

JavaScript

variáveis

JS
JavaScript

parte 5

aprendendo Javascript

agenda

já conhecemos os tipos primitivos do JavaScript e agora vamos aprender um pouco mais sobre escopo das variáveis para usar os tipos nos nossos códigos.



como declaro uma variável em JS

preocupar com escopo

toda linguagem de programação, ou quase na sua totalidade tem uma forma sintática de declarar variáveis. O JavaScript não foge a regra e tem três formas de declarar suas variáveis.

1. ***var.***
2. ***let.***
3. ***const.***

as duas primeiras formas de declarar criam variáveis mutáveis, ou seja, é permitido que você as crie, atribua valores e durante todo o seu código é permitido alterar este valor até que em algum momento esta variável seja destruída da memória.

com o operador `const` você estará declarando uma constante que nada mais é que uma variável imutável. Ela não pode ter seu valor alterado durante o curso do seu programa.

há uma diferença importante entre usar o operador `var` e o `let` no que se refere ao escopo da visualização. Nós vamos falar mais sobre isto adiante.

Exemplos de declaração de uma variável.

```
// declarando variáveis com JavaScript
let nome = 'Alan';
var sobrenome = "Alencar";
const naturalDe = 'Montes Claros';
var altura = 1.85; // altura atual.
let moraEm = 'Belo Horizonte'; // cidade atual
```

constantes

nasce e morre com o mesmo valor

se por algum motivo, você tentar mudar o conteúdo de uma constante, o JavaScript vai lançar uma exceção informando que o identificador foi declarado como constante e portanto não pode ter seu conteúdo alterado. Vejamos o exemplo a seguir.



```
// declarando constantes com JS.  
const pais = 'Brazil';  
  
console.log('Meu pais é o', pais);  
console.log('Espera, escrevi o nome do meu pais com Z e é com S');  
pais = 'Brasil'; // irá lançar uma exceção, que aprenderemos tratar depois.  
console.log('Agora com o nome correto', pais);
```

ao executar este código JavaScript no terminal com o Node, será lançado uma exceção com a mensagem de que uma constante não pode ter seu valor alterado após sua criação.




```
TypeError: Assignment to constant variable.
```

var ou let

global ou escopo

todas as variáveis criadas em JavaScript com o operador `var` vai determinar escopo **global**.

mesmo a variável `novoSalario` sendo criada dentro do bloco `{ ... }` ela ainda poderá ser acessada fora dele.



```
// declarar com VAR e ter escopo global
var salario = 2319.85;
var fatorReajuste = 1.12; // 12% de aumento.

console.log('Salário sem reajuste', salario);
{
    var novoSalario = salario * fatorReajuste;
    console.log('Reajuste foi de', fatorReajuste - 1, '%');
}
console.log('Novo Salário é', novoSalario);
```


var ou let

global ou escopo

já as variáveis criadas em JavaScript com o operador `let` vão ter escopo **de bloco**. Ou seja, para dentro vão ser acessadas, mas o contrário não vai funcionar.

```
// declarar com LET e ter escopo global
let salario = 2319.85;
let fatorReajuste = 1.12; // 12% de aumento.

console.log('Salário sem reajuste', salario);
{
    let novoSalario = salario * fatorReajuste;
    console.log('Reajuste foi de', fatorReajuste - 1, '%');
}
console.log('Novo Salário é', novoSalario); // irá gerar ERRO.
// ReferenceError: Can't find variable: novoSalario
```

exibindo variáveis

como dar saída em conteúdo

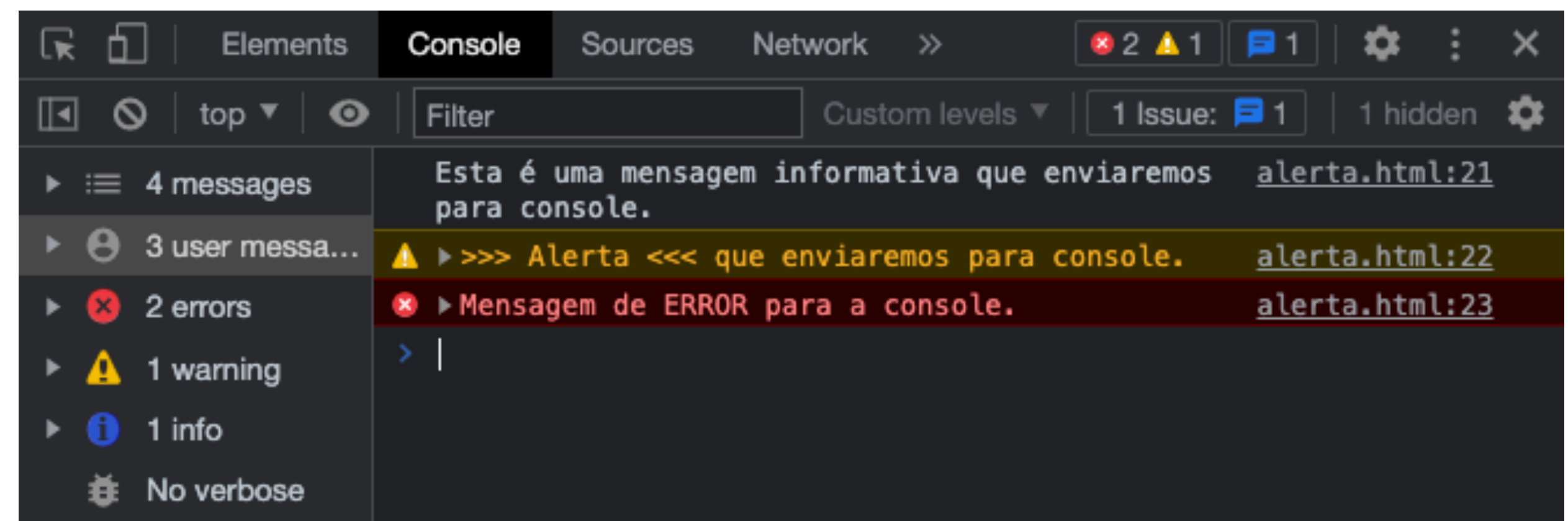
em JavaScript nós temos muitas formas de dar saída no conteúdo. Temos três saídas para console.

```
console.log( )  
console.warn( )  
console.error( )
```

Todas elas exibem o conteúdo no console do JavaScript do seu navegador, sendo uma para mensagens, outra para alertas e outra para erros.

```
/**  
 * @author Alan Alencar  
 * @description Como exibir alertas com JS na console.  
 */  
  
console.log('Esta é uma mensagem informativa que enviaremos para console.');
```

produz a saída no console javascript do browser.



hoisting

elevando declaração de VAR

o JavaScript tem uma particularidade interessante que poucas linguagens de programação são capazes de entregar.

you pode chamar uma variável antes mesmo dela ser declarada, desde que ela seja declarada com o operador `var`.

chamamos isto de ***hoisting***, mas já adianto que não é uma boa ideia.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains three lines of JavaScript code: a comment, a log statement, and a variable declaration.

```
// hoisting  
console.log(nome); // resulta: undefined  
var nome = 'Alan';
```


no código acima, `console.log` exibe a variável `nome`, porém ela só foi declarada após a execução do comando. a maioria das linguagens de programação apontaria um erro, porém JS tem este conceito de hoisting. cuidado porque sempre será `undefined`.

template string

formatar saída com mais clareza

o JavaScript trouxe uma forma muito interessante que outras linguagens de programação já implementava chamado de **Template Strings**. Nada mais é que uma forma de você escrever strings com uso de variáveis que serão automaticamente substituídas pelo seu conteúdo.

A regra é iniciar e fechar com **crase** ao invés de aspas (simples ou duplas)



```
// usando template string
let salario = 2319.85;

console.log('Salário é', salario);
console.log(`Salário é ${salario}`);
```

dentro da string entre crases você usa a sintaxe `${variavel}` que ao ecoar na saída do console vai ser substituído pelo conteúdo da variável ou constante. Muito mais simples e fácil de escrever no código.



VAMOS PRATICAR

programming

revendo a prática

princípio básico

1. vimos que JavaScript possui várias formas de dar saída na console do navegador.
2. aprendemos que **var** declara variáveis com escopo global e **let** variáveis de escopo de bloco.
3. **const** declara constantes que são imutáveis.
4. aprendemos o conceito de hoisting desde que declarado com **var**.
5. aprendemos o uso do template string.

