

{reprograma}

JavaScript I

O que é JavaScript?

De uma forma simples, JavaScript é uma linguagem de programação usada para desenvolver aplicações, sistemas e serviços de alta complexidade.

Com ela, você pode criar páginas web dinâmicas, animações, mapas interativos, gráficos em três dimensões, aplicativos para dispositivos móveis e games para plataformas portáteis.

O JavaScript atua como um complemento às linguagens HTML, CSS e PHP no momento em que o desenvolvedor vai construir uma página na internet.

https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/O_que_e_JavaScript

Utilizando JavaScript no projeto

1. Inserindo códigos na própria página (*inline*):

Cria-se uma *tag* `<script>`, informando que o valor do atributo 'type' é 'text/javascript', então, coloca-se o código JavaScript dentro dessa *tag*.

Exemplo:

```
1.<script type="text/javascript">  
2.  alert( 'Olá mundo!' );  
3.</script>
```

2. Relacionando um arquivo externo na página:

Essa forma é bem parecida com a inserção de códigos JavaScript *inline*, a maior diferença é que não coloca-se o código JavaScript dentro da *tag*, visto que esse código estará em um arquivo externo. Assim, simplesmente é preenchido o atributo 'src' da *tag* `<script>` com o caminho para o arquivo em questão.

Exemplo 1 - adicionando um JavaScript do nosso projeto:

```
1.<script type="text/javascript" src="js/meu-arquivo.js"></script>
```

Valores Primitivos

Todos os tipos, com a exceção de objetos, definem valores imutáveis (valores que são incapazes de mudar).

Boolean é um tipo de dado que só pode assumir os valores **true** e **false**, que em português é o equivalente a verdadeiro e falso respectivamente.

Em Javascript, Boolean é comumente utilizado como uma função em variáveis, condições, objetos dentre outros quando temos de verificar se um retorno foi verdadeiro ou não.

O tipo String em JavaScript é usado para representar textos.

O tipo Number em JavaScript é usado para representar dados numéricos.



Tipos Primitivos

```
//atribuindo as variáveis  
let meuNumero = 10;  
let minhaString = "Olá, mundo!";  
let meuBooleano = true;
```

```
//imprimindo na tela as variáveis e seus respectivos tipos  
console.log("Meu número é: ", meuNumero);  
console.log("O tipo do número é: ", typeof meuNumero);  
console.log("O tipo da minha string é: ", typeof minhaString);  
console.log("O tipo do meu booleano é: ", typeof meuBooleano);
```


Variáveis

Variáveis são usadas como nomes simbólicos para valores em sua aplicação. Os nomes das variáveis, chamadas *identificadores*, de acordo com certas regras.

Um identificador JavaScript deve começar com uma letra, sublinhado (`_`), ou cifrão (`$`); caracteres subsequentes podem também ser dígitos (0-9). Como o JavaScript é sensível a maiúsculas, as letras **A** algumas exceções de "A" até "Z" (maiúsculas) e os caracteres de "a" até "z" (minúsculas).

JavaScript não é uma linguagem tipada, e podemos declarar variáveis de duas maneiras:

```
const nome = "Bruna"; //para variáveis que não serão alteradas futuramente.
```

```
let idade = 26; //para variáveis que podem ser alteradas
```

É possível também, encontrarmos variáveis declaradas com var, mas essa prática caiu em desuso e podemos entender mais através do link: <https://www.alura.com.br/artigos/entenda-diferenca-entre-var-let-e-const-no-javascript>

Palavras Reservadas

abstract	enum	<u>interface</u>	throw
boolean	export	long	throws
break	extends	native	transient
byte	false	new	true
case	<u>final</u>	null	try
catch	finally	package	typeof
char	float	private	var
class	for	protected	volatile
const	function	public	void
<u>continue</u>	goto	return	while
debugger	if	short	with
default	implements	static	
<u>delete</u>	import	<u>super</u>	
do	in	switch	
double	instanceof	synchronized	
else	int	this	

Comentários

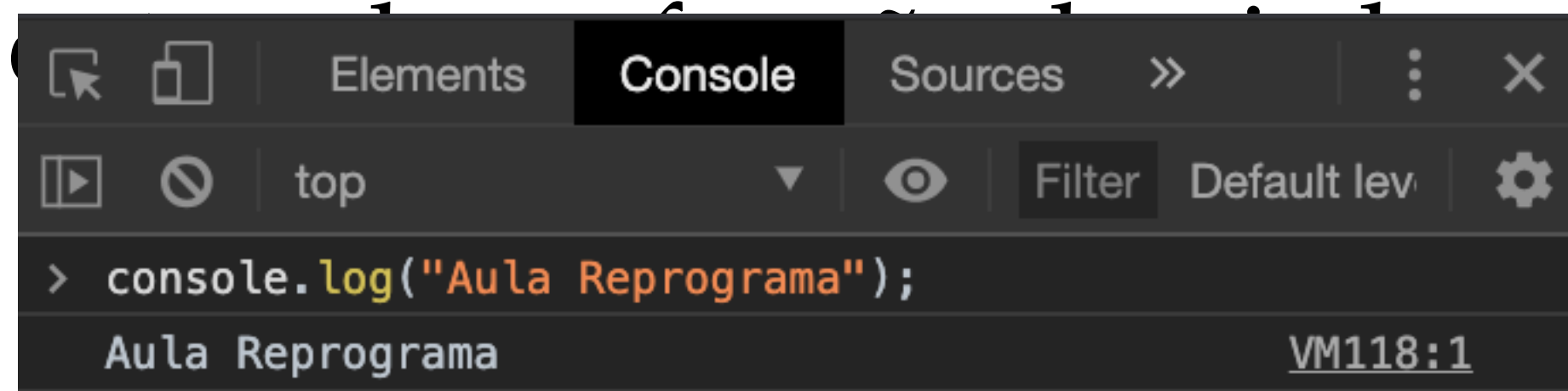
- // Comentário

De bloco

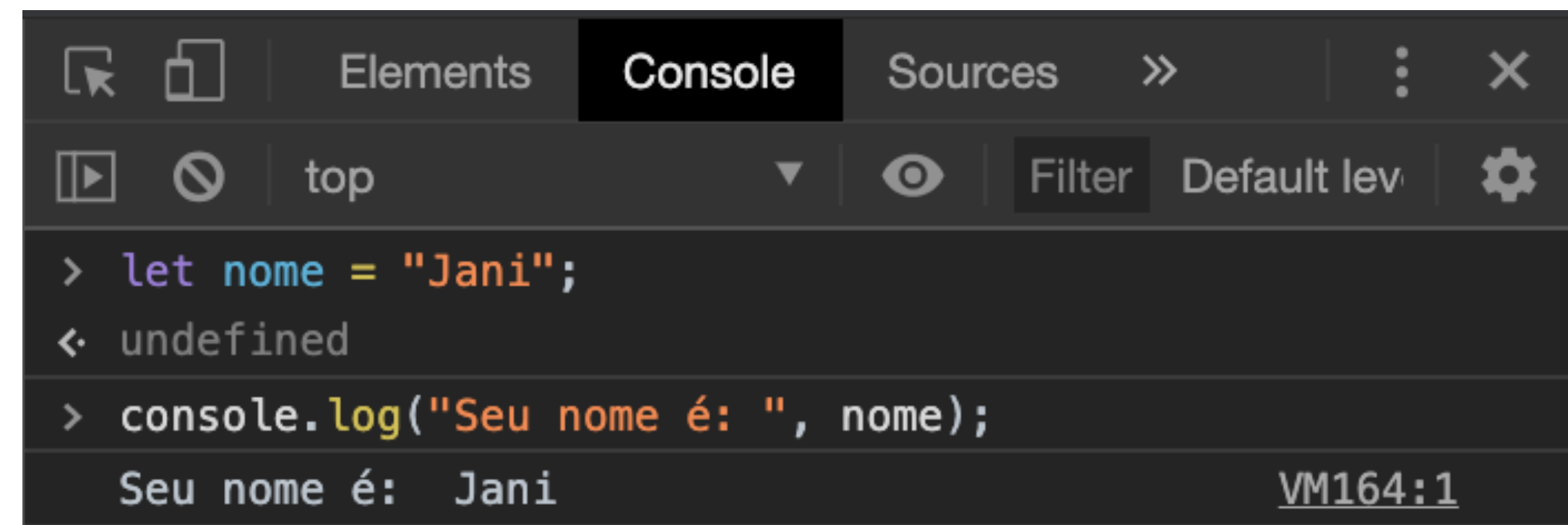
- /* Comentário de 1 ou n linhas */

Console.log()

O `console.log()` é uma ferramenta que nos permite exibir mensagens ou pode ser utilizado como forma de realizar debug na aplicação, exibindo o valor que está sendo recebido na variável ou até mesmo se a mesma está



A screenshot of a web browser's developer console. The 'Console' tab is selected. It shows a single log entry: `> console.log("Aula Reprograma");` with the output `Aula Reprograma` and the source `VM118:1`.



A screenshot of a web browser's developer console. The 'Console' tab is selected. It shows two log entries: `> let nome = "Jani";` with the output `< undefined`, and `> console.log("Seu nome é: ", nome);` with the output `Seu nome é: Jani` and the source `VM164:1`.

Para maiores informações e outros tipos de console: <https://developer.mozilla.org/pt-BR/docs/Web/API/Console>
<https://imasters.com.br/desenvolvimento/o-poder-do-console-do-navegador>

Operadores Aritméticos

Operador	Descrição
Divisão	x / y
Adição	$x + y$
Subtração	$x - y$
Multiplicação	$x * y$

Operador	Descrição	Exemplo
Módulo (%)	Operador binário. Retorna o inteiro restante da divisão dos dois operandos.	12 % 5 retorna 2.
Incremento (++)	Operador unário. Adiciona um ao seu operando. Se usado como operador prefixado (++x), retorna o valor de seu operando após a adição. Se usado como operador pósfixado (x++), retorna o valor de seu operando antes da adição.	Se x é 3, então ++x define x como 4 e retorna 4, enquanto x++ retorna 3 e, somente então, define x como 4.
Decremento (--)	Operador unário. Subtrai um de seu operando. O valor de retorno é análogo àquele do operador de incremento.	Se x é 3, então --x define x como 2 e retorna 2, enquanto x-- retorna 3 e, somente então, define x como 2.
Negação (-)	Operador unário. Retorna a negação de seu operando.	Se x é 3, então -x retorna -3.
Adição (+)	Operador unário. Tenta converter o operando em um número, sempre que possível.	+ "3" retorna 3. +true retorna 1.

Operadores de atribuição

Nome	Operador encurtado	Significado
Atribuição	<code>x = y</code>	<code>x = y</code>
Atribuição de adição	<code>x += y</code>	<code>x = x + y</code>
Atribuição de subtração	<code>x -= y</code>	<code>x = x - y</code>
Atribuição de multiplicação	<code>x *= y</code>	<code>x = x * y</code>
Atribuição de divisão	<code>x /= y</code>	<code>x = x / y</code>
Atribuição de resto	<code>x %= y</code>	<code>x = x % y</code>

Operadores de comparação

Operador	Descrição	Exemplos que retornam verdadeiro
Igual (==)	Retorna verdadeiro caso os operandos sejam iguais.	<pre>3 == var1 "3" == var1 3 == '3'</pre>
Não igual (!=)	Retorna verdadeiro caso os operandos não sejam iguais.	<pre>var1 != 4 var2 != "3"</pre>
Estritamente igual (===)	Retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo.	<pre>3 === var1</pre>
Estritamente não igual (!==)	Retorna verdadeiro caso os operandos não sejam iguais e/ou não sejam do mesmo tipo.	<pre>var1 !== "3" 3 !== '3'</pre>
Maior que (>)	Retorna verdadeiro caso o operando da esquerda seja maior que o da direita.	<pre>var2 > var1 "12" > 2</pre>
Maior que ou igual (>=)	Retorna verdadeiro caso o operando da esquerda seja maior ou igual ao da direita.	<pre>var2 >= var1 var1 >= 3</pre>

Operadores Lógicos

Operador	Utilização	Descrição
AND lógico (& &)	<code>expr1</code> <code>&&</code> <code>expr2</code>	(E lógico) - Retorna <code>expr1</code> caso possa ser convertido para falso; senão, retorna <code>expr2</code> . Assim, quando utilizado com valores booleanos, <code>&&</code> retorna verdadeiro caso ambos operandos sejam verdadeiros; caso contrário, retorna falso.
OU lógico ()	<code>expr1</code> <code> </code> <code>expr2</code>	(OU lógico) - Retorna <code>expr1</code> caso possa ser convertido para verdadeiro; senão, retorna <code>expr2</code> . Assim, quando utilizado com valores booleanos, <code> </code> retorna verdadeiro caso ambos os operandos sejam verdadeiro; se ambos forem falsos, retorna falso.
NOT lógico (!)	<code>!expr</code>	(Negação lógica) Retorna falso caso o único operando possa ser convertido para verdadeiro; senão, retorna verdadeiro.

Para maiores informações: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_Operators#operadores_logicos

Estruturas de Decisão

If (se) estiver tudo bem eu vou pra aula,

Else (senão) eu assisto de casa.

Estruturas de Decisão

- ▶ **Decisão Simples:**

```
if <(condição)>{  
    /*Instruções para condição verdadeira*/  
}
```

- ▶ **Decisão Composta:**

```
if <(condição)>{  
    /*Instruções para condição verdadeira*/  
else{  
    /*Instruções para condição falsa*/  
}
```

Operador Ternário

O operador condicional ternário é um atalho para o condicional if.

condition ? expr1 : expr2

- condition é uma expressão que é avaliada como true ou false.
- expr1, expr2 são expressões com valores de qualquer tipo.

O operador condicional é o único operador ternário de JavaScript.

Exemplo

```
resultado = (a > b) ? "a é maior que b" : "b é maior que a";
```

O código acima é equivalente ao de baixo:

```
if (a > b) {  
    resultado = "a é maior que b";  
} else {  
    resultado = "b é maior que a";  
}
```

Switch

```
switch (variavel){  
    case valor 1:  
        //comandos 1  
        break;  
    case valor 2:  
        //comandos 2  
        break;  
    case valor 3:  
        //comandos 3  
        break;  
    default:  
        //comandos alternativos  
}
```


Função/Function

Utilizada para criar um função.

```
//Função sem passagem de parâmetros  
function dividir(){  
    alert(6 /2 );  
}  
  
//Função com passagem de parâmetros  
function multiplicar(num1, num2){  
    alert(num1 * num2);  
}  
multiplicar(6,2)
```

return

Qual a diferença de uma Função e um Método?

```
//Função sem passagem de parâmetros  
function subtrair(num1, num2){  
    return num1 - num2;  
}  
  
resultado = subtrair(6, 2);  
alert(resultado);
```

Métodos sempre retornam valores.

Estruturas de Repetição

while

```
while (i<3){  
    //comandos  
    i++  
}
```

do while

```
do{  
    //comandos  
    i++  
} while (i<3);
```