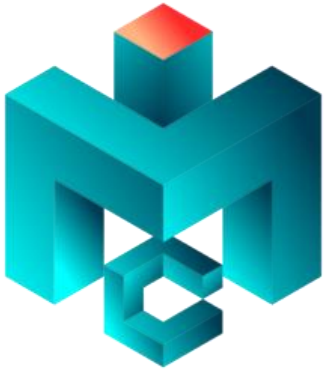


Aula – 7
Introdução ao JavaScript

Disciplina: XDES03 – Programação Web

Prof: Phyllipe Lima Francisco
phyllipe@unifei.edu.br

Universidade Federal de Itajubá – UNIFEI
IMC – Instituto de Matemática e Computação

Agenda

- ❑ Variáveis e Constantes
- ❑ Tipos Primitivos
- ❑ Condicionais
- ❑ Arrays
- ❑ Repetição

JavaScript

O que é JavaScript?

Linguagem de programação utilizada **principalmente** para adicionar interatividade em páginas web

Quais suas características?

Multi-paradigma, de tipagem dinâmica e do tipo função *first-class*

Variáveis e Constantes

Variáveis com *Let* - Definição

- Para criar variáveis em JS utilizamos a palavra-chave "let".
- Dado que JS é uma linguagem de tipagem dinâmica, o tipo só será definido em tempo de execução.

Variáveis com *Let* - Exemplo

```
let x; //declarando a variável cujo identificador é x
x = 3; //atribuindo valor após a definição
console.log(x); // 3
let y = 4; //atribuindo valor no momento da definição
console.log(y);// 4
```

Constantes com *const* - Definição

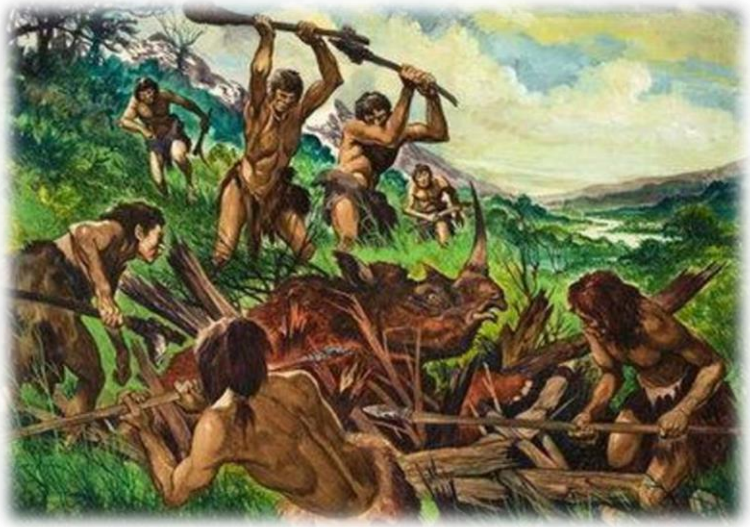
- Área de memória que não pode ser modificada após ter o seu valor atribuído.
- Em JS usamos a palavra-chave "const".
- Uma das principais razões pelas quais é importante usar "const" é para evitar a reatribuição acidental de valores de variáveis e referências.

Constantes com *const* - Exemplo

- O uso de constantes aprimora a legibilidade e ajuda a criar um código mais seguro.

```
const x = 7;
x = 8; //ERRO
```

Tipos Primitivos



Tipos Primitivos em JS

- Number
- Boolean
- String
- Null
- Undefined

Numbers - Definição

- A linguagem JS possui um único tipo de dado para número. Esse tipo é capaz de armazenar números inteiros e valores reais.
- Essa abordagem é diferente de outras linguagens populares como C e Java. Pois estas utilizam tipos diferentes, como “int” e “float”.

Numbers – Precisão/Capacidade

- Formato IEEE 754 64bits com dupla precisão.
- Sua precisão pode variar de 15 a 17 casas decimais
- Existem diversos métodos/propriedades que nos informam valores máximos e mínimos para o tipo “number”.

Numbers – Precisão/Capacidade

```
//Maior valor
console.log(Number.MAX_VALUE); //1.7976931348623157e+308

//Menor valor
console.log(Number.MIN_VALUE); //5e-324

//Maior valor seguro para cálculos sem perder precisão
console.log(Number.MAX_SAFE_INTEGER); //9007199254740991

//Menor valor seguro para cálculos sem perder precisão
console.log(Number.MIN_SAFE_INTEGER); //-9007199254740991
```

Numbers – Exemplos

```
let idade = 34;
console.log(typeof(idade));//number

let telefone = 988998899;
console.log(typeof(telefone));//number

let peso = 84.56;
console.log(typeof(peso));//number

let temp = -14.673561;
console.log(typeof(temp));//number
```

Numbers – Operações Matemáticas

- Podemos executar operações matemáticas básicas com valores do tipo “number” tais como:
  - Soma (+), Subtração (-), Multiplicação (*).
  - Divisão (/), Resto (%), Exponenciação (**).
  - Outros..

Numbers – NaN (Not a Number)

- Algumas operações matemáticas podem resultar em NaN (Not a Number).
- Exemplo: 0/0 irá resultar em NaN
- O valor NaN é do tipo “number” e representa algo que não é um valor numérico.

Numbers – Funções em *Math*

```
let x = 34.56;
let y = -78.12;

//remove a parte fracionária
console.log(Math.floor(x));//34

//arredonda para cima
console.log(Math.ceil(x));//35

//valor absoluto
console.log(Math.abs(y));//78.12

//exponenciação da base 2 pelo expoente 3
console.log(Math.pow(2,3)); //8

//constante PI
console.log(Math.PI); //3.141592653589793
```

Numbers – Incremento/Decremento

```
let x = 6;
console.log(x++);//6
console.log(x);//7
console.log(++x);//8
console.log(--x);//7
```

Numbers – Números Aleatórios

- Geramos números aleatórios usando a função Math.random() que retorna um valor real aleatório entre 0 e 1 (exclusivo).
- É possível obter números inteiros e outras faixas de valores combinando com algumas funções matemáticas.

Numbers – Números Aleatórios - Exemplo

```
//Valor aleatório entre 0 e 9
let x = Math.floor(Math.random() * 10);

console.log(x);
//Valor aleatório entre 1 e 10

x = Math.floor(Math.random() * 10) + 1;
console.log(x);

//Valor aleatório entre 2 e 6
x = Math.floor(Math.random() * 5) + 2;
console.log(x);
```

Boolean - Definição

- Existem situações que podem apresentar apenas dois estados.
- Uma porta pode se encontrar `aberta` ou `fechada`.
- Uma lâmpada pode estar `acesa` ou `apagada`.
- Um valor numérico qualquer pode ser `maior` ou `menor` que uma dada referência.

Boolean - Definição

- Para representar tais situações, a linguagem JS oferece o tipo primitivo “Boolean”.
- Variáveis desse tipo podem ter o valor “true” ou “false”.
- Importante notar que estamos falando do valor true/false e não do literal “true” ou “false”.

Boolean - Exemplo

```
let x = true;
let y = false;
let z = "true"; //diferente de atribuir true ou false sem aspas
console.log(typeof(x)); //Boolean
console.log(typeof(y)); //Boolean
console.log(typeof(z)); //string
```

Boolean - Exemplo

- Como a tipagem em JS é dinâmica, nada impede de atribuirmos um valor de outro tipo a está variável.

```
let x = true;
console.log(typeof(x)); //boolean
x = 1;
console.log(typeof(x)); //number
```

Boolean – Por que usar?

- Ao invés de usar números como 1/0, o uso de Boolean aprimora a legibilidade do código e evita a possibilidade de usar valores fora do padrão (exemplo: “3” para verdadeiro e “-6” para falso).
- Esse cenário é conhecido como “magic numbers”

String - Definição

- Representa uma sequência de caracteres. É utilizada para armazenar informações textuais e precisam ser envolvidas por aspas duplas/simples.
- Podem ser envolvidas pelo acento grave (`) para criar *template strings*.

String - Exemplo

```
let nomeUsuario = "mestreDosMagos";
let nome = 'Maria’;
let cidade = `Itajubá`;
let endereco = `Moro em ${cidade}, no estado de Minas Gerais. `;

console.log(typeof(nomeUsuario));//string
console.log(typeof(nome)); //string
console.log(typeof(cidade)); //string
```

String – Comprimento e Concatenação

```
let cidade = 'Itajubá’;
console.log(cidade.length); //7

let nome = "João";
let sobrenome = “ da Silva";
let nomeCompleto = nome + " " + sobrenome;
console.log(nomeCompleto); //João da Silva
```

String – Concatenação com Números

```
let num = 2; //sou number
num += 1;
console.log(num); //3
let literal = "2"; //sou string
literal += 1; //Literal + Number = Literal
console.log(literal) //"21“

literal++ //??????
```

String – Métodos Auxiliares

```
let nome = "   João   ";
let ret;

nome = nome.trim(); //remove espaços em branco no início e fim
ret = nome.toLowerCase(); //todas as letras ficam minúsculas
ret = nome.toUpperCase(); //todas as letras ficam maiúsculas
ret = nome.startsWith("J"); //retorna true se nome se iniciar com "J"
ret = nome.startsWith("o",2); //true se nome se iniciar com "o" no
índice 2
ret = nome.endsWith("o"); //retorna true se nome terminar com "o"
```

String – Métodos Auxiliares com Argumentos

```
let frase = "Eu quero tomar café";

//indice onde se encontra a primeira ocorrência de "E"
frase.indexOf('E’));

//indice onde se encontra a primeira ocorrência de "quero"
frase.indexOf('quero');//3

//sequencia entre os caracteres na posição 0 (inclusivo) e 2 (exclusivo)
frase.slice(0,2); //"Eu“

//sequencia de caracteres a partir da posição 3
frase.slice(3); //"quero tomar café“

//sequencia de 4 caracteres a partir do final da string
frase.slice(-4); //"café“

frase.replace('café','suco');
```

String – Template Literals

- Sequencias de caracteres que permitem interpolação e embutir expressões que serão calculadas e substituídas em tempo de execução em seus *placeholders*.

String – Template Literals Exemplo

```
let precoCafe = 4.50;
let precoCoxinha = 6.00;
const msg = `O café ${precoCafe} e coxinha ${precoCoxinha} resultam no total
de $${precoCafe + precoCoxinha}.`;
console.log(msg);
//O café 4.5 e coxinha 6 resultam no total de $10.5.
```

Null e Undefined – Definição

- O tipo ***null*** se refere a ausência proposital de valor e, portanto, é necessária a operação de atribuição do valor null.
- O tipo ***undefined*** ocorre quando alguma variável não teve nenhum valor atribuído e apenas foi definida.

Null e Undefined – Exemplo

```
let usuarioLogado = null;
console.log(typeof(usuarioLogado)); //null
let x;
console.log(x);//undefined
```

Condicionais

Condicionais – Definição

- Estrutura que permite executar um trecho de código se uma dada condição for verdadeira ou falsa.
- Se nenhuma condição é testada, o código executa de forma sequencial.

Condicionais – Operadores de Comparação

- Podemos comparar valores e verificar se resulta em um valor verdadeiro ou falso

>	**maior que**
<	menor que
>=	maior ou igual
<=	menor ou igual
==	igual
!=	diferente
===	estritamente igual
!==	estritamente diferente

Condicionais – Operadores de Comparação

- Diferença entre a igualdade (==) e estritamente igual (===)

```
5 == '5'; //true
5 === '5'; //false
0 == false; // true
0 === false; //false
undefined == null; //true
undefined === null; //false
```

Condicionais – Estrutura If-Else

- Estrutura tradicional presente em diversas linguagens para testar condições.

```
if(condição){
    //executa se a condição é verdadeira
}else{
    //caso contrário
}
```

Condicionais – Estrutura If-Else

- Encadear diversos *else-if* com novas condições

```
if(condição1){
    //executa se a condição1 é verdadeira
}else if(condição2){
    //executa se a condição2 é verdadeira
}else if(condição3){
    //executa se a condição2 é verdadeira
}else{
    //executa se todas as condições forem falsas
}
```

Condicionais – Operadores Lógicos

- Operadores binários cujos operandos são expressões lógicas.
- O resultado é um valor booleano

AND (E)	**&&**
OR (OU)	**||**
NOT (NEGAÇÃO)	**!**

Condicionais – Valores Booleanos Internos

- Todo valor em JS, isoladamente, retorna um valor verdadeiro com exceção dos seguintes:
  - false
  - 0
  - “” (string vazia)
  - Null, undefined e NaN

Condicionais – Estrutura Switch

- Estrutura alternativa de controle condicional
- Se usa uma chave, e executa a comparação estritamente igual com os diversos casos.
- O primeiro que resultar em verdadeiro é executado e nenhuma outra comparação será feita.
- A estrutura executará até encontrar um *break*

Condicionais – Estrutura Switch

```
switch (valor testado) {
    case valor1:
        break;
    case valor2:
        break;
    default:
        break;
}
```

Arrays

Arrays - Definição

- Coleções heterogêneas que armazenam dados de forma indexada. O seu tamanho pode se modificar durante a execução.
- Mesmo que seja possível, pode ser confuso criar Arrays cujos elementos são de tipos diferentes.

Arrays – Exemplo Inicialização

```
//array vazio
let discentes = []

//array de strings
let cores = ['red','green','blue’];

//array de numbers
let versoesWindows = [95,98,7,8,8.1,10,11];

//array com tipos diferentes
let coisas = [true, 18,'café',null];


console.log(cores[1]);//green

console.log(versoesWindows[3]);//8

console.log(versoesWindows.length);//7
```

Arrays – Exemplo Atualizando Valores

```
let cores = ['red','green','blue’];
console.log(cores[1]);//green
cores[0] = 'yellow'; //Modificando o valor da posição 0 para yellow
console.log(cores[0]);//green
```

Arrays – Função Push/Pop

```
let num = [1,2,3];
console.log(num.length); // 3
num.push(4);
num.push(5,6);
console.log(num); //[1, 2, 3, 4, 5, 6]
console.log(num.length);//6
num.pop();
console.log(num); //[1, 2, 3, 4, 5]
console.log(num.length);//5
```

Arrays – Função Shift/Unshift

```
let num = [1,2,3];
console.log(num.length); // 3
num.unshift(4);
num.unshift(5,6);
console.log(num); //[5, 6, 4, 1, 2, 3]
console.log(num.length);//6
num.shift();
console.log(num); //[6, 4, 1, 2, 3]
console.log(num.length);//5
```

Arrays – Função Concat/Includes/IndexOf

```
const arr1 = ['a','b','c’];
const arr2 = ['d','e','f’];
const arr3 = arr1.concat(arr2);
console.log(arr3);//['a', 'b', 'c', 'd', 'e', 'f']
//includes verifica se um valor está presente
console.log(arr3.includes('b'));//true
console.log(arr3.includes('g'));//false
//indexOf verifica a posição de um valor presente.
console.log(arr3.indexOf('c'));//2
//Caso o valor não esteja presente, a função retorna -1
console.log(arr3.indexOf('g'));//-1
```

Arrays – Função Reverse

```
const arr1 = ['a','b','c’];
const arr2 = ['d','e','f’];
const arr3 = arr1.concat(arr2);
console.log(arr3);//['a', 'b', 'c', 'd', 'e', 'f']
//Reverse irá sobrescrever o array com os valores reversos
arr3.reverse();
console.log(arr3);//['f', 'e', 'd', 'c', 'b', 'a']
```

Arrays – Função Slice

```
const arr1 = ['a','b','c’];
const arr2 = ['d','e','f’];
const arr3 = arr1.concat(arr2);
console.log(arr3);//['a', 'b', 'c', 'd', 'e', 'f']
//Slice irá retornar um cópia de uma porção do array
//de acordo com os índices passados.
const arr4 = arr3.slice(1);
console.log(arr4);//['b', 'c', 'd', 'e', 'f']

const arr5 = arr3.slice(2,4);
console.log(arr5);//[‘c', ‘d']
```

Repetição

Repetição – Definição

- Estrutura de controle que executa um trecho de código ***enquanto*** uma dada condição for verdadeira.
- JS conta com as tradicionais estruturas *for* e *while*.

Repetição – *For*

- ***For*** tradicional utilizando um contador

```
let num = [1,2,3,4,5]

for(let i = 0; i < num.length; i++)
    console.log(num[i]);
```

Repetição – *For - in*

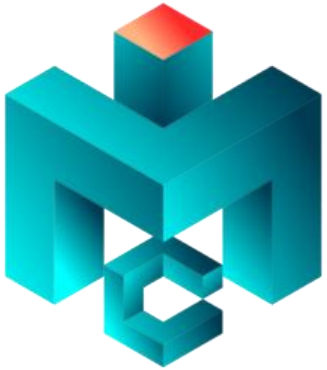
- ***For*** utilizado para percorrer todas as propriedades de um objeto.

```
const carro = {
    nome: "Jaum",
    idade: 32,
  };

for(const i in carro)
    console.log(i, carro[i]);
```

Repetição – *For - of*

- ***For*** utilizado para percorrer coleções

```
let num = [1,2,3];

for(const valores of num)
    console.log(valores);
```



Aula – 7
Introdução ao JavaScript

Disciplina: XDES03 – Programação Web

Prof: Phyllipe Lima Francisco
phyllipe@unifei.edu.br

Universidade Federal de Itajubá – UNIFEI
IMC – Instituto de Matemática e Computação