Linguagens de Script para Web
Tipos de Dados e Operadores

Centro Universitário Senac

Curso: Tecnologias em Sistemas para Internet
Professor: Dennis Lopes da Silva

Linguagens de Script Web – Java Script

- Criado em 1995 para tornar páginas web **dinâmicas e interativas**
- Executado diretamente no navegador, sem recarregar a página
- Base para sites modernos, SPAs, jogos, dashboards e muito mais
- Também usado no backend com Node.js, permitindo projetos full stack
- Grande comunidade e constante evolução
- Permite criar **experiências ricas e responsivas** para os usuários

Linguagens de Script Web – Java Script

Aplicações reais:

- Validação de formulários
- Menus interativos e animações
- Integração com APIs externas
- Sistemas de chat em tempo real

Objetivos da Aula

- Nesta aula aprofundaremos tipos de dados e operadores, com muitos exemplos e exercícios.
- Entender tipos primitivos e de referência.
- Praticar operadores: aritméticos, comparação, lógicos e atribuição.
- Resolver desafios de implementação (codar e testar).

Comentários em Java Script

Exemplos:

// Comentário de uma linha

/
Comentário
de múltiplas linhas
**/*

Comentários em HTML/CSS

```
<head>
   <title>HTML and CSS Example</title>
   <!-- this is an HTML comment...
   which can span multiple lines. -->
   <style>
      body: {
         color: red;
      }

      /* this is a CSS comment...
      which can span multiple lines. */
   </style>
   <script>
      console.log("echo"); // back in JavaScript...
      /* ...so both inline and block comments
      are supported. */
   </script>
</head>
```

Variáveis

Exemplos:

let nome = "Maria";

const anoNascimento = 2000;

var idade = 25;

let tempSalaC, sala1 = “sala de conferências", sala2 = "lobby";

Variáveis

- **let:** variável que pode mudar
- **const:** constante, não muda
- **var:** forma antiga, evitar

Exemplos:

```
let nome = "Maria";
const anoNascimento = 2000;
var idade = 25;
```

Identificadores

- Devem iniciar com uma **letra**, sinal **$** ou underscore (_);
- Podem conter **letras**, **números**, sinal **$** ou underscore(_);
- Não podem usar **palavras reservadas** da linguagem

Convenção: *Camel case*

Exemplos:

nomePai;
anoNascimento;
var idade = 25;

Script x Literal

- O JavaScript distingue o identificador do literal pelo uso das aspas, não causando confusão com os números porque os identificadores não podem se iniciar com um número.

Exemplos:

let sala1 = “sala de conferência"; // um literal

let salaAtual = sala1; // salaAtual tem o mesmo valor que sala1

let salaAtual = sala2; // erro, pois o identificador sala2 não existe

Tipagem Dinâmica

- JavaScript é dinamicamente tipado: o tipo é determinado pelo valor em tempo de execução.

Exemplos:

```
let x = 10; // número
x = 'texto'; // agora string
```

Tipos Primitivos

- String
- Number
- Boolean
- Null
- Undefined
- Symbol
- BigInt

Tipos Primitivos - Exemplos

Exemplos:

```
let s = 'Olá'; //String
let n = 3.14; //Number
let b = true; //Boleano – nunca coloque true/false entre aspas
let vazio = null; //null
let indef = undefined; //indefinido
let id = Symbol('id'); //Symbol
let big = 1234567890123456789On; //BigInt

console.log(typeof s, typeof n, typeof b);
```

Números – Operações e Particularidades

Números em JS: inteiros e ponto flutuante (IEEE 754).

Exemplo:

```
console.log(0.1 + 0.2); // 0.30000000000000004
console.log(Number.isNaN(NaN));
console.log(Number.isInteger(5));
```

Tipos Primitivos – Big Int

- O BigInt é um **tipo primitivo** do JavaScript usado para representar **números inteiros muito grandes**, maiores do que o tipo number consegue representar com segurança.

Number.MAX_SAFE_INTEGER // 9007199254740991 (≈ 9 quatrilhões)

Number.MIN_SAFE_INTEGER // -9007199254740991

Tipos Primitivos – Big Int (Exemplos de Uso)

```
console.log(9007199254740991 + 1); // 9007199254740992
console.log(9007199254740991 + 2); // 9007199254740992

let big = 9007199254740991n; // o "n" no final cria um BigInt
console.log(big + 2n); // 9007199254740993n

let a = 5n;
let b = 2;

console.log(a + b); // TypeError
console.log(a + BigInt(b)); // 7n
```

Tipos Primitivos – Números (Exemplos)

Exemplos:

```
let count = 10; // inteiro literal e também um double
const blue = 0x0000ff; // hexadecimal (hex ff = decimal 255)
const umask = 0o0022; // octal (octal 22 = decimal 18)
const roomTemp = 21.5; // decimal
const c = 3.0e6; // exponential (3.0 × 10^6 = 3,000,000)
const e = -1.6e-19; // exponential (-1.6 × 10^-19 = 0.00000000000000000016)
const inf = Infinity;
const ninf = -Infinity;
const nan = NaN; // "not a number"
```

Strings – Exemplos e Template Literals

Atribuição, concatenação e template.

Exemplos:

```
let nome = 'Carlos';
let sobrenome = "SilvaCarlos";
let saud = 'Olá, ' + nome + '!';
let saud2 = `Olá, ${nome}! Hoje é ${newDate().toLocaleDateString()}`;
console.log(saud, saud2);
```

Strings – Escape

Exemplos:

```
const dialogo = 'Eu entrei na sala e disse "Olá meu amigo!", assim que o vi.';

const imperative = "Don't do that!";

const dialog = "Sam looked up and said "don't do that!" to Max."; // erro

const dialog1 = "Ele olhou e disse:\"Não faça isso!\" para Maria.";

const s = " Em JavaScript, use \\ como um caracter de escape em strings.";
```

Strings – Escape

Exemplos:

\n Quebra de linha

"Line1\nLine2“

\r “Retorno do Carro”

"Windows line 1\r\nWindows line 2“

\t Tabulação

"Speed:\t60km//h"

Number como String

Exemplos:

```
const resultado1 = 3 + '30';

const resultado2 = 3 * '30’;
```

Quando trabalhar com números não use aspas

Tipos de Referência – Objetos e Arrays

Objetos e arrays armazenam referências.

Exemplos:

```
let pessoa = { nome: 'Ana', idade: 28 };
let frutas = ['maçã','banana','laranja'];
console.log(pessoa.nome, frutas[0]);
```

Operadores Aritméticos - Exemplos

+, -, *, /, %, ** (potência)

Exemplos:

let a = 7, b = 3;
*console.log(a + b, a - b, a * b, a / b, a % b, a ** b);*

Operadores de Atribuição e Incremento

=, +=, -=, *=, /=, ++, --

Exemplos:

```
let x = 5; x += 2; // 7
x++; // 8
console.log(x);
```

Comparação

- == (igualdade simples)
- === (igualdade estrita)
- != (diferente)
- !== (diferente estrito)
- > , <, >=, <=

Comparação com == e com ===

'==' compara com coerção de tipo; *'==='* compara valor e tipo (recomendado).

Exemplos:

```
console.log(0 == '0'); // true
console.log(0 === '0'); // false
```

Operadores Lógicos - Exemplos

&& (AND), **||** (OR), **!** (NOT), **short circuit**

Exemplos:

function foo(){ console.log('foo'); return true; }

console.log(false && foo()); // foo não é chamado

console.log(true || foo()); // foo não é chamado

typeof e instanceof

typeof informa tipos primitivos;

instanceof verifica protótipo/constructor.

Exemplos:

```
console.log(typeof []); // object
console.log([] instanceof Array); // true
```

Arrays – Métodos Úteis

push, pop, shift, unshift, map, filter, reduce, forEach, find

Exemplos:

```
let nums = [1,2,3,4];
nums.push(5);
nums = nums.map(n => n*2);
console.log(nums);
```

Array.push()

Adiciona um ou mais elementos ao final do array.

Exemplo:

```
const arr = [1, 2];
arr.push(3);
console.log(arr); // [1, 2, 3]
```

Array.pop()

Remove o último elemento do array e retorna ele.

Exemplo:

```
const arr = [1, 2, 3];
const removido = arr.pop();
console.log(removido); // 3
console.log(arr); // [1, 2]
```

Array.shift()

Remove o primeiro elemento do array e retorna ele.

Exemplo:

```
const arr = [1, 2, 3];
const removido = arr.shift();
console.log(removido); // 1
console.log(arr); // [2, 3]
```

Array.unshift()

Adiciona um ou mais elementos no início do array.

Exemplo:

```
const arr = [2, 3];
arr.unshift(1);
console.log(arr); // [1, 2, 3]
```

Array.reduce()

Executa uma função acumuladora em todos os elementos do array, retornando um único valor.

Exemplo:

```
const arr = [1, 2, 3, 4];
const soma = arr.reduce((acumulador, valor) => acumulador + valor, 0);
console.log(soma); // 10
```

Array.forEach()

Executa uma função para cada elemento do array (não retorna nada).

Exemplo:

```
const arr = [1, 2, 3];
arr.forEach(n => console.log(n * 2));
// Imprime 2, 4, 6 no console
```

Array.find()

Retorna o primeiro elemento que satisfaz a condição da função.

Exemplo:

```
const arr = [1, 2, 3, 4];
const encontrado = arr.find(n => n > 2);
console.log(encontrado); // 3
```

Array.map()

Cria um novo array aplicando uma função a cada elemento do array original.

Exemplo:

```
const numeros = [1, 2, 3];
const dobrados = numeros.map(n => n * 2);
console.log(dobrados); // [2, 4, 6]
```

Array.filter()

Cria um novo array contendo apenas os elementos que passam no teste definido por uma função.

Exemplo:

```
const numeros = [1, 2, 3, 4];
const maioresQueDois = numeros.filter(n => n > 2);
console.log(maioresQueDois); // [3, 4]
```

Interação com Usuário

Exemplo:

```
let nome = prompt("Qual seu nome?");
alert("Olá, " + nome + "!");
```

Convertendo Tipos

Exemplo:

```
let idade = prompt("Digite sua idade:");
idade = Number(idade);
console.log(typeof idade); // number
```

Condicional: if/else

Exemplo:

```
if (idade >= 18) {
  alert("Você é maior de idade.");
} else {
  alert("Você é menor de idade.");
}
```

Introdução a Loops

Exemplo:

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

JavaScript com HTML

O JavaScript pode ser inserido diretamente no código HTML usando a tag <script>.

Pode estar no <head>, no <body> ou em arquivo externo.

JavaScript com HTML (Exemplo)

```
<!DOCTYPE html>
<html>

<head>
   <title>Exemplo JS com HTML</title>
   <script>
      function mostrarMensagem() {
         alert('Ola do JavaScript!');
      }
   </script>
</head>

<body>
   <h1>Exemplo simples</h1>
   <button onclick="mostrarMensagem()">Clique aqui</button>
</body>

</html>
```

JavaScript com HTML (Boas Práticas)

- Colocar o <script> no final do <body>, assim o HTML carrega antes do JS executar.

- Usar o atributo defer na tag <script> no <head>, que faz o navegador carregar o script de forma assíncrona, executando-o só depois que o HTML estiver completamente carregado.

Exemplo:

<script src="script.js" defer></script>

Objetos – Acesso e Destructuring

Acesso por ponto/colchete e destructuring

Exemplo:

```
let p = { nome:'Ana', idade: 30 };
let { nome, idade } = p;
console.log(nome, idade);
```

Mini Projeto Final

Criar arquivo HTML com script que:

1. Pergunte o nome do usuário
2. Pergunte 3 notas
3. Calcule a média
4. Mostre um alert com o resultado

Boas Práticas e Erros Comuns

- Prefira **const** quando possível e **let** quando precisar reatribuir.
- Use === em vez de == para evitar coerção indesejada.
- Cuidado com operações em ponto flutuante; use toFixed/Math.round quando necessário.
- Teste valores com typeof e validações antes de operar.

Material de Apoio e Próximos Passos

- MDN Web Docs (referência oficial).

Semana 3: controle de fluxo (if/switch/for/while) — iremos aplicar tudo que vimos hoje.