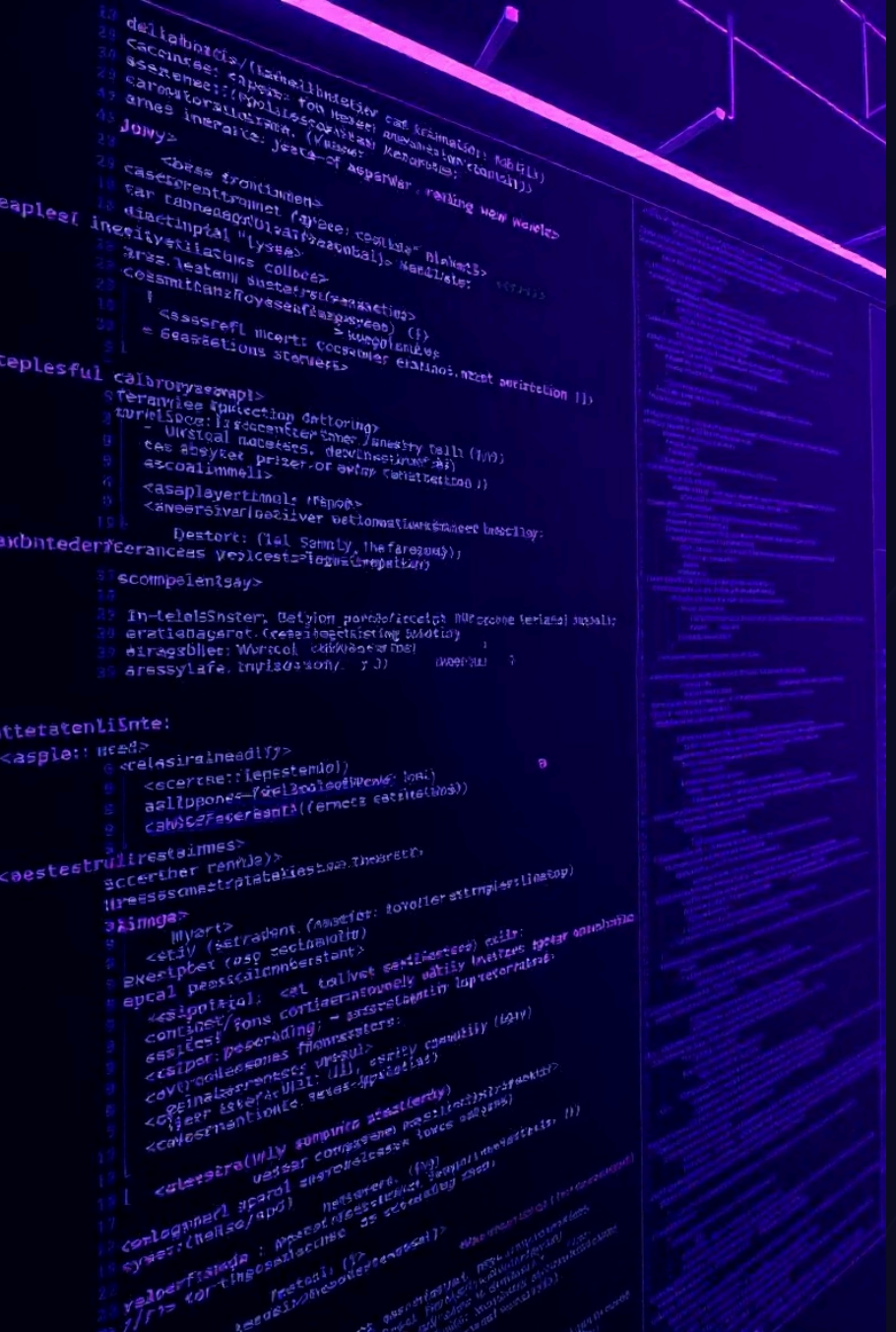


Monitoria Javascript

Por Que JavaScript é Tão Popular?

O JavaScript, carinhosamente conhecido como JS, é uma linguagem de programação interpretada de alto nível. Segundo a Pesquisa de Desenvolvedores do Stack Overflow de 2022, ele se mantém como a **linguagem mais popular no mundo** — e não é por acaso!

Sua principal característica é a execução direta nos navegadores, o que tornou a navegação na web dinâmica e interativa. Essa facilidade de ver o código funcionando sem grandes configurações de ambiente o torna perfeito para quem está **iniciando seus estudos em programação**.

Client-Side e Server-Side: Onde o JS Atua

Client-Side

No lado do cliente (navegador), o JavaScript é responsável por criar páginas web interativas, animações, validação de formulários e muito mais. É a magia que vemos acontecer diretamente na tela.

Server-Side

Com o Node.js, o JavaScript expandiu seu poder para o lado do servidor. Isso permite construir APIs, gerenciar bancos de dados e criar aplicativos completos, tudo com uma única linguagem.

Essa versatilidade é um dos pilares da popularidade do JavaScript, permitindo que desenvolvedores trabalhem em diversas frentes com um conjunto de habilidades unificado.

Variáveis: Armazenando Informações Essenciais

Variáveis são como caixas nomeadas onde armazenamos dados que serão utilizados e manipulados ao longo do nosso código. Compreender como declarar e gerenciar variáveis é fundamental.

- **var:** Variáveis declaradas com `var` podem ser redeclaradas e reatribuídas.
- **let e const:** Estas declaram variáveis locais com escopo de bloco, sendo ideais para evitar efeitos colaterais indesejados. `let` permite reatribuição, enquanto `const` não.

Tipos de Dados Primitivos

No JavaScript, os tipos de dados primitivos são a base para construir informações mais complexas.

String

Sequências de caracteres, declaradas com aspas simples ou duplas. Ex: `"Olá, mundo!"` ou `'JavaScript'`.

Number

Valores numéricos, sejam inteiros ou decimais. Ex: `10`, `3.14`.

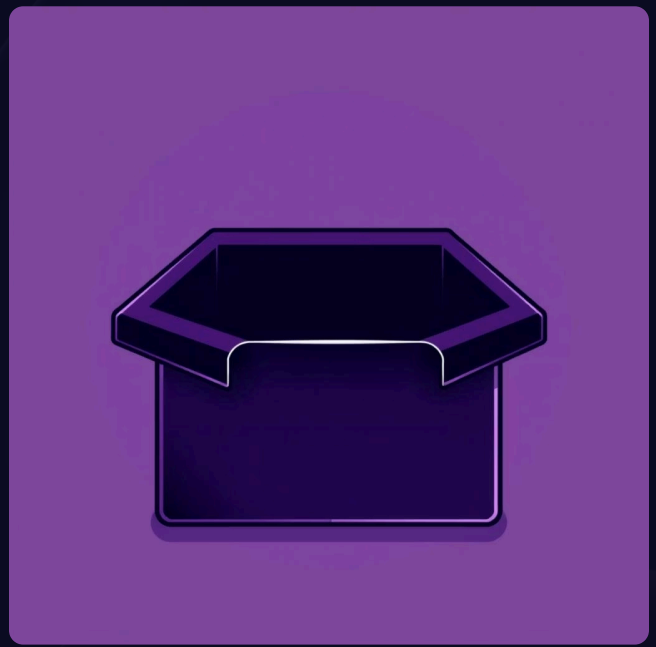
Boolean

Representa valores lógicos: `true` ou `false`. Ex: `isLogged = true`.

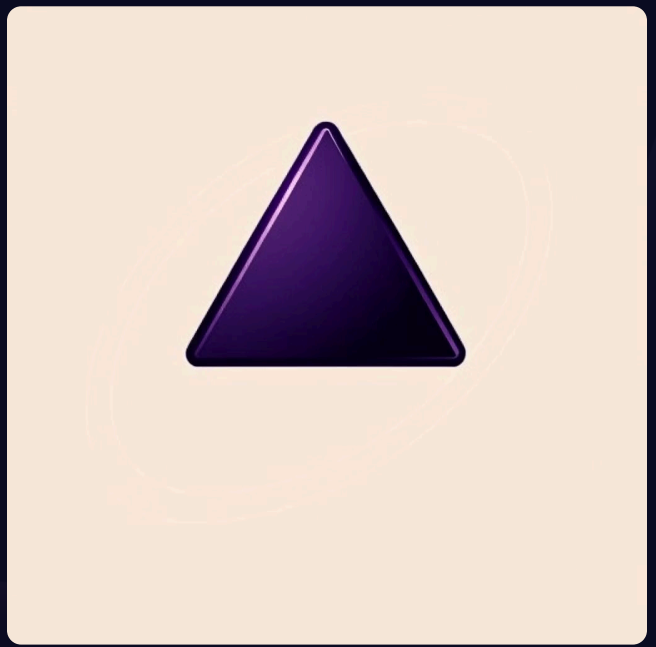
Os Primitivos Especiais: undefined, null e NaN

Undefined: Ocorre quando uma variável é declarada, mas ainda não recebeu um valor. É o valor padrão.

Null: Indica a ausência intencional de qualquer valor ou referência a um objeto. É um ponteiro de objeto vazio.



NaN (Not a Number): Um valor numérico especial que indica que o resultado de uma operação não é um número válido.



Operadores no JS

Os operadores são símbolos que instruem o JavaScript a executar uma ação.

Atribuição

`=` (atribui um valor)

Comparação

`==` (igualdade), `===` (igualdade estrita)

Lógicos

`||` (OU), `&&` (E)

Aritméticos

`+`, `-`, `*`, `/`, `%`

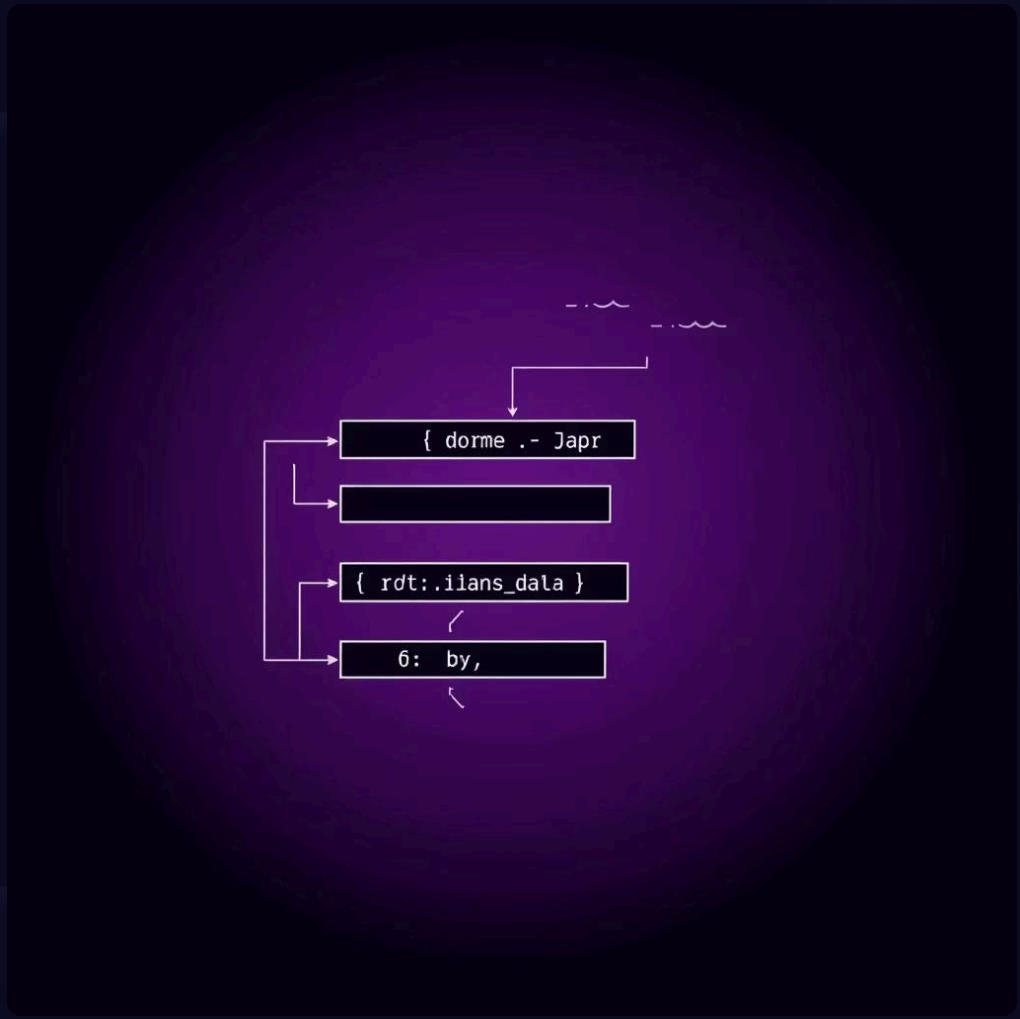
Além disso, temos operadores de **incremento (++)** e **decremento (--)** para modificar valores numéricos.

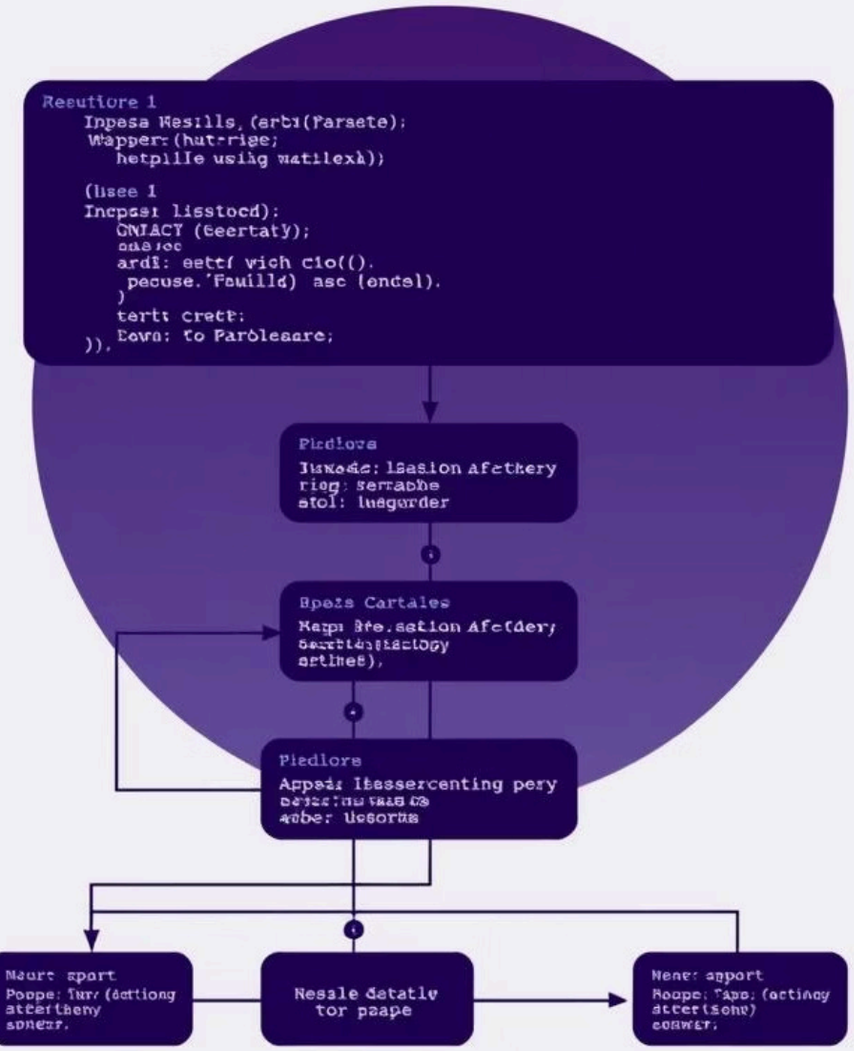
Objetos: Estruturando Dados Complexos

Em JavaScript, um objeto é uma coleção dinâmica de propriedades, onde cada propriedade é definida por um par de **chave (nome)** e **valor**. Eles são a espinha dorsal para organizar dados de forma estruturada.

Permitem agrupar dados e funcionalidades relacionadas, criando blocos lógicos em seu código.

Exemplo: `{ nome: "João", idade: 30 }`





Estruturas de Controle: Direcionando o Fluxo do Código

As estruturas de controle permitem que você tome decisões e repita ações em seu código, tornando-o dinâmico e inteligente.

Condicionais

`if/else`, `switch`: Executam blocos de código com base em condições.

Laços (Loops)

`for`, `while`, `do/while`: Repetem ações até que uma condição seja satisfeita.

Operador Ternário

Uma forma concisa de escrever condicionais simples: `condição ? valor1 : valor2`.

Funções: Reutilizando Código e Organizando Lógica

As funções são blocos de código reutilizáveis que executam uma tarefa específica. Elas são essenciais para organizar seu código, torná-lo mais modular e fácil de manter.

- **Declaração:** Definir uma função com parâmetros de entrada.
- **Chamada:** Executar a função em qualquer parte do código.
- **Retorno:** Funções podem retornar valores para serem usados em outras partes do programa.

Dominar funções é um passo crucial para escrever códigos eficientes e escaláveis em JavaScript.