte Arial, cor azul e tamanho de 16px.

**Correção (exemplo)**:

**HTML:**

<p class="texto-azul">Parágrafo 1</p>

<p class="texto-azul">Parágrafo 2</p>

<p class="texto-azul">Parágrafo 3</p>

<p>Parágrafo 4 (sem estilo)</p>

<p>Parágrafo 5 (sem estilo)</p>

**CSS:**

css

.texto-azul {

font-family: Arial;

color: blue;

font-size: 16px;

}

### **📌 Exercício 2 – IDs**

**Instruções:**

* Crie uma página HTML contendo um título principal, dois subtítulos e três parágrafos.
* Use um ID para estilizar o título principal com a cor vermelha e centralizado.

**Correção (exemplo):**

**HTML:**

html

<h1 id="titulo-principal">Título Principal</h1>

<h2>Subtítulo 1</h2>

<h2>Subtítulo 2</h2>

<p>Parágrafo 1</p>

<p>Parágrafo 2</p>

<p>Parágrafo 3</p>

**CSS:**

css

#titulo-principal {

color: red;

text-align: center;

}

### **📌 Exercício 3 – Classes e IDs Juntos**

**Instruções:**

* Crie uma página HTML contendo:  
  + Um título com um ID chamado "topo".
  + Três botões com a classe "botao".
* Utilize CSS para estilizar o título com fonte maior e centralizado, e os botões com cor de fundo azul e letras brancas.

**Correção (exemplo):**

**HTML:**

html

<h1 id="topo">Bem-vindo!</h1>

<button class="botao">Botão 1</button>

<button class="botao">Botão 2</button>

<button class="botao">Botão 3</button>

**CSS:**

css

#topo {

font-size: 40px;

text-align: center;

}

.botao {

background-color: blue;

color: white;

padding: 10px;

border: none;

margin: 5px;

border-radius: 5px;

}

# **Lista das Principais Propriedades CSS com Exemplos**

### **🔹 1. Propriedades de Texto**

* **color** – Define a cor do texto.

css

p {

color: blue;

}

* **font-family** – Define o tipo da fonte.

css

h1 {

font-family: Arial, sans-serif;

}

* **font-size** – Define o tamanho da fonte.

css

p {

font-size: 18px;

}

* **font-weight** – Define o peso/negrito da fonte.

css

h2 {

font-weight: bold;

}

* **text-align** – Define o alinhamento horizontal do texto.

css

h1 {

text-align: center;

}

* **text-decoration** – Decoração (ex: sublinhado).

css

a {

text-decoration: none;

}

* **line-height** – Espaçamento entre linhas.

css

p {

line-height: 1.6;

}

### **🔹 2. Propriedades de Cor e Fundo**

* **background-color** – Cor de fundo.

css

body {

background-color: #f2f2f2;

}

* **background-image** – Imagem de fundo.

css

div {

background-image: url("fundo.jpg");

}

### **🔹 3. Propriedades de Espaçamento e Posicionamento**

* **margin** – Margem externa (espaço fora do elemento).

css

h1 {

margin: 20px;

}

* **padding** – Espaçamento interno (espaço dentro do elemento).

css

div {

padding: 15px;

}

* **width e height** – Largura e altura do elemento.

css

img {

width: 200px;

height: 100px;

}

### **🔹 4. Propriedades de Borda**

* **border** – Adiciona borda ao redor dos elementos.

css

div {

border: 1px solid black;

}

* **border-radius** – Cantos arredondados.

css

button {

border-radius: 5px;

}

### **🔹 5. Propriedades de Exibição (Display)**

* **display** – Define como um elemento é exibido.

css

div {

display: block; /\* Pode ser block, inline, inline-block ou none \*/

}

* **visibility** – Define se um elemento é visível ou não.

css

p {

visibility: hidden; /\* ou visible \*/

}

### **🔹 6. Propriedades de Posicionamento**

* **position** – Define o tipo de posicionamento (relative, absolute, fixed, sticky).

css

div {

position: absolute;

top: 20px;

left: 30px;

}

* **top, bottom, left, right** – Define a posição exata dos elementos posicionados.

css

div {

position: relative;

top: 10px;

left: 50px;

}

### **🔹 7. Flexbox (para layouts responsivos)**

* **display: flex** – Ativa layout Flexbox.

css

.container {

display: flex;

}

* **justify-content** – Alinhamento horizontal dos elementos filhos.

css

.container {

justify-content: center;

}

* **align-items** – Alinhamento vertical dos elementos filhos.

css

.container {

align-items: center;

}

### **🔹 8. Propriedades de Listas**

* **list-style-type** – Define o estilo dos marcadores.

css

ul {

list-style-type: square;

}

* **list-style** – Define várias propriedades da lista em um comando.

css

ul {

list-style: none;

}

### **🔹 9. Transições e Animações Simples**

* **transition** – Animação suave nas mudanças de estilo.

css

button {

transition: background-color 0.3s;

}

button:hover {

background-color: blue;

}

### **🔹 10. Opacidade**

* **opacity** – Define a transparência do elemento.

css

div {

opacity: 0.5; /\* entre 0 e 1 \*/

}

## **🚩 Exemplo Completo (Com várias dessas propriedades reunidas)**

**HTML:**

html

<!DOCTYPE html>

<html lang="pt-br">

<head>

<link rel="stylesheet" href="estilos.css">

<meta charset="UTF-8">

<title>Exemplo Completo</title>

</head>

<body>

<div class="container">

<h1 id="titulo">Título Principal</h1>

<p class="texto">Este é um exemplo prático de várias propriedades CSS.</p>

<button class="botao">Clique Aqui</button>

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

</div>

</body>

</html>

**CSS (estilos.css):**

css

body {

font-family: Arial, sans-serif;

background-color: #f9f9f9;

}

.container {

background-color: #fff;

width: 80%;

margin: auto;

padding: 20px;

border-radius: 10px;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

#titulo {

text-align: center;

color: #333;

margin-bottom: 15px;

}

.texto {

font-size: 16px;

color: #555;

}

.botao {

padding: 10px;

background-color: #008CBA;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.botao:hover {

background-color: #005f73;

}

ul {

list-style-type: square;

padding-left: 20px;

}

## **🎓 Exercícios Práticos para os Alunos:**

### **📝 Atividade 1:**

Crie uma página HTML com três botões diferentes. Ao passar o mouse sobre cada um deles, a cor deve mudar suavemente usando a propriedade transition.

### **📝 Atividade 2:**

Crie um layout simples utilizando Flexbox, com três caixas coloridas lado a lado, com alinhamento vertical e horizontal ao centro.

### **📝 Atividade 3:**

Estilize uma lista não ordenada removendo os marcadores padrões e adicionando um ícone personalizado com CSS (list-style-image ou background-image).

## **💡 Sugestões para Aplicação em Aula:**

* Peça aos alunos que escolham propriedades CSS específicas para criar pequenos projetos.
* Realize debates sobre boas práticas.
* Incentive a criatividade através de desafios como recriar sites famosos usando CSS.

## **1) Propriedades de Fonte e Texto**

* **font-family**: Define a família de fontes (ex: Arial, Verdana, Times New Roman).
* **font-size**: Controla o tamanho do texto (ex: 16px, 1em, 1.5rem).
* **font-weight**: Ajusta a espessura da fonte (ex: normal, bold, 100, 700).
* **font-style**: Define se o texto será normal ou itálico (ex: normal, italic).
* **text-align**: Alinha o texto (ex: left, right, center, justify).
* **text-decoration**: Adiciona efeitos como sublinhado, linha sobre o texto, etc. (ex: underline, line-through).
* **line-height**: Define o espaçamento vertical entre linhas.
* **letter-spacing e word-spacing**: Ajusta o espaçamento entre letras e palavras, respectivamente.

## **2) Cores e Fundos**

* **color**: Define a cor do texto. Pode usar nomes de cor (ex: red), valores hexadecimais (ex: #ff0000), RGB (ex: rgb(255, 0, 0)) ou HSL (ex: hsl(0, 100%, 50%)).
* **background-color**: Define a cor de fundo do elemento.
* **background-image**: Define uma imagem de fundo (ex: url('imagem.jpg')).
* **background-repeat**: Define se a imagem de fundo repete (repeat, no-repeat, repeat-x, repeat-y).
* **background-position**: Especifica a posição da imagem de fundo (ex: center, top, left, ou valores em px e %).
* **background-size**: Ajusta o tamanho da imagem de fundo (ex: cover, contain, valores em px ou %).

## **3) Box Model (Espaçamentos e Bordas)**

* **margin**: Espaçamento externo do elemento, empurra o elemento em relação aos outros.  
  + Pode ser definido como valores únicos (ex: margin: 20px;) ou em cada lado (margin-top, margin-right, margin-bottom, margin-left).
* **padding**: Espaçamento interno, afasta o conteúdo das bordas do elemento.  
  + Semelhante ao margin, pode ser detalhado em cada lado.
* **border**: Cria uma borda em torno do elemento.  
  + Combina estilo, largura e cor (ex: border: 2px solid #000;).
* **border-radius**: Arredonda os cantos do elemento (ex: border-radius: 10px;).
* **box-sizing**: Define como o tamanho total do elemento é calculado (ex: content-box ou border-box).

## **4) Dimensões e Layout**

* **width e height**: Largura e altura do elemento (ex: 100px, 50%, auto).
* **max-width e min-width**: Define valores máximos e mínimos de largura.
* **display**: Controla como o elemento é exibido (ex: block, inline, inline-block, flex, grid).
* **position**: Define a forma de posicionamento (ex: static, relative, absolute, fixed, sticky).
* **top, right, bottom, left**: Ajusta a posição do elemento quando position for diferente de static.
* **overflow**: Determina o que acontece quando o conteúdo ultrapassa o tamanho do elemento (ex: visible, hidden, scroll, auto).
* **z-index**: Controla a sobreposição dos elementos (aplicado a elementos posicionados).

## **5) Flexbox**

* **display: flex;**: Habilita o contexto flex no container.
* **flex-direction**: Define a direção dos itens (row, column).
* **justify-content**: Alinha os itens ao longo do eixo principal (start, center, space-between, space-around).
* **align-items**: Alinha os itens ao longo do eixo transversal (start, center, stretch).
* **flex-wrap**: Define se os itens devem quebrar de linha quando não couberem (nowrap, wrap).
* **align-content**: Controla o espaçamento em múltiplas linhas de itens flex.

## **6) Grid Layout**

* **display: grid;**: Habilita o contexto grid no container.
* **grid-template-columns e grid-template-rows**: Define a quantidade e tamanho das colunas e linhas (ex: grid-template-columns: 1fr 1fr;).
* **grid-gap ou gap**: Define o espaçamento entre as células do grid.
* **justify-items, align-items**: Ajustam o alinhamento do conteúdo dentro das células.
* **grid-column, grid-row**: Permitem que um item ocupe várias colunas ou linhas.

## **7) Seletores e Especificidade**

* **Seletores de tipo**: Ex: p { ... } seleciona todos os parágrafos <p>.
* **Seletores de classe**: Ex: .minha-classe { ... } para elementos com class="minha-classe".
* **Seletores de ID**: Ex: #meu-id { ... } para elementos com id="meu-id".
* **Seletores de atributo**: Ex: input[type="text"] { ... }.
* **Seletores descendentes**: Ex: div p { ... } para parágrafos dentro de uma <div>.
* **Pseudo-classes**: Ex: a:hover, input:focus (estado do elemento).
* **Pseudo-elementos**: Ex: p::first-line, p::before, p::after (partes de um elemento).

## **8) Outros Conceitos Úteis**

* **float** e **clear**: Método mais antigo para posicionar elementos (muito usado antes do Flexbox e Grid).
* **Transições (transitions)**:  
  + **transition**: Cria efeitos de animação suave entre os valores de propriedades (ex: transition: all 0.3s ease;).
* **Transformações (transform)**:  
  + **transform: scale(), translate(), rotate()**: Rotaciona, movimenta ou redimensiona um elemento.
* **Animações (animations)**:  
  + **@keyframes**: Define quadro a quadro a animação.
  + **animation**: Aplica a animação ao elemento (ex: animation: nomeDaAnimacao 2s infinite;).
* **Responsividade**:  
  + **Media queries** (@media): Aplica estilos diferentes em tamanhos de tela distintos (ex: @media (max-width: 600px) { ... }).
* **Sombras**:  
  + **box-shadow**: Aplica sombra ao redor de um elemento (ex: box-shadow: 0 2px 5px rgba(0,0,0,0.3);).
  + **text-shadow**: Aplica sombra ao texto (ex: text-shadow: 1px 1px 2px #000;).

Elementos

Bloco x em linha:

Em bloco: <h1> <p> <div><ul>, etc.

vai ocupar todo o espaço horizontal disponível de onde ela está

Em linha:

Ela continua na mesma linha de onde foi chamada:

<strong> <img> <a>

As tags em bloco podem conter as tags em linha

<p> Oi <strong> Tudo bem? </strong> com você? </p>

Posso colocar div dentro de div, mas não posso colocar uma div dentro de um <p>

só a tag <a> que pode conter uma tag em bloco dentro (para fazer aquele pedaço todo ser um link.)

Elementos em linha podem conter outras tags em linha

Obs: no html é uma coisa

o css, mesmo que eu mude, o display, ela ainda será uma propriedade e evitará tag proibida dentro

**PRINCIPAIS FUNÇÕES DE ARRAY**

**Funções de Arrays mais utilizadas**

**1. push()**

Adiciona um elemento ao final do array.  
💡 Muito usado para construir arrays dinamicamente.

let frutas = ['maçã'];

frutas.push('banana');

console.log(frutas); // ['maçã', 'banana']

**2. pop()**

Remove o último elemento do array e o retorna.  
💡 Usado quando se quer remover o item mais recente adicionado.

let frutas = ['maçã', 'banana'];

let ultima = frutas.pop();

console.log(ultima); // 'banana'

console.log(frutas); // ['maçã']

**3. shift()**

Remove o primeiro elemento do array.  
💡 Útil quando você quer remover o item mais antigo.

let fila = ['a', 'b', 'c'];

fila.shift();

console.log(fila); // ['b', 'c']

**4. unshift()**

Adiciona elementos no início do array.  
💡 Útil para construir filas com prioridade.

let fila = ['b', 'c'];

fila.unshift('a');

console.log(fila); // ['a', 'b', 'c']

**5. length**

Retorna o tamanho do array.  
💡 Muito usado em for para percorrer o array.

let frutas = ['maçã', 'banana'];

console.log(frutas.length); // 2

**6. includes()**

Verifica se um valor existe dentro do array. Retorna true ou false.  
💡 Ideal para validações simples.

let cores = ['azul', 'vermelho'];

console.log(cores.includes('azul')); // true

**7. indexOf()**

Retorna o índice do elemento, ou -1 se não encontrar.  
💡 Usado para localizar posições.

let letras = ['a', 'b', 'c'];

console.log(letras.indexOf('b')); // 1

**8. splice()**

Adiciona, remove ou substitui elementos em qualquer posição.  
💡 Muito versátil.

let numeros = [1, 2, 3, 4];

numeros.splice(1, 2); // Remove 2 elementos a partir da posição 1

console.log(numeros); // [1, 4]

**9. slice()**

Copia parte do array, sem alterar o original.  
💡 Bom para dividir dados.

let nomes = ['Ana', 'Bruno', 'Carlos', 'Duda'];

let grupo = nomes.slice(1, 3);

console.log(grupo); // ['Bruno', 'Carlos']

**10. forEach()**

Executa uma função para cada item.  
💡 Alternativa moderna ao for tradicional.

let numeros = [1, 2, 3];

numeros.forEach(function(n) {

console.log(n \* 2);

});

**11. map()**

Cria um novo array transformando cada item.  
💡 Muito usado para aplicar mudanças.

let numeros = [1, 2, 3];

let dobrados = numeros.map(n => n \* 2);

console.log(dobrados); // [2, 4, 6]

**12. filter()**

Cria um novo array com itens que passam em uma condição.  
💡 Ideal para filtrar dados.

let numeros = [1, 2, 3, 4, 5];

let pares = numeros.filter(n => n % 2 === 0);

console.log(pares); // [2, 4]

**13. reduce()**

Transforma o array em um único valor (soma, média, etc).  
💡 Excelente para cálculos agregados.

let numeros = [1, 2, 3];

let soma = numeros.reduce((total, n) => total + n, 0);

console.log(soma); // 6

**14. sort()**

Ordena os elementos.  
💡 Cuidado: modifica o array original!

let nomes = ['Carlos', 'Ana', 'Bruno'];

nomes.sort();

console.log(nomes); // ['Ana', 'Bruno', 'Carlos']

**15. reverse()**

Inverte a ordem do array.  
💡 Útil para listas decrescentes.

let numeros = [1, 2, 3];

numeros.reverse();

console.log(numeros); // [3, 2, 1]

**Objetivo do Jogo**

O jogador deve **adivinhar um número aleatório entre 1 e 100**, e o sistema fornece dicas se o número é maior ou menor, até que o jogador acerte.

**🔧 1. Estrutura HTML**

html

CopiarEditar

<div class="game-container">

<h2>🎯 Adivinhe o Número</h2>

<p>Digite um número entre 1 e 100</p>

<input type="number" id="guessInput" min="1" max="100">

<button onclick="checkGuess()">Tentar</button>

<div class="message" id="message"></div>

</div>

**✨ O que temos aqui:**

* <input type="number">: permite que o jogador digite um número.
* <button onclick="checkGuess()">: chama a função checkGuess() quando o botão é clicado.
* <div id="message">: onde são mostradas mensagens como “Muito alto” ou “Acertou!”.

**🎨 Estilização com CSS**

css

CopiarEditar

input[type="number"] {

padding: 10px;

width: 100%;

}

button {

background-color: #3498db;

color: white;

}

.message {

margin-top: 15px;

font-weight: bold;

}

**✨ O que foi feito:**

* Estilização básica para input e botão.
* Mensagens em destaque para o jogador.

**🧠 Lógica em JavaScript**

**1. Gerar o número aleatório**

js

CopiarEditar

const numeroSecreto = Math.floor(Math.random() \* 100) + 1;

**📌 Explicação:**

* Math.random() → retorna um número decimal entre 0 e 1.
* \* 100 → transforma em um número entre 0 e 99.999...
* Math.floor() → arredonda para baixo, deixando um número inteiro.
* + 1 → para garantir que o número vá de 1 a 100.

**2. Função principal: checkGuess()**

js

CopiarEditar

function checkGuess() {

const input = document.getElementById("guessInput");

const mensagem = document.getElementById("message");

const chute = parseInt(input.value);

**📌 Explicação:**

* document.getElementById() → seleciona elementos do HTML.
* parseInt() → converte o valor digitado (string) em número inteiro.

**3. Validação do valor**

js

CopiarEditar

if (isNaN(chute) || chute < 1 || chute > 100) {

mensagem.textContent = "Por favor, digite um número entre 1 e 100.";

return;

}

**📌 Explicação:**

* isNaN() → verifica se **não** é um número.
* Impede que o usuário insira valores inválidos.

**4. Comparação com o número secreto**

js

CopiarEditar

if (chute < numeroSecreto) {

mensagem.textContent = "🔼 Muito baixo! Tente um número maior.";

} else if (chute > numeroSecreto) {

mensagem.textContent = "🔽 Muito alto! Tente um número menor.";

} else {

mensagem.textContent = `🎉 Parabéns! Você acertou em ${tentativas} tentativa(s).`;

input.disabled = true;

}

**📌 Explicação:**

* Usamos if, else if, else para comparar o chute com o número secreto.
* Quando o jogador acerta, o input é desativado com input.disabled = true.

**5. Limpeza e foco no campo**

js

CopiarEditar

input.value = "";

input.focus();

**📌 Explicação:**

* Limpa o valor digitado para uma nova tentativa.
* input.focus() posiciona o cursor no campo automaticamente.

**✅ Resumo das Funções Usadas**

| **Função / Comando** | **O que faz** |
| --- | --- |
| Math.random() | Gera número decimal aleatório entre 0 e 1 |
| Math.floor() | Arredonda um número para baixo |
| parseInt() | Converte string em número inteiro |
| isNaN() | Verifica se algo **não** é um número |
| getElementById() | Acessa um elemento HTML pelo id |
| textContent | Define o conteúdo de texto de um elemento |
| disabled = true | Desativa o campo de input |
| focus() | Move o cursor para o campo de entrada |

**Melhorias Implementadas**

**1. Contador de Tentativas**

* Agora há um **limite de 10 tentativas** para acertar o número.
* O jogador vê na tela quantas tentativas ainda restam:

js

CopiarEditar

tentativasRestantes = limiteTentativas - tentativas;

document.getElementById("tentativasInfo").textContent = `Tentativas restantes: ${tentativasRestantes}`;

**2. Encerramento do jogo**

* Quando o jogador acerta ou atinge o limite de tentativas:

js

CopiarEditar

if (tentativas >= limiteTentativas) {

mensagem.textContent = `😢 Fim de jogo! O número era ${numeroSecreto}.`;

encerrarJogo();

}

* O campo de input é desabilitado e aparece um botão para reiniciar.

**3. Botão "🔁 Jogar Novamente"**

* Aparece somente ao fim do jogo, para reiniciar tudo:

js

CopiarEditar

document.getElementById("restartBtn").style.display = "block";

* Ele chama a função reiniciarJogo() que reinicializa o jogo com um novo número aleatório.

!@$$$$$$$$$$$$$$$$$@#

**Resolução dos Exercícios – Funções em JavaScript**

**📘 Nível 1 – Básico**

**1. Função de saudação personalizada**

js

CopiarEditar

function saudacao(nome) {

return "Olá, " + nome + "!";

}

console.log(saudacao("João")); // Olá, João!

**2. Função para calcular o dobro de um número**

js

CopiarEditar

function dobrar(numero) {

return numero \* 2;

}

console.log(dobrar(4)); // 8

**🔧 Nível 2 – Aplicações reais simples**

**3. Função para calcular desconto**

js

CopiarEditar

function aplicarDesconto(preco, porcentagem) {

return preco - (preco \* (porcentagem / 100));

}

console.log(aplicarDesconto(100, 10)); // 90

**4. Função para calcular IMC**

js

CopiarEditar

function calcularIMC(peso, altura) {

return (peso / (altura \* altura)).toFixed(2);

}

console.log(calcularIMC(70, 1.75)); // 22.86

**🧠 Nível 3 – Retorno lógico**

**5. Verificar maioridade**

js

CopiarEditar

function maiorDeIdade(idade) {

return idade >= 18;

}

console.log(maiorDeIdade(17)); // false

console.log(maiorDeIdade(20)); // true

**6. Formatador de nome**

js

CopiarEditar

function formatarNome(nome, sobrenome) {

const nomeFormatado = nome.charAt(0).toUpperCase() + nome.slice(1);

const sobrenomeFormatado = sobrenome.charAt(0).toUpperCase() + sobrenome.slice(1);

return nomeFormatado + " " + sobrenomeFormatado;

}

console.log(formatarNome("ana", "silva")); // Ana Silva

**💼 Nível 4 – Funções combinadas**

**7. Cálculo de salário com bônus**

js

CopiarEditar

function calcularSalario(base, bonusPercentual) {

return base + (base \* (bonusPercentual / 100));

}

function formatarSalario(valor) {

return "R$ " + valor.toFixed(2);

}

let salarioFinal = calcularSalario(2000, 10);

console.log(formatarSalario(salarioFinal)); // R$ 2200.00

**8. Exibir lista de produtos**

js

CopiarEditar

function mostrarProdutos(produtos) {

produtos.forEach(produto => {

console.log(produto);

});

}

mostrarProdutos(["Mouse", "Teclado", "Monitor"]);

// Mouse

// Teclado

// Monitor

**🏁 Desafio Final – Sistema de usuário**

js

CopiarEditar

function solicitarDados() {

const nome = prompt("Digite seu nome:");

const idade = parseInt(prompt("Digite sua idade:"));

const salario = parseFloat(prompt("Digite seu salário:"));

return { nome, idade, salario };

}

function verificarIdade(idade) {

return idade >= 18 ? "maior de idade" : "menor de idade";

}

function mostrarResumo(dados) {

const status = verificarIdade(dados.idade);

const salarioReajustado = dados.salario \* 1.10;

alert(`Nome: ${dados.nome}

Idade: ${dados.idade} (${status})

Salário com reajuste: R$ ${salarioReajustado.toFixed(2)}`);

}

// Execução

const usuario = solicitarDados();

mostrarResumo(usuario);