Innominatum CONSULTORIA

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Innominatum CONSULTORIA

JavaScript

Módulo 2

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Innominatum CONSULTORIA

Sintaxe JS

Sessão 5

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Objetivo geral

Dotar os formandos dos conhecimentos fundamentais da linguagem JavaScript, permitindo-lhes compreender a sua sintaxe base, o funcionamento das variáveis, dos tipos de dados e das operações, bem como os principais comportamentos da linguagem relacionados com coerção de tipos.

Objetivos específicos

Ao final da sessão, os formandos deverão ser capazes de:

- Declarar e utilizar variáveis em JavaScript com **let** e **const**;
- Distinguir tipos de dados primitivos de tipos por referência;
- Aplicar operadores aritméticos, de atribuição e lógicos;
- Analisar comparações em JavaScript, compreendendo a coerção de tipos e a diferença entre igualdade solta (==) e estrita (===);.

Innominatum CONSULTORIA

Fundamentos e Sintaxe

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

O que é JavaScript?

- Uma linguagem de programação que serve para:
  - Criar lógica;
  - Tomar decisões;
  - Manipular valores.
- O JavaScript (JS) dará comportamento às páginas web.

Onde o JavaScript é executado?

- Ambientes de execução:
  - No navegador;
  - Ambientes de Desktop e Shell, através do Node;
  - Em servidores.

O papel do navegador

- Cada navegador tem um **motor JavaScript**, esse motor:
  - Lê código;
  - Executa instruções;
  - Gestão da tríade web (JS, HTML, CSS);
  - Mostra erros.

O console do navegador

- O console permite:
  - Testar código;
  - Depurar código;
  - Ver resultados;
  - Observar erros.

Primeiro contato com o código

- Abra o seu navegador:
  - F12;
  - Ou botão direito do rato (inspecionar);
  - Clique na aba console;

```
console.log("Olá JavaScript");
// Digite o comando acima.
```

JavaScript trabalha com valores

- JS trabalha com:
  - Números (number);
  - Texto (string);
  - Verdadeiro / falso (boolean);
  - Objetos (object).

O código é lido de cima para baixo

- JS executa:
  - A primeira linha;
  - Depois a segunda;
  - A sequência importa.

```
console.log("Primeiro");
console.log("Segundo");
```

Innominatum CONSULTORIA

Variáveis e Memória

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

O que é uma variável?

- Uma **variável** é um nome que aponta para um valor;
- O valor é guardado na **memória**;
- O nome permite reutilizar esse valor;

Declarar variáveis em JS

```
// Sintaxe básica
let idade = 30;
const nome = "Ana";
```

- let -> valor pode mudar;
- const -> valor não pode ser reatribuído;
- var -> valor pode mudar;

Reatribuição de valores

```
// Reatribuição permitida
let contador = 1;
contador = 2;
contador = contador + 1;
```

```
// Reatribuição não permitida
const pi = 3.14;
pi = 3.15; // ❌ erro
```

Com let o valor muda, contador agora será 3, contudo a variável continua sendo a mesma;

- O const é protegido pelo JS, ao tentar mudar o seu valor, o compilador lançará um erro.

Ordem de declaração e execução importa

```
console.log(x);
let x = 10;
// ❌ erro: Cannot access 'x' before initialization
```

O código é lido de cima para baixo, lembra?

Temporal dead zone (TDZ)

O **TDZ** é o período entre o início da execução do bloco (escopo) e o momento em que a variável é efetivamente declarada com um valor. Enquanto a variável está nessa "zona morta", ela existe, **mas não pode ser acessada**. Se você tentar acessá-la, o JavaScript lançará um **ReferenceError**.

Tipos de Dados

O que são tipos de dados?

Tipos de dados em JS são as categorias de valores que podem ser representados e manipulados pela linguagem. Eles definem a natureza dos dados (como números ou texto) que um programa processa para realizar cálculos ou alterar o estado de um sistema. O JS é uma linguagem dinamicamente tipada, ou seja, o tipo existe, mesmo sem ser declarado.

Tipagem dinâmica em JS

Em **JS** a tipagem é dinâmica, por que o tipo pertence ao valor e não a variável, e isso permite então que o JS converta os valores de um tipo para outro de forma livre, conforme a necessidade do programa.

```
let valor = 10;
valor = "dez";
```

Duas grandes categorias

- Em JS os tipos dividem-se em:
  - Tipos primitivos;
  - Tipos por referência.

Tipos primitivos

- São valores simples e imutáveis:
  - Number;
  - String;
  - Boolean;
  - Null;
  - Undefined.

Tipos primitivos – Exemplo

```
let idade = 30;     // number
let nome = "Ana";   // string
let ativo = true;   // boolean
let vazio = null;   // null
let indefinido;    // undefined
```

Tipos por referência

- São valores mais complexos:
  - Object;
  - Array;
  - Function.
- Os tipos por referência não guardam o valor diretamente, guardam uma referência na memória.

Tipos por referência – Exemplo

```
let pessoa = { nome: "Ana" };
// A variável aponta para um objeto na memória.
```

Referências partilhadas

```
let a = { valor: 10 };
let b = a;

b.valor = 20;
```

Resultado:

b.valor é 20;

Agora a terá o valor de 20.

Ambas apontam para o mesmo objeto.

Operadores Aritméticos

O que são operadores?

Os **operadores** são definidos como símbolos (geralmente sinais de pontuação ou palavras-chave) que atuam sobre valores (chamados de operandos) para produzir um novo valor. Eles representam a maneira mais comum de combinar expressões simples para formar expressões complexas. Eles portanto, permitem:

- Combinar valores;
- Transformar valores;
- Tomar decisões.

Tipos de operadores

- Operadores aritméticos (soma, subtração, multiplicação, divisão);
- Operador de resto (%);
- Operadores de atribuição (=, +=, -=, *=);
- Operadores lógicos (true, false).

Operadores de resto (%)

```
resto = 10 % 3 // 1
```

Estamos falando aqui de resto da divisão e não da percentagem, o resto da divisão é muito usado para:

- Verificar pares/ímpares;
- Ciclos;
- Lógica condicional.

Operadores de atribuição

```
let resultado = 10;
// O valor da direita é atribuído à variável da esquerda.
```

Atribuição combinada

```
let contador = 10;
contador += 1; // contador = contador + 1
contador -= 2; // contador = contador - 2
contador *= 3; // contador = contador * 3
```

Ordem das operações

```
ordem1 = 10 + 5 * 2 // 20
```

```
ordem2 = (10 + 5) * 2 // 30
```

A ordem é importante e segue quase que fielmente a mesma lógica dos precedentes matemáticos, chamamos esse conceito de **Precedência de Operadores**.

1. Agrupamento **()**;
2. Objetos e arrays;
3. Chamada de função;
4. Incremento/decremento (**++**, **--**);
5. Aritmética **, *****, **/**, **%**, **+**, **-**;
6. Comparações **>, <, >=, <=, ===**;
7. Lógicos && (AND)[maior precedência] || (OR);
8. Atribuição =.

Operadores lógicos e seus tipos

- Os operadores lógicos trabalham com:
  - True;
  - False;
- Eles possuem três tipos:
  1. && -> AND;
  2. || -> OR;
  3. ! -> NOT.

Operador AND (&&)

```
true && true   // true
true && false  // false
// Só retorna true se ambos forem verdadeiros.
```

Operador OR (||)

```
true || false // true
false || false // false
// Retorna true se pelo menos um for verdadeiro.
```

Operador NOT (!)

```
!true  // false
!false // true
// Inverte o valor lógico.
```

Innominatum CONSULTORIA

Comparações e Coerção de Tipos

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL

Comparações e Coerção de Tipos – Definição

A **coerção de tipos** (ou conversão de tipos) é a característica do JavaScript de ser muito flexível e **converter valores de um tipo para outro de forma automática** (ou livre) conforme a necessidade da operação. Se um programa espera um booleano, a linguagem converte o valor recebido adequadamente, se espera uma string, converte qualquer valor fornecido em string, e o mesmo ocorre para números.

A comparação, por sua vez, utiliza esses mecanismos de conversão de formas diferentes dependendo do operador escolhido que pode ser:

- Igualdade Estrita (===) e Relativa (==);
- Comparações Relacionais (<, >, <=, >=);
- Comparação de Objetos.
- As comparações sempre retornam um boolean (**true** ou **false**).

Operadores de comparação

```
10 > 5
10 < 5
10 >= 10
10 <= 9
// Todos retornam true ou false.
```

Igualdade em JS

JavaScript possui **dois operadores de igualdade** que são eles:

- **==** -> igualdade simples;
- **===** -> igualdade estrita (restrita ou de identidade).

Eles não fazem a mesma coisa.

Igualdade simples (==)

O JavaScript **converte os tipos** antes de comparar, é um caso claro de coerção de tipos e é muito comum em comparações e operações.

```
5 == "5" // true
// O JS converteu o string "5" para number 5 antes de comparar.
```

Igualdade estrita (===)

O JavaScript na igualdade estrita **compara o valor e o tipo**, ou seja,

aqui não há coerção de tipos, não acontece a conversão automática.

```
5 === "5" // false
// O JS não converteu o string "5" para number 5 antes de comparar.
```

Síntese

- O que é JS;
- Variáveis e memórias;
- Escopo de variáveis;
- Tipos de dados;
- Operadores;
- Comparações e Coerção de tipos.

Conclusão

Innominatum CONSULTORIA

Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

IPCA POLITÉCNICO DO CÁVADO E DO AVE
INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL
PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU
IAPMEI Parcerias para o Crescimento
PORTUGAL DIGITAL