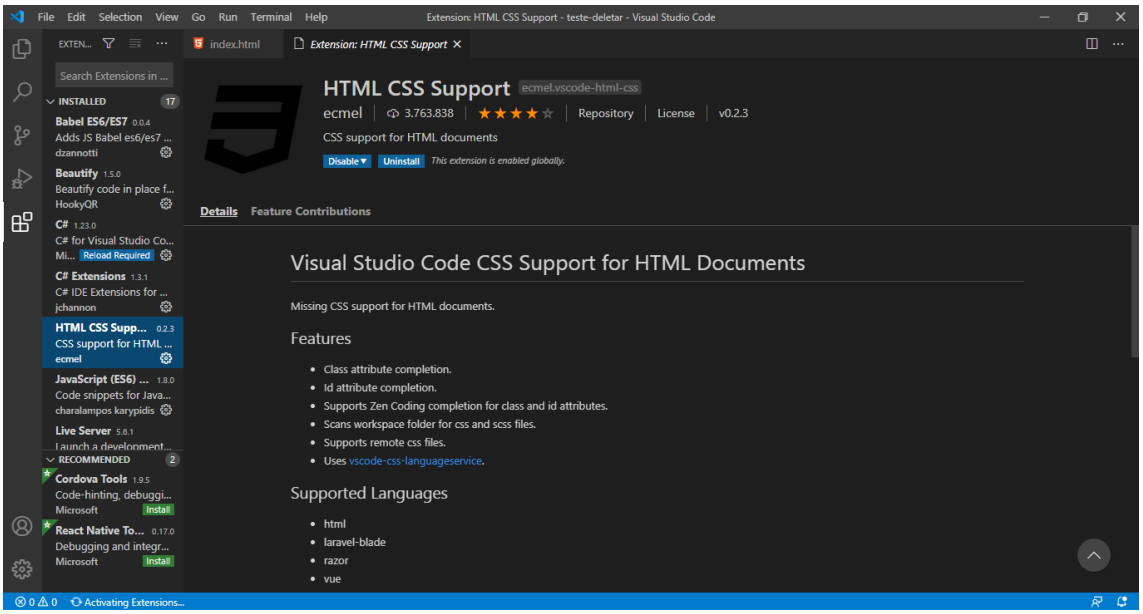
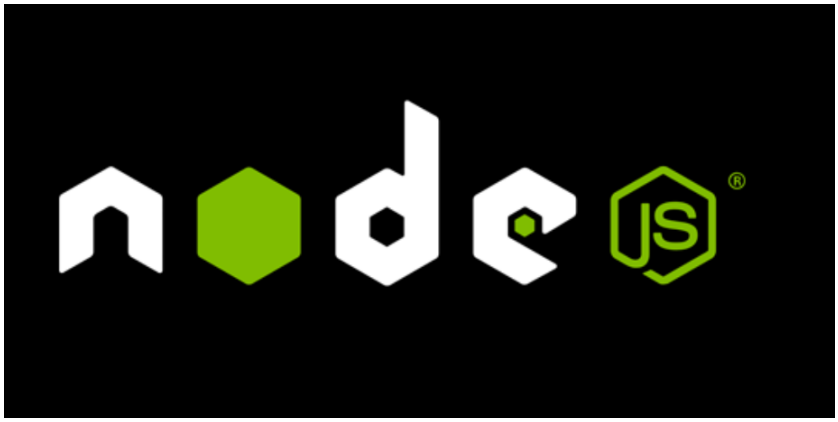
Instalando o VS Code



Instalando as extensões

- vscode-html-css
- auto-rename-tag
- LiveServer
- vscode-icons
- HTML CSS Support
- HTML end Tag Labels
- JavaScript (ES6) code snippets

Node.JS



O que é Lógica?

A lógica é a maneira como pensamos e organizamos nossas ideias para tomar decisões ou resolver problemas. É como seguir um caminho de raciocínio que faz sentido.

Imagine que você está jogando um jogo de adivinhação. Você recebe dicas e pistas, e precisa pensar sobre o que essas pistas significam para descobrir a resposta correta.

Isso é usar a lógica. Você está conectando as informações de forma inteligente para chegar a uma conclusão.

Na vida cotidiana, usamos a lógica o tempo todo. Quando fazemos planos, resolvemos quebra-cabeças, escolhemos entre opções ou entendemos como as coisas funcionam, estamos aplicando a lógica para tomar decisões e resolver desafios.

O que é lógica de programação?

A lógica de programação é uma maneira de pensar e estruturar as etapas ou instruções necessárias para resolver um problema por meio de um programa de computador.

É como criar um plano detalhado para que o computador possa entender e executar as ações corretas.

Pense nisso como dar instruções muito precisas para alguém que está seguindo um conjunto de regras bem definidas.

A lógica de programação envolve quebrar um problema complexo em partes menores, definir a ordem em que essas partes devem ser executadas e usar conceitos como repetição (fazer algo várias vezes) e decisão (escolher entre diferentes opções) para alcançar o resultado desejado.

Os programadores usam a lógica de programação para criar algoritmos, que são sequências específicas de passos que um computador pode entender e executar. É como escrever um roteiro para que o computador saiba exatamente o que fazer para resolver um problema ou realizar uma tarefa.

O que é linguagem de programação?

Linguagem de programação é um conjunto de regras e símbolos que os programadores usam para escrever instruções que um computador pode entender e executar.

Ela atua como uma ponte de comunicação entre humanos e máquinas, permitindo que você escreva um código para que o computador possa realizar uma ou várias tarefas específicas.

Existem muitas linguagens de programação diferentes, cada uma com suas próprias características e finalidades. Algumas linguagens são mais adequadas para tarefas

específicas, enquanto outras são mais versáteis e podem ser usadas em uma variedade de contextos. Aqui estão alguns exemplos populares de linguagens de programação:

Python: Conhecida por sua sintaxe legível e fácil de entender, é usada para desenvolvimento web, análise de dados, automação etc.

Java: Uma linguagem versátil usada em desenvolvimento de aplicativos, jogos, sistemas embarcados etc.

C++: Uma extensão da linguagem C que adiciona recursos de programação orientada a objetos, usada em desenvolvimento de software, jogos, sistemas de tempo real, entre outros.

JavaScript: Linguagem de script utilizada principalmente no desenvolvimento web para criar interações dinâmicas em páginas.

Ruby: Conhecida por sua simplicidade e produtividade, é usada para desenvolvimento web e automação.

C#: Desenvolvida pela Microsoft, é amplamente usada para criar aplicativos Windows e jogos usando a plataforma Unity.

PHP: Linguagem frequentemente usada para desenvolvimento web e criação de sites dinâmicos.

SQL: Linguagem de consulta de banco de dados usada para gerenciar e manipular dados em bancos de dados relacionais.

Esses são apenas alguns exemplos, e há muitas outras linguagens de programação por aí, cada uma com suas próprias vantagens e casos de uso específicos. Cada linguagem tem sua própria sintaxe e conjunto de regras, mas todas servem ao propósito de permitir que os programadores expressem instruções para o computador de maneira compreensível e eficaz.

O que é Algoritmo?

Um algoritmo é um conjunto finito e organizado de passos ou instruções que são seguidos para realizar uma tarefa ou resolver um problema específico. É uma sequência lógica de ações que, quando executadas corretamente, levam a um resultado desejado.

Pense em um algoritmo como uma receita de culinária. A receita é um conjunto de instruções claras e precisas sobre o que fazer e em que ordem para preparar um prato específico. Da mesma forma, um algoritmo fornece uma série de etapas bem definidas para realizar uma ação ou chegar a uma conclusão.

Os algoritmos são usados em várias áreas, incluindo ciência da computação, matemática, engenharia e muitas outras disciplinas. Eles são essenciais para a programação de computadores, pois os programadores escrevem algoritmos para criar software que execute tarefas específicas.

Um exemplo simples de algoritmo seria o processo de multiplicação de dois números:

1. Escreva os dois números a serem multiplicados.
2. Multiplique o primeiro número pelo segundo número.

3. O resultado é o produto da multiplicação.

Nesse caso, os passos são claros e precisos, e se você seguir esses passos corretamente, obterá o resultado correto da multiplicação. Isso é um algoritmo em ação.

Começando com a prática

Um pouquinho de HTML, estrutura básica

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Algoritmo</title>
  <style>
    / SEU CSS AQUI /
  </style>
</head>
<body>

    <!- SEU CÓDIGO HTML AQUI -->

  <script>
    // SEU CÓDIGO JAVASCRIPT AQUI
  </script>
</body>
</html>
```

Algumas Tags

Para iniciar seus estudos em JavaScript, é útil ter conhecimento básico de HTML, pois o HTML é a estrutura sobre a qual os scripts JavaScript são executados nos navegadores da web. Aqui estão algumas das tags básicas de HTML que um aluno deve estar familiarizado:

1. **<html>:** Define o início e o fim do documento HTML.
2. **<head>:** Contém informações sobre o documento HTML, como metadados, scripts, estilos, etc.
3. **<title>:** Define o título da página, exibido na barra de título do navegador ou na guia.
4. **<body>:** Contém o conteúdo visível da página, como texto, imagens, links, etc.
5. **<h1>, <h2>, ..., <h6>:** Define os títulos e subtítulos da página, com <h1> sendo o título mais importante e <h6> o menos importante.

6. **<p>:** Define um parágrafo de texto.
7. **<a>:** Define um hyperlink, usado para criar links para outras páginas ou recursos.
8. **<img>:** Define uma imagem.
9. **<ul>:** Define uma lista não ordenada.
10. **<ol>:** Define uma lista ordenada.
11. **<li>:** Define um item de lista, dentro de <ul> ou <ol>.
12. ** :** Define uma divisão ou seção genérica em um documento.
13. **:** Define uma pequena seção de conteúdo em linha.
14. **<input>:** Define um campo de entrada de formulário, como caixa de texto, botão, etc.
15. **<button>:** Define um botão clicável.
16. **<script>:** Define um script JavaScript que será executado no navegador.

Essas são algumas das tags básicas de HTML que você encontrará ao começar a aprender JavaScript. Compreender essas tags é importante porque o JavaScript frequentemente interage com o conteúdo HTML da página para modificar seu comportamento e aparência.

Comandos de saída

Em programação, os comandos de saída são instruções que permitem que um programa exiba informações na tela ou em outro dispositivo de saída, como um arquivo de texto.

Esses comandos são usados para comunicar informações aos usuários ou para verificar o estado do programa durante a execução.

Os comandos de saída variam dependendo da linguagem de programação que você está usando, mas geralmente envolvem a exibição de valores de **variáveis**, mensagens de texto ou resultados de cálculos.

Alguns exemplos de comandos de saída em diferentes linguagens de programação são:

JavaScript

```
console.log("Olá, mundo!"); // Exibe a mensagem no console do navegador
```

Python:

```
print("Olá, mundo!")  # Exibe a mensagem na tela
```

C++:

```
#include <iostream>
using namespace std;

int main() {
```

```
    cout << "Olá, mundo!" << endl; // Exibe a mensagem na tela
    return 0;
}
```

Java:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Olá, mundo!"); // Exibe a mensagem na tela
    }
}
```

C#:

```
using System;
class Program {
    static void Main() {
        Console.WriteLine("Olá, mundo!"); // Exibe a mensagem na tela
    }
}
```

Em todos esses exemplos, os comandos de saída são usados para mostrar uma mensagem simples na tela ou no console. Eles são uma parte fundamental da programação, pois permitem que os programas interajam com os usuários ou forneçam informações importantes durante a execução.

Saída de dados no JavaScript

Console.Log

No JavaScript, para exibir informações no console do navegador, você pode usar o método **console.log()**. Aqui está um exemplo simples de como usar o comando de saída no console usando JavaScript:

```
console.log("Olá, mundo!");  // Exibe "Olá, mundo!" no console do navegador
```

Você pode abrir o console do navegador (geralmente pressionando F12 ou clicando com o botão direito e selecionando "Inspecionar" e, em seguida, navegando para a guia "Console") para ver a saída resultante.

O **console.log()** é amplamente utilizado para depuração e registro de informações durante o desenvolvimento de aplicativos da web. Ele permite que você visualize variáveis, mensagens e outros dados importantes enquanto seu código está sendo executado no navegador.

Explicando o console.log

console: "**Console**" se refere a uma interface onde você pode ver mensagens, erros e informações enquanto um programa está sendo executado. No contexto do JavaScript em um navegador, o "**console**" é uma ferramenta que permite a comunicação entre seu código JavaScript e o ambiente do navegador.

log: "**Log**" é uma abreviação de "logar" (registar), que é o ato de registrar informações em um log ou arquivo. No contexto do JavaScript, "log" se refere a exibir uma mensagem ou valor no console para que você possa visualizá-lo durante a execução do programa.

Portanto, **console.log** é um método que faz parte do objeto console no JavaScript, permitindo que você registre informações no console.

Quando você chama **console.log()**, você está instruindo o navegador a exibir uma mensagem ou valor específico no console para que você possa acompanhar o que está acontecendo no seu código enquanto ele é executado. Isso é particularmente útil para depurar erros, verificar o valor de variáveis e acompanhar o fluxo do seu programa.

Exemplos de como utilizar o console.log

```
console.log('Olá mundo!!!');
console.log('Um exemplo\nde texto\nquebrando linhas');
console.log('Um exemplo\n\tde texto\n\t\tquebrando linhas');

console.log('Posso escrever com aspas simples');
console.log("Posso escrever com aspas duplas")
```

Alert

Em JavaScript, o comando **alert()** é uma função que exibe uma caixa de diálogo na janela do navegador com uma mensagem especificada. É comumente usada para fornecer feedback imediato ao usuário ou para exibir mensagens de aviso.

```
alert("Sua mensagem aqui");
```

Dentro dos parênteses, você pode inserir qualquer mensagem de texto que deseja exibir. Por exemplo:

```
alert("Olá, mundo!");
```

Quando esse código é executado em um navegador da web, uma caixa de diálogo será exibida com o texto **"Olá, mundo!"** e um botão **"OK"** para que o usuário possa fechar a caixa de diálogo.

É importante notar que o **alert()** é uma função de JavaScript específica para interações de usuário em navegadores da web. Ele não está disponível em outros contextos de execução JavaScript, como no Node.js, que é usado no lado do servidor.

Id, innerHtml e document.getElementById

Vamos criar um exemplo simples de como você pode usar JavaScript para selecionar um elemento HTML por seu ID e em seguida, inserir texto dentro desse elemento.

Vou criar um parágrafo com um ID específico e em seguida, usar JavaScript para inserir texto dentro desse parágrafo.

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Exemplo de Inserção de Texto com JavaScript</title>
</head>
<body>
   <p id="meuParagrafo">Este é um parágrafo vazio.</p>

   <script src="script.js"></script>
</body>
</html>
```

JavaScript

```
// Seleciona o elemento do parágrafo pelo ID
var paragrafo = document.getElementById("meuParagrafo");

// Insere texto dentro do parágrafo
paragrafo.innerHTML = "Este é o texto inserido pelo JavaScript!";
```

Neste exemplo:

1. No HTML, temos um parágrafo com o ID "meuParagrafo".
2. No JavaScript, usamos document.getElementById("meuParagrafo") para selecionar o parágrafo pelo seu ID e armazená-lo na variável paragrafo.
3. Em seguida, usamos paragrafo.innerHTML para definir o conteúdo HTML dentro desse parágrafo. Neste caso, estamos simplesmente atribuindo uma string de texto para inserir no parágrafo.

Ao abrir esta página HTML no navegador, o JavaScript será executado e o texto ***"Este é o texto inserido pelo JavaScript!"*** será adicionado ao parágrafo, resultando em:

```
<p id="meuParagrafo">Este é o texto inserido pelo JavaScript!</p>
```

Isso demonstra como você pode manipular o conteúdo de elementos HTML usando JavaScript.

InnerHTML, InnerText e textContent

O innerHTML, innerText e textContent são propriedades de elementos HTML em JavaScript que permitem manipular o conteúdo de texto dentro desses elementos, mas há diferenças importantes entre eles:

innerHTML:

- A propriedade innerHTML permite acessar e modificar o conteúdo HTML dentro de um elemento. Isso significa que ele não apenas manipula texto, mas também qualquer conteúdo HTML, como tags, atributos, etc.

- Por exemplo, se você tem um elemento , e você define document.getElementById("exemplo").innerHTML = "Texto em negrito", o texto "Texto em negrito" será exibido em negrito, pois a tag foi interpretada como HTML.

innerText:

- A propriedade innerText define ou retorna o texto visível dentro de um elemento, removendo qualquer formatação HTML. Ele representa somente o texto que é visível na tela, ignorando quaisquer tags HTML dentro do elemento.

textContent:

- Similar ao **innerText**, a propriedade **textContent** retorna o conteúdo de texto de um nó e de todos os seus descendentes. Ele não interpreta as tags HTML, apenas retorna o texto puro.

Em resumo, **innerHTML** manipula o conteúdo HTML dentro de um elemento, enquanto **innerText** e **textContent** lidam apenas com o texto visível dentro do elemento, com textContent sendo mais estrito, pois não interpreta HTML. A escolha entre eles depende do que você está tentando alcançar em sua manipulação do DOM.

Concatenação, Interpolação e Template String

```
let meuNome = 'Rodrigo';
let meuSobrenome = 'Dionisio';

console.log(meuNome + ' ' + meuSobrenome);
console.log(meuNome , meuSobrenome);
console.log(${meuNome} ${meuSobrenome});
```

Exercícios

Utilizando o comando **console.log, alert** e **document.getElementById** com **innerHTML** faça os exercícios a seguir:

Exercício 1:

Escreva um programa que exiba a mensagem **"Olá, mundo!"**.

Exercício 2:

Escreva um programa que exiba o seu nome.

Exercício 3:

Crie um programa que exiba a soma de dois números.

Exercício 4:

Crie um programa que exiba o dobro de um número.

Exercício 5:

Escreva um programa que exiba a idade de um dos seus colegas.

Exercício 6:

Escreva um programa que exiba a mensagem **"Estou aprendendo JavaScript!"** 5 vezes.

Exercício 7:

Crie um programa que exiba a sequência de números de 1 a 10.

Exercício 8:

Escreva um programa que exiba a mensagem **"Feliz Ano Novo!"** 3 vezes.

Exercício 9:

Escreva um programa que exiba o resultado de um número elevado ao cubo.

Exercício 10:

Escreva um programa que exiba o seu nome, sobrenome e idade.