

Desenvolvedores de softwares (DEVs)

Professor: Helder Morais
helder.morais@posgrad.ufsc.br

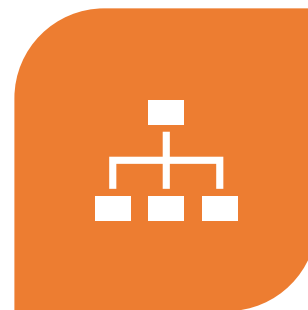




JavaScript (Aula 3-19/08/2021)

- Variáveis e constantes; e
- Funções.



Visual Studio Code: Teclas de atalho



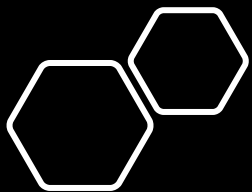
SHIFT + ALT + F – ORGANIZA O
CÓDIGO AUTOMATICAMENTE;



CTRL + ALT + N – EXECUTA O
SCRIPT NO CONSOLE;



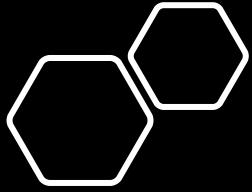
CTRL + SHIFT + P – OPÇÕES DE
CONFIGURAÇÃO;



Conversão de tipos de dados

JavaScript é uma linguagem dinamicamente tipada.

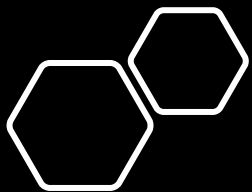
```
45  
46   var resultadoAvaliaca = 10;  
47  
48   resultadoAvaliaca = "Aprovado(a)";  
49  
50   resultadoAvaliaca = "Aprovado(a): " + 10;  
51  
52   console.log (resultadoAvaliaca);  
53
```



Conversão de strings para números

```
55     var resultadoAvaliaca = "9.52";  
56  
57     var convertStringParaInteiro = parseInt(resultadoAvaliaca);  
58  
59     var convertStringParaFloat = parseFloat(resultadoAvaliaca);  
60  
61     console.log (convertStringParaInteiro);  
62     console.log (convertStringParaFloat);  
63
```

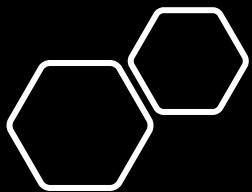
```
64  
65     var conversaoPorOperador = (+ "4.0") + (+ "10.0");  
66     console.log(conversaoPorOperador);  
67  
68
```



Literais

Utilizado para representar valores em JavaScript.

- Array literal
- Literais boolean
- Literais de ponto flutuante
- Inteiros
- Objeto literal
- String literal



Literais

Utilizado para representar valores em JavaScript.

- **Array literal**

```
68  
69   var alunos = ["Augusto", "Vicente", "Lucas"];  
70   console.log(alunos);  
71   console.log(alunos[0]);  
72
```

- **Literais boolean**

O tipo Boolean tem dois valores literal: true e false.

- **Literais de ponto flutuante**

3.1415926

-.123456789

-3.1E+12

.1e-23

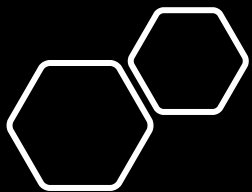
- **Inteiros**

0, 117 and -345 (decimal, base 10)

015, 0001 and -077 (octal, base 8)

0x1123, 0x00111 and -0xF1A7 (hexadecimal, "hex" or base 16)

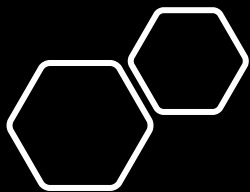
0b11, 0b0011 and -0b11 (binário, base 2)



Literais

Objeto literal: é uma lista de zero ou mais pares de nomes de propriedades e valores associados de um objeto, colocado entre chaves ({}).

```
74 var vendas = "Toyota";
75
76 function tipoCarro(nome) {
77     if (nome == "Fiat") {
78         return nome;
79     } else {
80         return "Desculpa, não vendemos carros " + nome + ".";
81     }
82 }
83
84 var carro = { meuCarro: "Punto", getCarro: tipoCarro("Fiat"), especial: vendas };
85
86 console.log(carro.meuCarro); // Punto
87 console.log(carro.getCarro); // Fiat
88 console.log(carro.especial); // Toyota
89
```

Literais

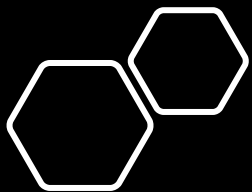
String literal: são zero ou mais caracteres dispostos em aspas duplas (") ou aspas simples (').

- **"curso"**
- **"101010"**
- **"um linha \n outra linha"**



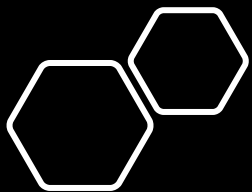
Operadores

Aritméticos, lógicos, de atribuição, de comparação, de tipo e Bitwise do Bit.



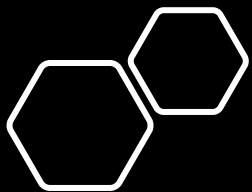
Operadores Aritméticos

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto de divisão
++	Incremento
--	Decremento



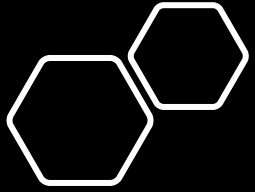
Operadores de atribuição

Operador	Exemplo	Igual a
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$



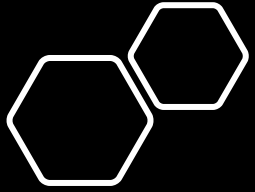
Operadores de comparação

Operador	Descrição
==	Igual a
===	Valor igual e tipo igual
!=	Não é igual
!==	Não é igual ao valor ou não é igual ao tipo
>	Maior que
<	Menor que
>=	Melhor que ou igual a
<=	Menos que ou igual a
?	Operador ternário



Operadores lógicos

Operador	Descrição
&&	lógico e
	lógico ou
!	lógico não



Operadores de tipo

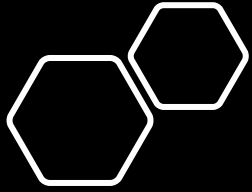
Operador	Descrição
typeof	Retorna o tipo de uma variável.
instanceof	Retorna verdadeiro se um objeto é uma instância de um tipo de objeto.

Operadores Bitwise do Bit

Operador	Descrição	Exemplo	Igual a	Resultado	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5 1	0101 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2
>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

Caracteres especiais no JavaScript

Caracter	Descrição
\0	Byte nulo
\b	Backspace
\f	Alimentador de formulário
\n	Nova linha
\r	Retorno do carro
\t	Tabulação
\v	Tabulação vertical
\'	Apóstrofo ou aspas simples
\"	Aspas dupla
\\	Caractere de barra invertida
\XXX	Caractere com a codificação Latin-1 especificada por três dígitos octal XXX entre 0 e 377. Por exemplo, \251 é sequência octal para o símbolo de direitos autorais.
\xxx	Caractere com a codificação Latin-1 especificada por dois dígitos hexadecimal XX entre 00 e FF. Por exemplo, \xA9 é a sequência hexadecimal para o símbolo de direitos autorais.
\uXXXX	Caractere Unicode especificado por quatro dígitos hexadecimal XXXX. Por exemplo, \u00A9 é a sequência Unicode para o símbolo de direitos autorais. Veja <u>sequências de escape Unicode</u> .



Diferença entre indefinido e nulo

```
Elements Console Sources Network
top [Filter]
> typeof undefined
< "undefined"
> typeof null
< "object"
> null === undefined
< false
> null == undefined
< true
```



Funções

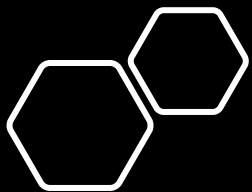
Literais

Anônimas

Autoexecutáveis

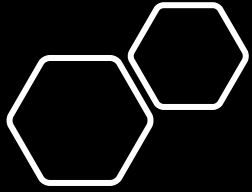
Funções pré-definidas

Funções	Descrição
<code>eval()</code>	O método <code>eval()</code> avalia código JavaScript representado como uma string.
<code>uneval()</code>	O método <code>uneval()</code> cria uma representação de string do código-fonte de um <code>Object</code> .
<code>isFinite()</code>	A função global <code>isFinite()</code> determina se o valor passado é um número finito. Se necessário, o parâmetro é primeiro convertido para um número.
<code>isNaN()</code>	A função <code>isNaN()</code> determina se um valor é NaN ou não. Nota: coerção dentro da função <code>isNaN</code> tem regras interessantes; você pode, alternativamente, querer usar <code>Number.isNaN()</code> , como definido no ECMAScript 6, ou você pode usar <code>typeof</code> para determinar se o valor não é um número.
<code>decodeURI()</code>	A função <code>decodeURI()</code> decodifica uma Uniform Resource Identifier (URI) criada anteriormente por <code>encodeURIComponent</code> ou por uma rotina similar.
<code>decodeURIComponent()</code>	O método <code>decodeURIComponent()</code> decodifica um componente Uniform Resource Identifier (URI) criado anteriormente por <code>encodeURIComponent</code> ou por uma rotina similar.
<code>encodeURIComponent()</code>	O método <code>encodeURIComponent()</code> codifica um Uniform Resource Identifier (URI), substituindo cada ocorrência de determinados caracteres por um, dois, três, ou quatro sequências de escape que representa a codificação UTF-8 do caractere (só serão quatro sequências de escape para caracteres compostos de dois caracteres "substitutos").



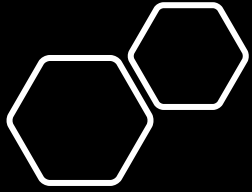
Função literal

```
//tipos de variáveis  
var nomeAluno = "Augusto";  
const universidade = "Universidade Federal de Santa Catarina";  
function media(n1, n2) {  
    let media = (n1 + n2) / 2;  
    console.log(media);  
}  
  
var nota = media(8, 8);  
nota;
```



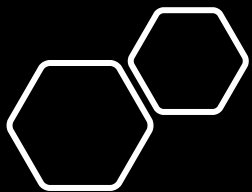
Funções anônimas

```
92 //função anônima
93 var valorMedia = function(n1, n2) {
94     let media = (n1 + n2) / 2;
95     console.log(media);
96 }
97
98 valorMedia(9,10);
99
```



Funções:
auto invocáveis
ou
autoexecutáveis

```
102 //funções autoexecutáveis.  
103 (function () {  
104     console.log(8 + 8);  
105 }  
106 )();
```



Desafio: calculadora de IMC

Muito abaixo do peso	16 a 16,9 kg/m ²
Abaixo do peso	17 a 18,4 kg/m ²
Peso normal	18,5 a 24,9 kg/m²
Acima do peso	25 a 29,9 kg/m ²
Obesidade Grau I	30 a 34,9 kg/m ²
Obesidade Grau II	35 a 40 kg/m ²
Obesidade Grau III	maior que 40 kg/m ²

Referências bibliográficas

- DUARTE, Nuno Filipe Brandão. Frameworks e Bibliotecas JavaScript. 2015. Disponível em: <recipp.ipp.pt/bitstream/10400.22/8223/1/DM_NunoDuarte_2015_MEI.pdf>. Acesso em: 9 abr. 2019.