



HELLO WORLD



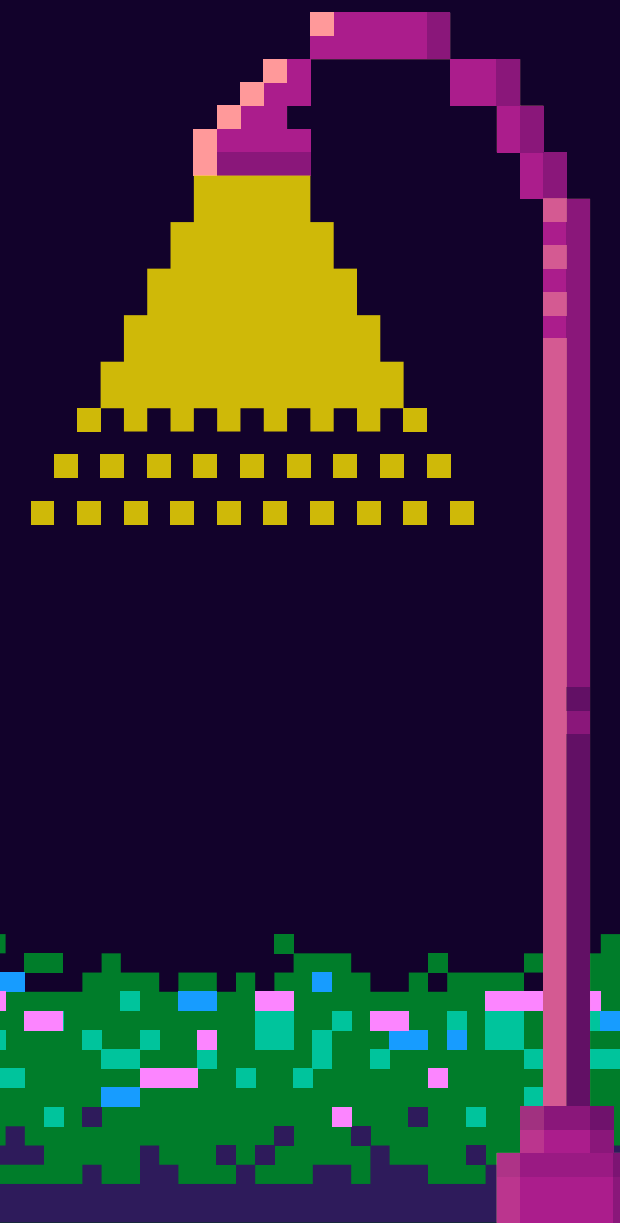
LÓGICA DE PROGRAMAÇÃO



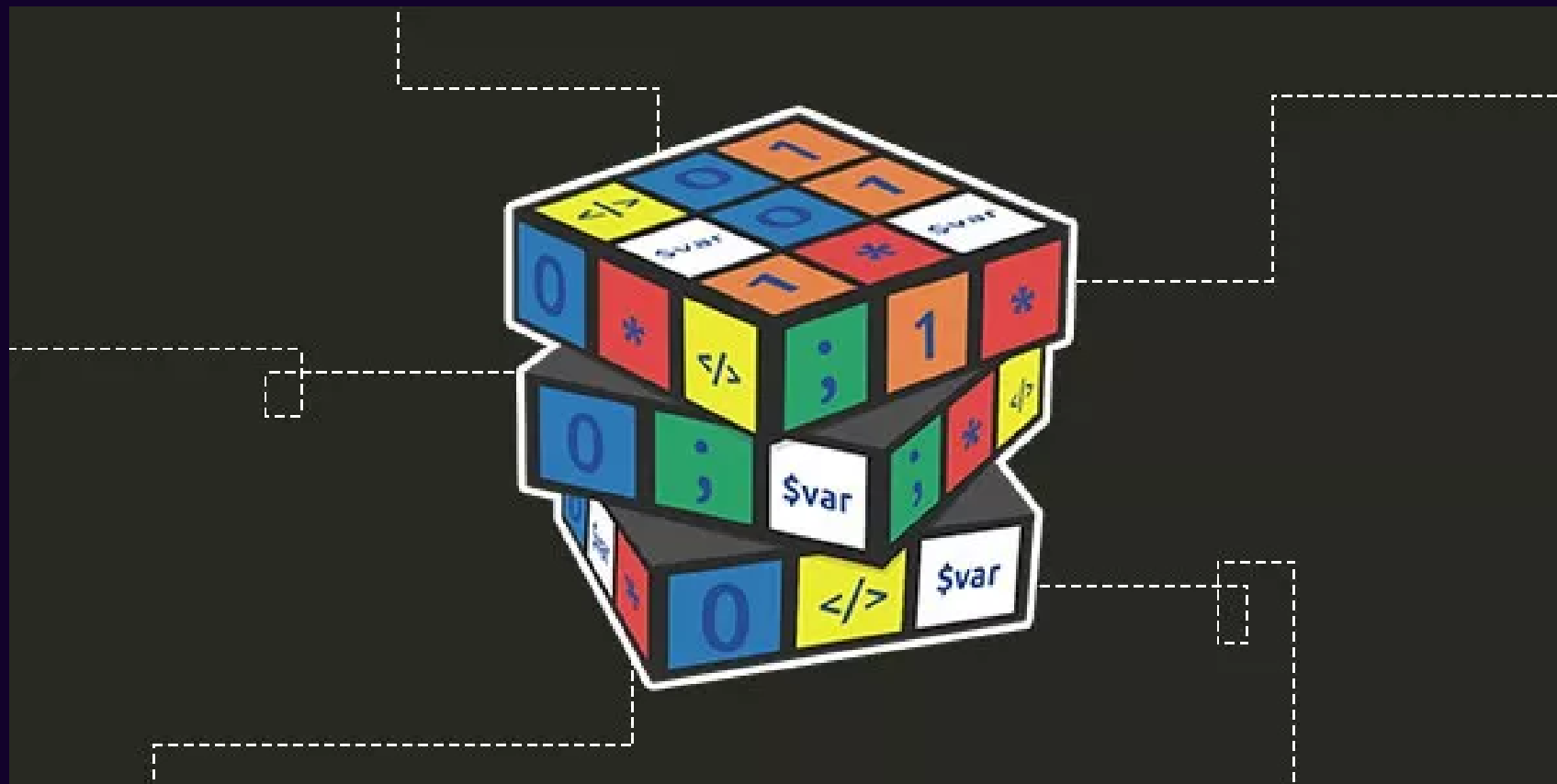
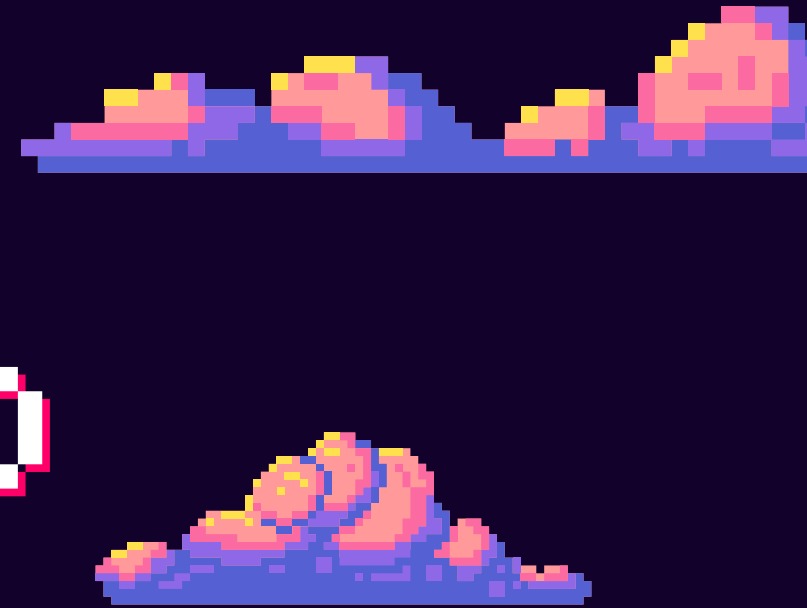
O QUE É LÓGICA DE PROGRAMAÇÃO?

A lógica de programação é o conjunto de regras e técnicas que os programadores utilizam para projetar e desenvolver programas de computador. É a habilidade de pensar de forma lógica e estruturada, decompondo um problema complexo em etapas mais simples. O objetivo é criar algoritmos claros e eficientes, que possam ser traduzidos em código de programação.

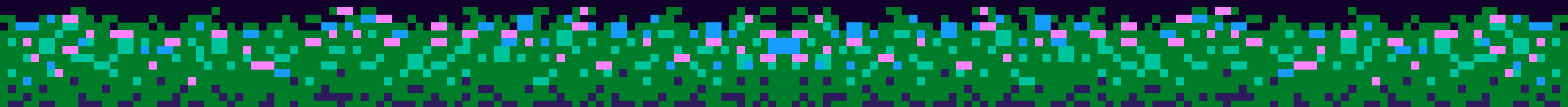
Um bom programador não é apenas alguém que sabe escrever código, mas também alguém que sabe pensar de forma lógica e resolver problemas de maneira eficaz. Isso envolve a capacidade de identificar padrões, organizar informações e criar soluções elegantes para desafios específicos.

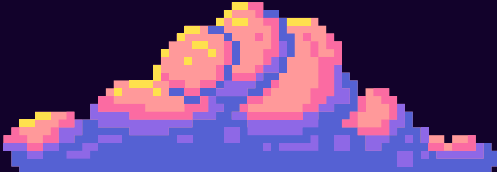


A IMPORTÂNCIA DA LÓGICA DE PROGRAMAÇÃO

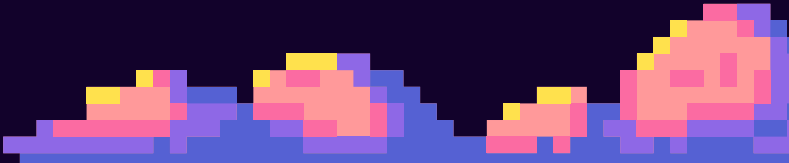


- Resolução de Problemas
- Eficiência
- Facilidade de Manutenção
- Adaptação a Novas Linguagens
- Algoritmos





PSEUDOCÓDIGO



Algoritmo Evolutivo

Gera populacao inicial(Pop)

nº de gerações = 1

melhor indivíduo = melhor indivíduo(Pop)

Enquanto nº de gerações < nº máximo de gerações **faça**

 Pop₊₁ = ∅

 reproducao(Pop, Pop₊₁)

 Pop = Pop₊₁

 Busca local(Pop)

 Atualizar melhor indivíduo

 nº de gerações++

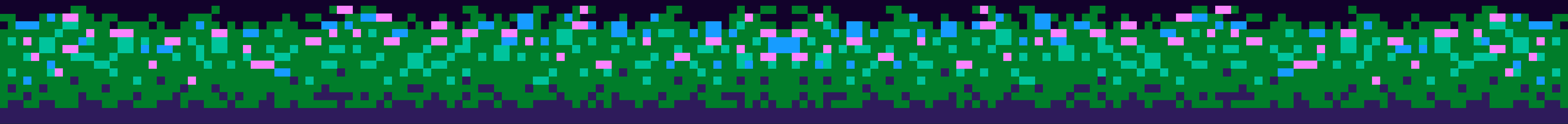
Fim-Enquanto

Imprimir melhor indivíduo

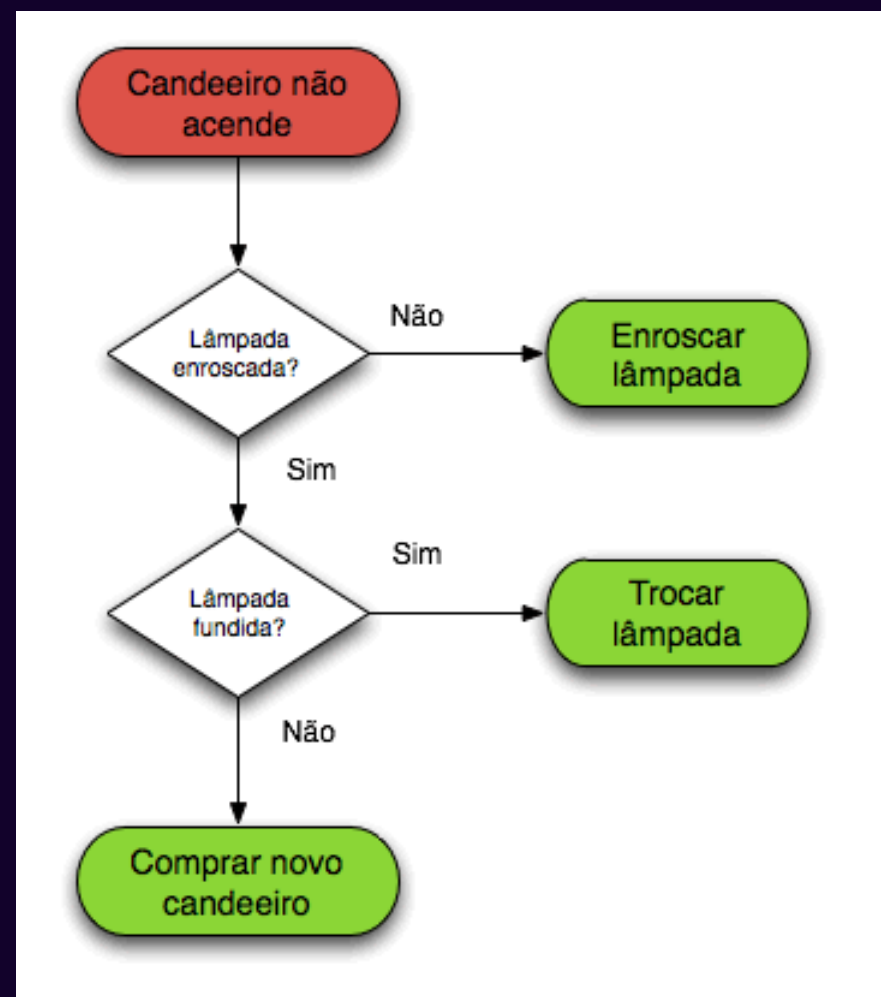
Fim Algoritmo Evolutivo

Pseudocódigo é uma forma de representar código, sejam algoritmos, funções ou outros processos, usando uma combinação de linguagem natural e elementos que se parecem com linguagem de programação.

É chamado de “pseudo” código porque não é realmente executável. Em vez disso, é uma forma de humanos entenderem facilmente e planejarem a lógica de codificação. Ele descreve os passos de um programa de maneira fácil de entender, mas ainda é detalhado o suficiente para ser rapidamente convertido em uma linguagem de programação específica.



A CONEXÃO ENTRE LÓGICA DE PROGRAMAÇÃO E ALGORITMOS



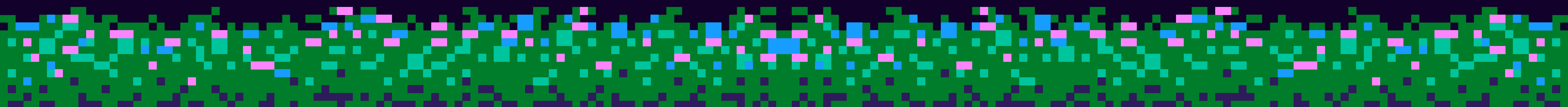
A lógica de programação está intrinsecamente ligada à criação de algoritmos. Um algoritmo é uma sequência finita de ações que, quando seguidas, resolvem um problema ou realizam uma tarefa específica. A lógica é usada para projetar e otimizar algoritmos, garantindo que eles sejam eficazes e eficientes.

O QUE É JAVASCRIPT?

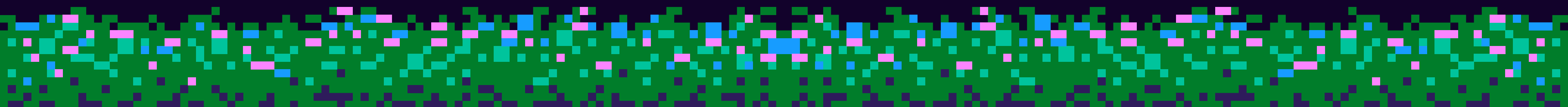
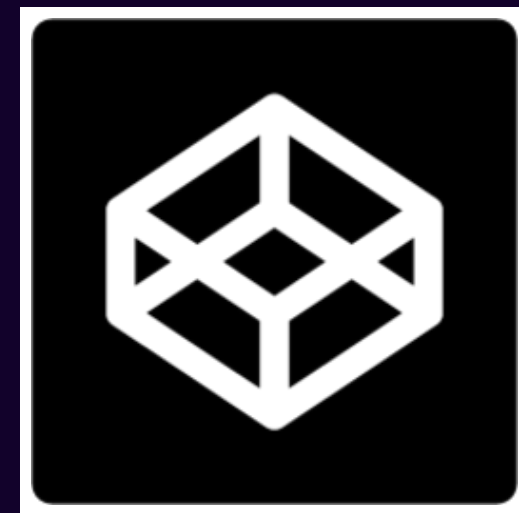
JavaScript é uma poderosa linguagem de programação multiparadigma que pode adicionar interatividade a um site. Foi inventado por Brendan Eich.

JavaScript é versátil e amigável para iniciantes. Com mais experiência, você poderá criar jogos, gráficos 2D e 3D animados, aplicativos abrangentes baseados em banco de dados e muito mais!

- Interfaces de programação de aplicativos de navegador (APIs)
- APIs de terceiros
- Estruturas e bibliotecas de terceiros



E COMO POSSO PRATICAR?







VARIÁVEIS



Variables são contêineres que armazenam valores. Você começa declarando uma variável com a palavra-chave `let`, seguida do nome que você dá à variável:

```
let myVariable;
```

Depois de declarar uma variável, você pode atribuir um valor a ela:


```
myVariable = "Bob";
```

Além disso, você pode fazer essas duas operações na mesma linha:

```
let myVariable = "Bob";
```

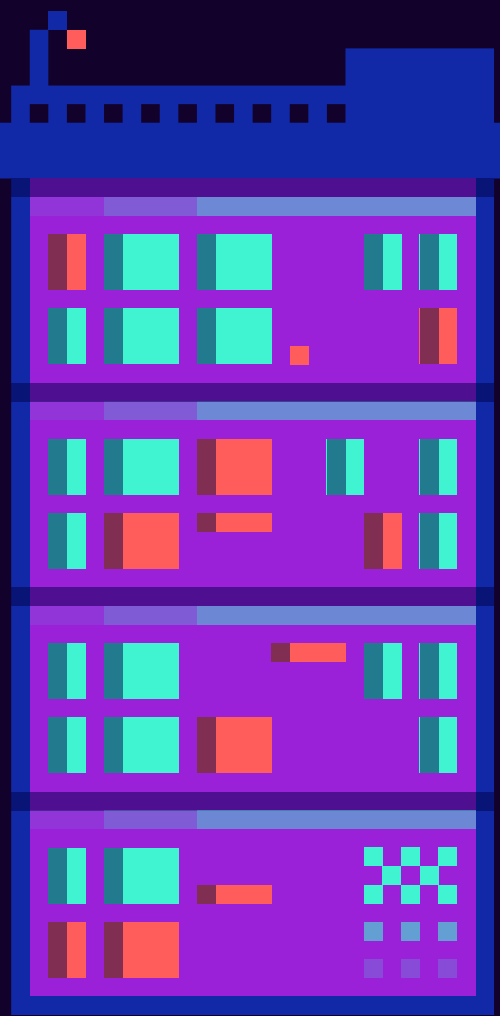
Você recupera o valor chamando o nome da variável:

```
myVariable;
```



Depois de atribuir um valor a uma variável, você pode alterá-lo posteriormente no código:

```
let myVariable = "Bob";  
myVariable = "Steve";
```



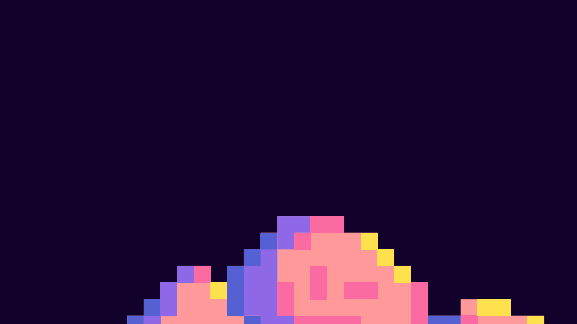


CONSTANTES

Nos primeiros dias do JavaScript, não existiam constants. No JavaScript moderno, temos a palavra-chave `const`, que nos permite armazenar valores que nunca podem ser alterados:

```
const diasNaSemana = 7;  
const horasNoDia = 24;
```

`const` funciona exatamente da mesma maneira que `let`, exceto que você não pode atribuir um novo valor a `const`. No exemplo a seguir, a segunda linha geraria um erro:



```
const diasNaSemana = 7;  
diasNaSemana = 8;
```

TIPOS DE DADOS

Variável	Explicação	Exemplo
String	Esta é uma sequência de texto conhecida como string. Para significar que o valor for uma string, coloque-a entre aspas simples.	<code>let myVariable = 'Bob';</code>
Number	Isto é um número. Os números não têm aspas.	<code>deixe minhaVariável = 10;</code>
Boolean	Este é um valor Verdadeiro/Falso. As palavras <code>true</code> e <code>false</code> são palavras-chave especiais que não precisam de aspas.	<code>let myVariable = true;</code>
Array	Esta é uma estrutura que permite armazenar vários valores em um único referência.	<code>let myVariable = [1, 'Bob', 'Steve', 10];</code> Refere-se a cada membro do array assim: <code>myVariable[0], myVariable[1],</code> etc.
Object	Isso pode ser qualquer coisa. Tudo em JavaScript é um objeto e pode ser armazenados em uma variável. Tenha isso em mente enquanto aprende.	<code>let myVariable = document.querySelector('h1');</code> Todos os exemplos acima também.

JavaScript é uma linguagem dinâmica com tipos dinâmicos. As variáveis em JavaScript não estão diretamente associadas a nenhum tipo de valor específico, e qualquer variável pode receber (e reatribuir) valores de todos os tipos.

Comentários

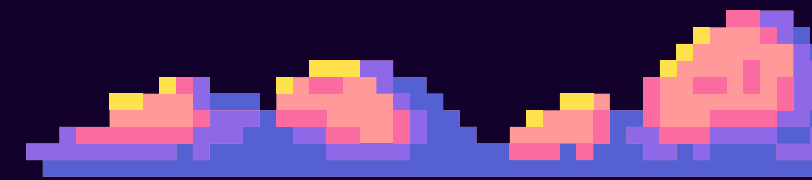
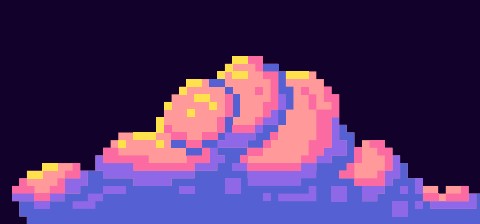
```
/*  
Tudo no meio é um comentário.  
*/
```

```
// Isso é um comentário
```

VALORES PRIMITIVOS

String	"hello world"	'abacaxi'	`5 patinhos`			
Number	20	3.1415	-18	-9.04	Infinity	NaN
Boolean	true	false				
Object	{}					
Outros	null	undefined				

ARRAY



Pilha

Em um array, é possível utilizar funções próprias para manipular elementos em qualquer posição da lista. Porém, há situações (veremos exemplos mais adiante) onde é desejável mais controle sobre as operações que podem ser feitas na estrutura. Aí entra a implementação de estruturas de dados como a pilha (stack) e a fila (queue).

Fila

A fila tem uma estrutura semelhante à pilha, porém com uma diferença conceitual importante: o paradigma por trás da fila é o FIFO - First In, First Out, ou “o primeiro a entrar é o primeiro a sair”, em tradução livre.

Deque

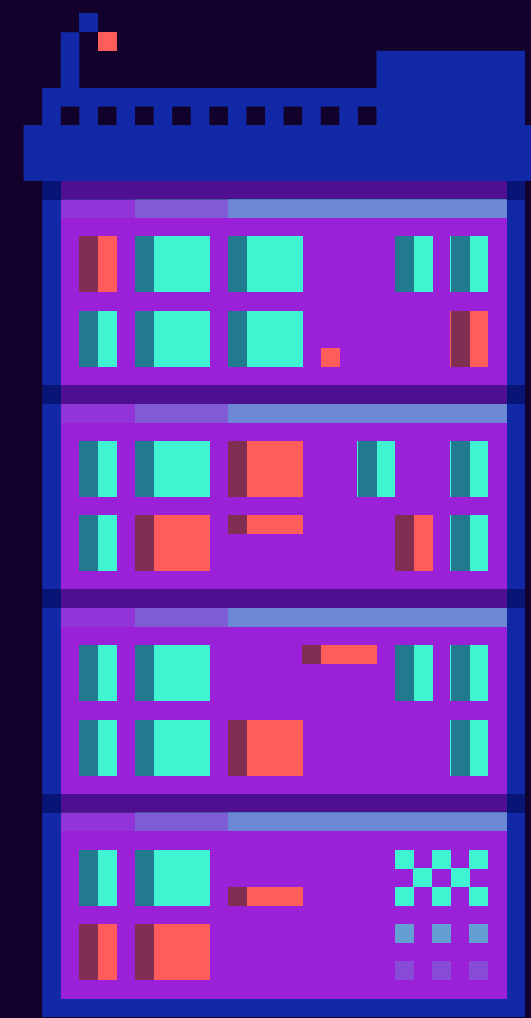
A estrutura de dados deque (abreviação de double-ended queue ou “fila de duas pontas”) é uma variação da fila que aceita inserção e remoção de elementos tanto do início quanto do final da fila.

OPERADORES

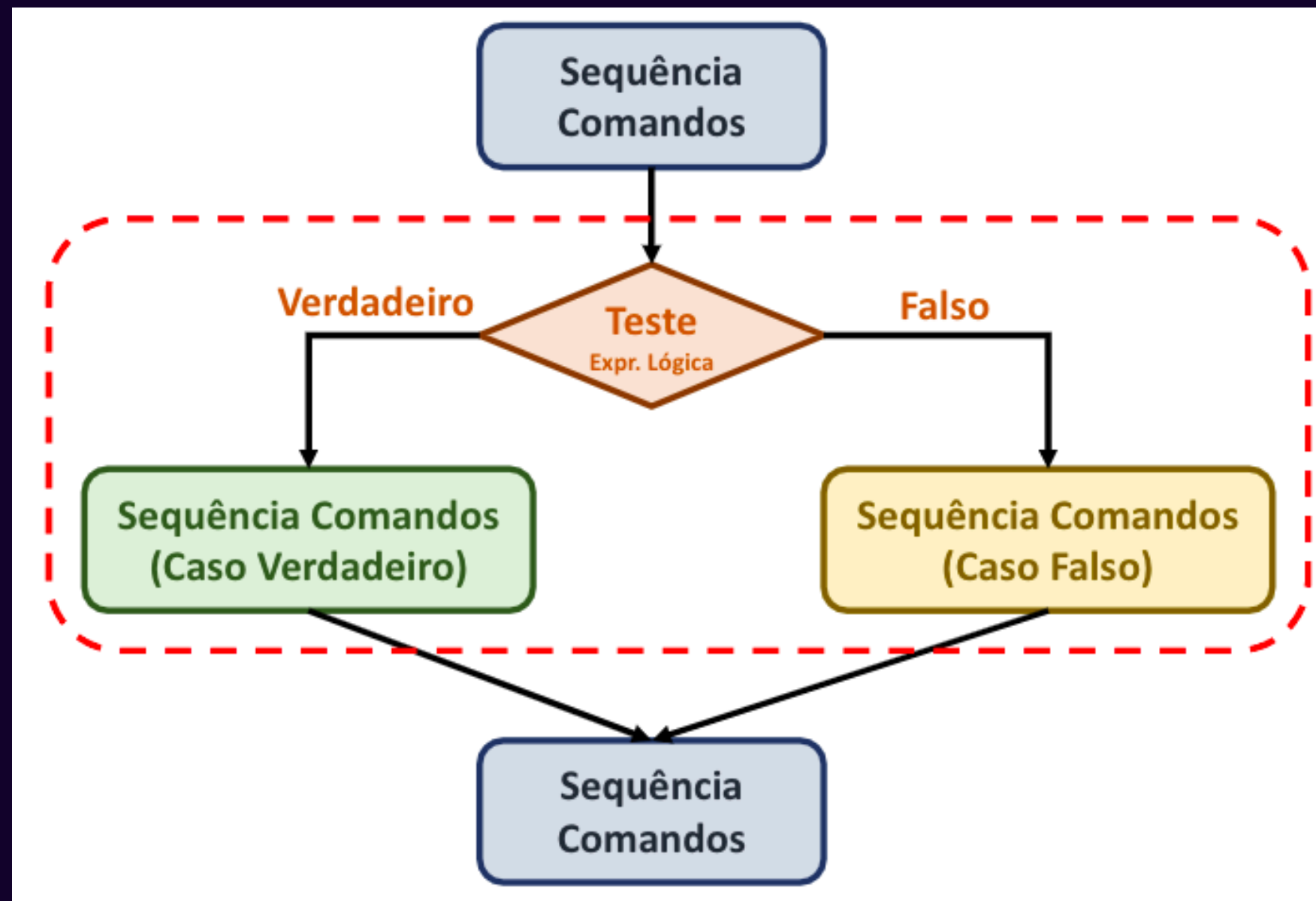
Um operator é um símbolo matemático que produz um resultado baseado em dois valores (ou variáveis). Na tabela a seguir, você pode ver alguns dos operadores mais simples, juntamente com alguns exemplos para experimentar no console JavaScript.

Operador	Explicação	Símbolo(s)	Exemplo
Adição	Adicione dois números ou combine duas strings.	+	<pre>6 + 9; 'Olá ' + 'mundo!';</pre>
Subtração, Multiplicação, Divisão	Eles fazem o que você espera que façam em matemática básica.	- , * , /	<pre>9 - 3; 8 * 2; // multiplicar em JS é um asterisco 9 / 3;</pre>
Atribuição	Como você já viu: isso atribui um valor a uma variável.	=	<pre>let myVariable = 'Bob';</pre>
Igualdade estrita	Isso executa um teste para ver se dois valores são iguais. Ele retorna um Resultado <code>true / false</code> (booleano).	===	<pre>let myVariable = 3; myVariable === 4;</pre>

Não, não é igual	Isso retorna o valor logicamente oposto do que precede. Acontece um <code>true</code> em um <code>false</code> , etc. Quando é usado ao lado do operador de igualdade, o operador de negação testa se dois os valores <i>não</i> são iguais.	!, !==	<p>Para "Not", a expressão básica é <code>true</code>, mas o a comparação retorna <code>false</code> porque nós a negamos:</p> <pre>let myVariable = 3; !(myVariable === 3);</pre> <p>"Não é igual" dá basicamente o mesmo resultado com diferentes sintaxe. Aqui estamos testando "é <code>myVariable</code> NÃO igual a 3". Isso retorna <code>false</code> porque <code>myVariable</code> É igual a 3:</p> <pre>let myVariable = 3; myVariable !== 3;</pre>
------------------	--	--------	--



ESTRUTURAS CONDICIONAIS



REPETIÇÕES

COM 'FOR'

As linguagens de programação são muito úteis para concluir rapidamente tarefas repetitivas, desde vários cálculos básicos até praticamente qualquer outra situação em que você tenha muitos itens semelhantes para manipular.

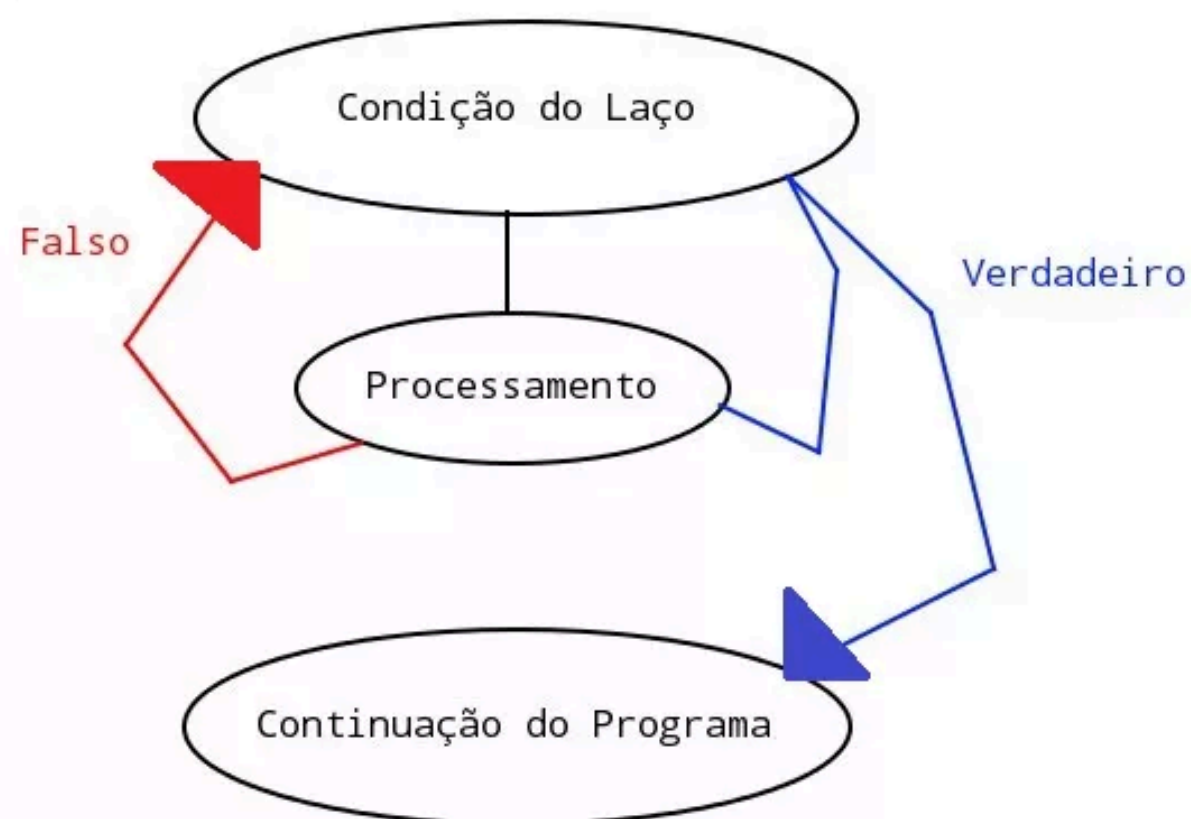


REPETIÇÕES

COM 'WHILE'

While funciona de maneira muito semelhante ao loop for, exceto que a variável inicializadora é definida antes do loop, e a expressão final é incluída dentro do loop após o código a ser executado - em vez de esses dois itens serem incluídos dentro dos parênteses. A condição de saída está incluída dentro dos parênteses, que são precedidos pela palavra-chave while e não por for.

While / Enquanto



FUNÇÕES

Functions são uma forma de empacotar a funcionalidade que você deseja reutilizar. É possível definir um corpo de código como uma função executada quando você chama o nome da função em seu código.

```
function () {  
  JS ? !  
}
```



MEUS CONTATOS:



27 99500-7495



<https://beacons.ai/prismatech>



producaoprismatech@gmail.com



Avenida Jerônimo Monteiro 145, Vitória





THANK
YOU