Técnicas de Programação e Algoritmos / Lógica de Programação

Revisão de Conteúdo

Professora Debora B. Paulo

Revisão de Lógica de Programação com JavaScript (Modo Console)

Introdução aos conceitos básicos de lógica de programação.

Uso do JavaScript para implementar esses conceitos.

Foco em exemplos práticos no console.

Fonte: Hostgator

1. Variáveis

- As variáveis são usadas para armazenar dados.
- Em JavaScript, podemos declarar variáveis usando var, let ou const.

```
1
2   //Declaração de Variáveis e Atribuição de Valor
3
4   let nome = "João";
5   const idade = 30;
6   var cidade = "São Paulo";
7
8   // Saída de Dados:
9   console.log(nome);
10  console.log(idade);
11  console.log(cidade);
12
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\projetosIW1\bimestre3\semana1\aula1> node revisao.js
João
30
São Paulo
PS D:\projetosIW1\bimestre3\semana1\aula1>
```

1.1. Escopo de Variáveis

- Ao declarar as variáveis é importante observar as formas de declaração: locais ou globais

1.2. Variáveis Locais

- As variáveis locais são declaradas dentro de um procedimento ou função, ou associadas a um evento, como por exemplo, um botão no formulário.
- Importante saber, que estas variáveis não são acessíveis por outras partes do programa.
- Elas só poderão ser utilizadas dentro do escopo que foi declarada.

1.3. Variáveis Globais

- As variáveis globais devem ser declaradas em um local que possa ser conhecida por todos os componentes do programa.
- Geralmente são declaradas no início.
- Ela poderá ser acessada e qualquer lugar dentro do programa.
- Exemplo, um botão que realiza operações em diferentes partes dentro do programa, pois ele têm acesso às mesmas variáveis.

1.4. Quando usar as variáveis Globais ou Locais

- Normalmente usamos as variáveis **globais** quando um valor dever ser conhecido manipulado em várias partes do seu código.
- Já as variáveis **locais** tem uma função específica dentro de um bloco de comandos, por exemplo, dentro de um bloco de repetição.

2. Tipos de Dados

- São diferentes formatos e estruturas que podem ser utilizados para armazenar informações
- O valor atribuído a uma variável pode ser de qualquer tipo de dados.

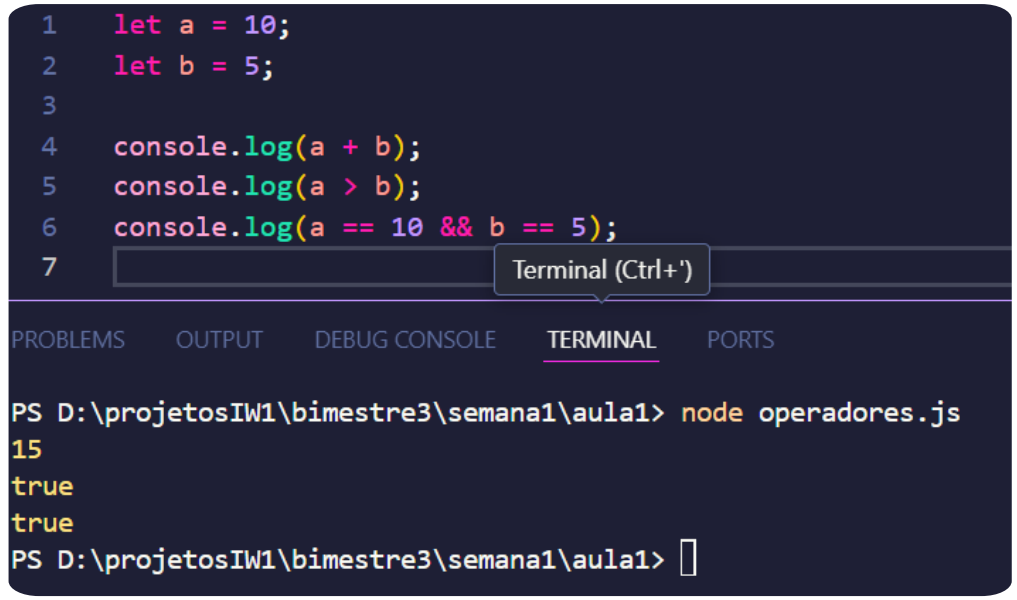
```
1   // Declaração de Variáveis
2   let texto = "Olá, mundo!"; //String
3   let numero = 42; // Inteiro
4   let decimal = 3.14; // Flutuante
5   let verdadeiro = true; // Booleano
6
7   // Saída de Dados:
8   console.log(texto);
9   console.log(numero);
10  console.log(decimal);
11  console.log(verdadeiro);
12
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\projetosIW1\bimestre3\semana1\aula1> node tiposDados.js
Olá, mundo!
42
3.14
true
PS D:\projetosIW1\bimestre3\semana1\aula1>
```

3. Operadores

- Aritméticos: +, -, *, /
- Comparação: ==, ===, !=, !==, >, <, >=, <=
- Lógicos: &&, ||, !



Igualdade: “==“

- O que faz: Compara dois valores para verificar se são iguais.
- Como funciona: Converte os valores para o mesmo tipo antes de comparar.
- Isso é chamado de "coerção de tipo".

Igualdade:

- Qual a diferença entre usar == ou === para comparar?

```
console.log(5 == "5"); // verdadeiro
```

- Neste exemplo, 5 (um número) e "5" (uma string) são considerados iguais porque == converte a string "5" para o número 5 antes de comparar.

Igualdade Estrita: “===“

- O que faz: Compara dois valores para verificar se são exatamente iguais.
- Como funciona: Não converte os valores para o mesmo tipo.
- Os valores devem ser do mesmo tipo e ter o mesmo valor. Exemplo:

Igualdade:

- Neste exemplo, 5 (um número) e "5" (uma string) não são considerados iguais porque === não converte os tipos. Eles são de tipos diferentes, então a comparação retorna false.

```
1   console.log(5 == "5"); // verdadeiro
2   console.log(5 === "5"); // falso
3
```

Concluindo:

- Use == quando quiser comparar valores, permitindo a conversão de tipos.
- Use === quando quiser comparar valores sem permitir a conversão de tipos, garantindo que os valores e tipos sejam exatamente os mesmos.

Para evitar erros e surpresas, é geralmente recomendado usar === para comparações, já que ele não faz conversão de tipo e pode ajudar a evitar bugs em seu código.

DÚVIDAS?