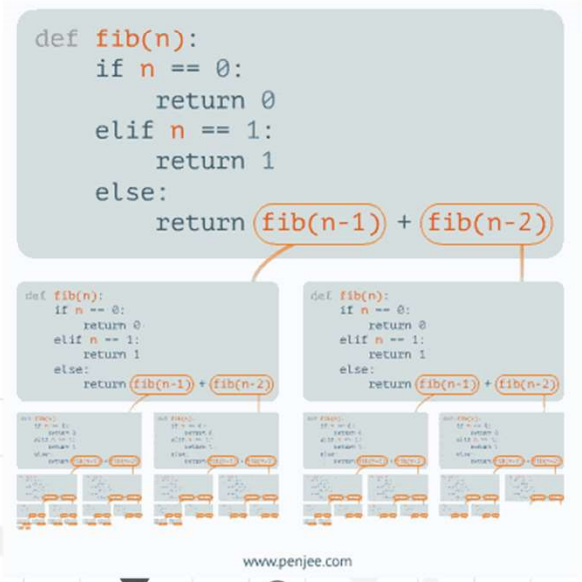
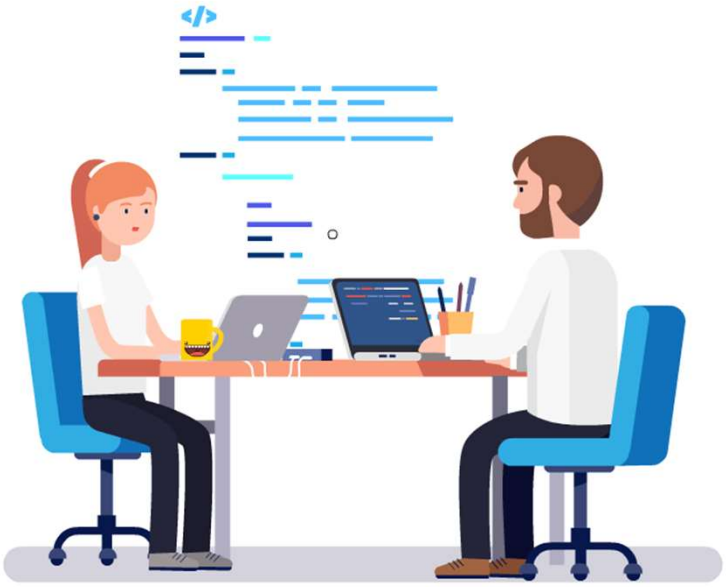
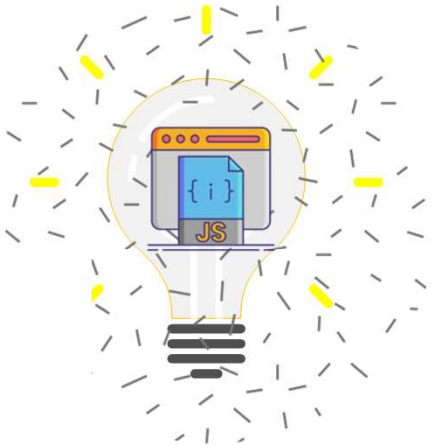
JAVASCRIPT – AULA 02







AGENDA

Parte I - JavaScript Básica

Parte II. Estrutura léxica

Parte III. Tipos, valores e variáveis

Parte IV. Expressões e operadores

Parte V. Instruções

Parte VI. Objetos

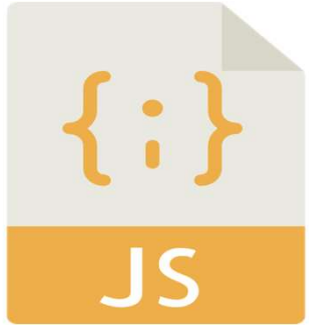
Parte VII. Arrays

Parte VIII. Funções

Parte IX. Classes e módulos

Parte X. JavaScript do lado do servidor

Parte XI. APIs de HTML5



ASSIM COMO QUALQUER LINGUAGEM, JAVASCRIPT RESERVA CERTOS IDENTIFICADORES PARA USO PRÓPRIO. ESSAS "PALAVRAS RESERVADAS" NÃO PODEM SER USADAS COMO IDENTIFICADORES NORMAIS. ELAS ESTÃO LISTADAS A SEGUIR

JAVASCRIPT

Assim como qualquer linguagem, JavaScript reserva certos identificadores para uso próprio.

Essas "palavras reservadas" não podem ser usadas como identificadores normais. Elas estão listadas a seguir.

TIPOS DE DADOS

Os tipos de JavaScript podem ser divididos em tipos primitivos e tipos de objeto.

E podem ser divididos em tipos com métodos e tipos sem métodos. Também podem ser classificados como tipos mutáveis e imutáveis. Um valor de um tipo mutável pode mudar.

Objetos e arrays são mutáveis: um programa JavaScript pode alterar os valores de propriedades do objeto e de elementos de arrays.

Números, booleanos, null e undefined são imutáveis – nem mesmo faria sentido falar sobre alterar o valor de um número, por exemplo. As strings podem ser consideradas arrays de caracteres, sendo que se poderia esperar que fossem mutáveis. No entanto, em JavaScript as strings são imutáveis: você pode acessar o texto de determinado índice de uma string, mas JavaScript não fornece uma maneira

NÚMEROS

Ao contrário de muitas linguagens, JavaScript não faz distinção entre valores inteiros e valores em ponto flutuante. Todos os números em JavaScript são representados como valores em ponto flutuante. JavaScript representa números usando o formato de ponto flutuante de 64 bits definido pelo padrão IEEE 7541 , isso significa que pode representar números tão grandes quanto ±1,7976931348623157 × 10308 e tão pequenos quanto ±5 × 10−324.

LITERAIS INTEIROS

Em um programa JavaScript, um inteiro de base 10 é escrito como uma sequência de dígitos.

Por exemplo:

0

3

10000000

```
<body>

    <h1> Imprimindo tipos de dados (números) </h1>

    <script>

        /* JavaScript têm muita precisão e
        podem se aproximar bastante de 0.1
        */
var a = 0
var b= 3
var c = 1000
var d=0xff
var e = 0xCAFE911
```

LITERAIS EM PONTO FLUTUANTE

Os literais em ponto flutuante podem ter um ponto decimal; eles usam a sintaxe tradicional dos números reais. Um valor real é representado como a parte inteira do número, seguida de um ponto decimal e a parte fracionária do número

3.14

2345.789

.3333333333333333

6.02e23 // 6.02 × 1023

1.4738223E-32 // 1.4738223 × 10−32

```
<body>

    <h1> Imprimindo tipos de dados (flutuantes) </h1>

    <script>

        /* JavaScript têm muita precisão e
        podem se aproximar bastante de 0.1
        */
var a = 3.14
var b= 2345.789
var c = .3333333333333333
var d=6.02e23 // 6.02 × 1023
var e = 1.4738223E-32 // 1.4738223 × 10-32
```

ARITMÉTICA COM JAVASCRIPT

Os programas JavaScript trabalham com números usando os operadores aritméticos fornecidos pela linguagem. Isso inclui + para adição, – para subtração, * para multiplicação, / para divisão e % para módulo (resto da divisão).

```
Math.pow(2,53)          // => 9007199254740992: 2 elevado à potência 53
Math.round(.6)          // => 1.0: arredonda para o inteiro mais próximo
Math.ceil(.6)           // => 1.0: arredonda para cima para um inteiro
Math.floor(.6)          // => 0.0: arredonda para baixo para um inteiro
Math.abs(-5)            // => 5: valor absoluto
Math.max(x,y,z)         // Retorna o maior argumento
Math.min(x,y,z)         // Retorna o menor argumento
Math.random()           // Número pseudoaleatório x, onde 0 <= x < 1.0
Math.PI                 // π: circunferência de um circulo / diâmetro
Math.E                  // e: A base do logaritmo natural
Math.sqrt(3)            // A raiz quadrada de 3
Math.pow(3, 1/3)        // A raiz cúbica de 3
Math.sin(0)             // Trigonometria: também Math.cos, Math.atan, etc.
```

```
<body>

    <h1> Imprimindo tipos de dados (Aritmética em JavaScript) </h1>

    <script>

        /* JavaScript têm muita precisão e
        podem se aproximar bastante de 0.1
        */
       var x=10
       var y=8
       var z=2

        document.write (Math.pow(2,53)) // => 9007199254740992: 2 elevado à potência 53
        document.write("<br>");
        document.write (Math.round(.6)) // => 1.0: arredonda para o inteiro mais próximo
        document.write("<br>");
        document.write (Math.ceil(.6) )// => 1.0: arredonda para cima para um inteiro
        document.write("<br>");
        document.write (Math.floor(.6)) // => 0.0: arredonda para baixo para um inteiro
        document.write("<br>");
```

DATA E HORA

JavaScript básico inclui uma construtora Date() para criar objetos que representam datas e horas.

Esses objetos Date têm métodos que fornecem uma API para cálculos simples de data. Os objetos

Date não são um tipo fundamental como os número

```
<body>

    <h1> Imprimindo tipos de dados (Tipo data) </h1>

    <script>

        /* JavaScript têm muita precisão e
        podem se aproximar bastante de 0.1
        */
var then = new Date(2021, 1, 8); // O 8º dia do 1º mês de 2021
var later = new Date(2021, 1, 8, // O mesmo dia, às 5:10:30 da tarde, hora local
 17, 10, 30);
var now = new Date(); // A data e hora atuais
var elapsed = now - then; // Subtração de data: intervalo em milissegundos
document.write (later.getFullYear() )// => 2021
document.write("<br>");
document.write (later.getMonth()) // => 1: meses com base em um
```

TIPO TEXTO

Uma string é uma sequência ordenada imutável de valores de 16 bits, cada um dos quais normalmente representa um caractere Unicode – as strings são um tipo de JavaScript usado para representar texto. O comprimento de uma string é o número de valores de 16 bits que ela contém. As strings (e seus arrays) de JavaScript utilizam indexação com base em zero: o primeiro valor de 16 bits está na posição 0, o segundo na posição 1 e assim por diante. A string vazia é a string de comprimento 0

```
<body>

    <h1> Imprimindo tipos de dados (Tipo Texto) </h1>

    <script>

        /* JavaScript têm muita precisão e
        podem se aproximar bastante de 0.1
        */
var p = "π"; // π é 1 caractere com posição de código de 16 bits 0x03c0
var e = "e"; // e é 1 caractere com posição de código de 17 bits 0x1d452
var nome ="Sandra Bozolan"

document.write (p.length )// => 1: p consiste em 1 elemento de 16 bits
document.write("<br>");
document.write (e.length) // => 2: a codificação UTF-16 de e são 2 valores de 16 bits:
document.write("<br>");
document.write (nome.length) // => 3: a codificação UTF-16 de e são 14 valores de 16 bits:
document.write("<br>");
document.write (nome) // => 3: a codificação UTF-16 de e são 14 valores de 16 bits:
document.write("<br>");
    </script>
```

PALAVRAS RESERVADAS

O JavaScript reserva vários identificadores como palavras-chave da própria linguagem. Você não pode usar essas palavras como identificadores em seus programas:

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

Palavras marcados com * são novos no ECMAScript5

LÓGICA

Se a lógica é escrita a partir de um determinado evento, não é necessário o uso dos comandos <SCRIPT> e </SCRIPT>. Os comandos JavaScript são sensíveis ao tipo de letra (maiúsculas e minúsculas) em sua sintaxe. Caso seja cometido algum erro de sintaxe quando da escrita de um comando, o JavaScript interpretará, o que seria um comando, como sendo o nome de uma variável.

Operadores lógicos

Em geral são usados com expressões que retornam valores booleanos, isto é, verdadeiro ou falso.

Operador	Descrição	Exemplo(s), supondo a = 3 e b = 5

&&	E lógico: retorna verdadeiro se ambas as expressões são verdadeiras e falso nos demais casos	a==3 && b<10 // retorna verdadeiro
		a!=3 && b==5 // retorna falso

||	OU lógico: retorna verdadeiro se pelo menos uma das expressões é verdadeira e falso se todas são falsas	a==3 || b<10 // retorna verdadeiro
		a!=3 || b==5 // retorna verdadeiro
		a==1 || b==3 // retorna falso

!	NÃO lógico: retorna verdadeiro se o operando é falso e vice-versa	! (a==3) // retorna falso ! (a!=3) // retorna verdadeiro

LIVROS RECOMENDADOS