

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA

JavaScript
Estrutura léxica

JavaScript

Estrutura léxica



Prof. Mr. Fabio Abrantes Diniz
fabio.abrantes.diniz@gmail.com

Introdução

Introdução

- **Programação nada mais é do que ensinar o computador**

- Algoritmos

Sequência de passos, conjunto de regras

- Lógica de programação

Maneira de pensar

- Sintaxe

Maneira correta de escrever



Introdução

Introdução

- **Instruções e sintaxe**

Toda linguagem é escrita com esses 2 princípios

- **Instruções (declarações)**

Ordens ao computador

- **Sintaxe**

Maneira correta de
escrever

```
alert("Fala, Dev!") // Fala, Dev!
```

```
alert((10 * 100) + " abraços")  
// 1000 abraços !
```



Existem **palavras reservadas** da linguagem.

Elas são responsáveis em dar significado a diversas instruções.

Léxica

- **sistema léxico** de uma linguagem de programação se refere ao conjunto de regras
 - Como os **tokens** são formados e interpretados

Sistema Léxico

Instruções

Comentários

Case sensitive

Palavras reservadas

Ponto e vírgula

Nomear variáveis



Estrutura

Léxica

- **Tokens**

- São as menores unidades reconhecidas pela linguagem



Estrutura Léxica

- **Comentários em JavaScript**

- Comentário em linha: **//**

- Comentário em bloco: **`/* */`**

- **Instruções**

- Qualquer comando que a gente passa no javascript

- **Case Sensitive**

- Diferencia maiúscula de minúscula



Estrutura Léxica

- **Palavras Reservadas
(chave)**





Estrutura

Léxica

- Nomeação de objetos (variáveis, funções, classes,...). **Não pode**

- Nomes compostos separados,
- iniciar com números , caracteres especiais ✓ Exceto underline e cifrão (não dar erro)



Estrutura Léxica



- **Nomeação de objetos (variáveis, funções, classes,...).**

Boas práticas:

- Iniciar com letra minúscula nome de variáveis e funções.
- Maiúscula para nome de classes e objetos
- Nomes compostos: usar a regra queima-case.
 - ✓ Colocar a primeira letra maiúscula





Estrutura Léxica

- **Ponto-e-vírgula no final da linha é opcional**
 - Só é obrigatório se tiver duas instruções na mesma linha ●

Usar língua inglesa (Boas práticas)



Estrutura Léxica

- **Tipagem dinâmica**
 - **Ex: Pode atribuir uma String a um variável numa linha**
 - ✓ E na outra linha atribuir um número a mesma variável
 - Ex2: um fação com mesmo identificador

mas que recebe parâmetros diferentes ou/e
retorna valores diferentes ou não

- **Aspas simples ou dupla**

- Pode usar qualquer uma que tem o mesmo

resultado



Declaração de variáveis

- Três formas principais de declarar variáveis: **var, let e const.**

- A sintaxe para declarar variáveis em JavaScript segue a estrutura:



- **Variáveis**
 - **Inicia com a palavras reservadas e**

nomeDaVariavel

✓ **var**, **let** e **const** (os dois últimos no ECMA6)

- **Regras de nomenclatura de nomes de variáveis;**
- **Não posso usar ou atribuir um valor antes de declarar uma variável;**
- **const** é mais veloz que **let** que é mais veloz que **var**



Revisão
programação
(javascript)

lógica de

- **Variáveis**

- **var**

- ✓ funciona em todos browsers cross-browser

- ✓ **Posso** criar variáveis com mesmo nome

- ✓ **Posso** declarar sem atribuir valor

- **let**

- ✓ **Não** deixa criar variáveis com mesmo nome

- ✓ **Posso** declarar sem atribuir valor

- **const**

- ✓ **Não** posso declarar sem atribuir um valor

- ✓ **Não** posso atribuir outro valor caso ele foi já atribuído



Tipos

primitivos

- **Tipos primitivos**
 - **number;**
 - **string;**
 - **boolean;**
 - **undefined**
 - **Null**
- **Podemos usar o operador **typeof** para descobrir o tipo de um valor**



Tipo primitivo



Number

- **number**

- Não distingue valores inteiros e valores flutuantes.



- Existem **três valores especiais** que são considerados **números**:

- **Infinity** (infinito), **-Infinity** (infinito negativo)
- **NaN** (não é um número – *not a number*)













Infinity

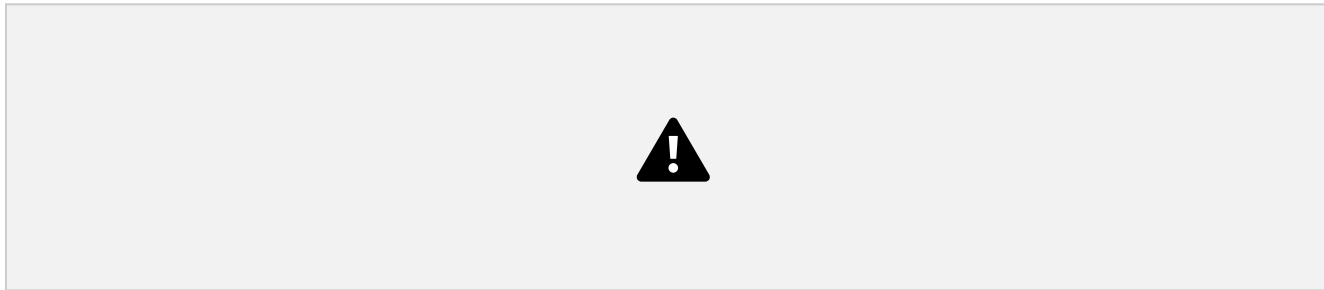
- **Infinity** (infinito) e **-Infinity** (infinito negativo) ■ representa o **infinito matemático**
- **Ocorre quando o resultado de uma operação:** ✓ É maior do que o **maior** ou **menor** número representável
- ✓ Divisão de um **número** por **zero**



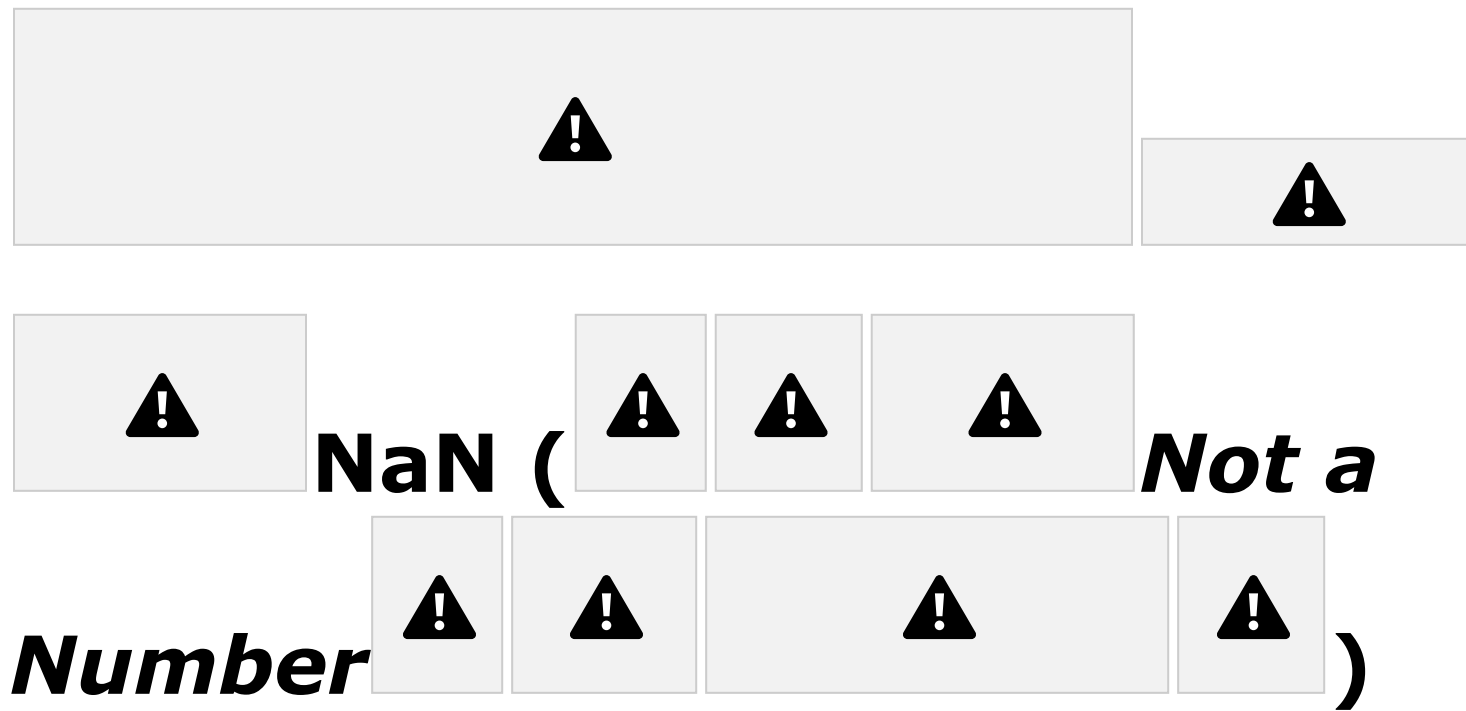
 NaN (   *Not a*
Number    )

- Quando realiza operações matemáticas com valores que **não são números** ou quando **o cálculo não faz sentido**.

- Operação matemática inválida

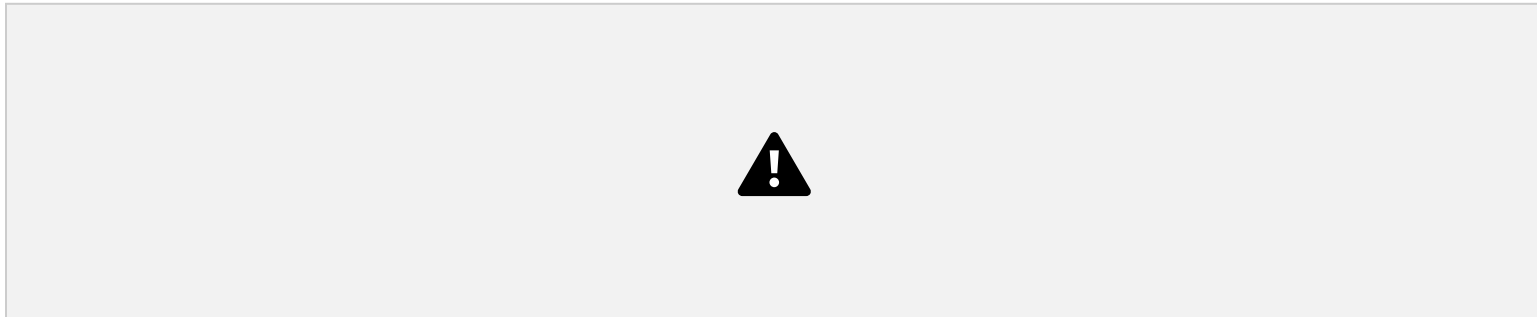


- Operação matemática com strings não numéricas



- Quando realiza operações matemáticas com valores que **não são números** ou quando **o cálculo não faz sentido**.

- Tentativa de converter valores inválidos para número



- Como verificar se um valor é **NaN**?
 - Usando a função **isNaN**(valor)



■ Cuidado!

✓ `isNaN()` converte o valor para número antes de verificar, o que pode gerar resultados inesperados.



NaN

(



Not a

Number



)

- **Curiosidade!**

- O NaN é o único valor em Javascript que nunca é igual a si mesmo!



String

- É uma sequência de caracteres usada para representar **texto**

- Estão incluídos entre:
 - **Duas Aspas simples:** `' '`
 - **Duas Aspas dupla:** `" "`
 - **Template literals:** crase `` `` (recomendado ES6+)



String

- **Operador de concatenação**

- Processo de juntar (ou unir) **duas** ou **mais strings** para formar uma nova.

✓ **Usando o + (operador sobrecarregado) ou template literals;**





String

- **Principais métodos**





Booleanos

- Diferente de number e string, apresenta apenas dois valores: **true e false**
- Geralmente é o resultado de comparações que fazemos em JavaScript (if, while, etc)





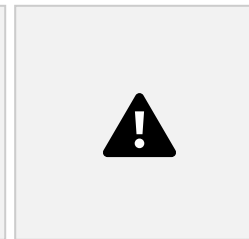
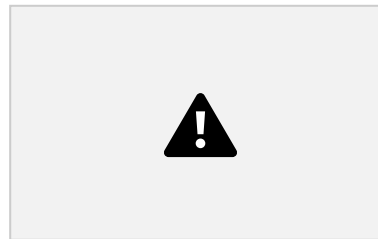
Vamos praticar!

- Transforme esse pseudocódigo em javascript

undefined



Tipos Null e





- **Null**

- **Representa ausência intencional de valor)**

✓ Essa variável existe, mas ainda não tem valor





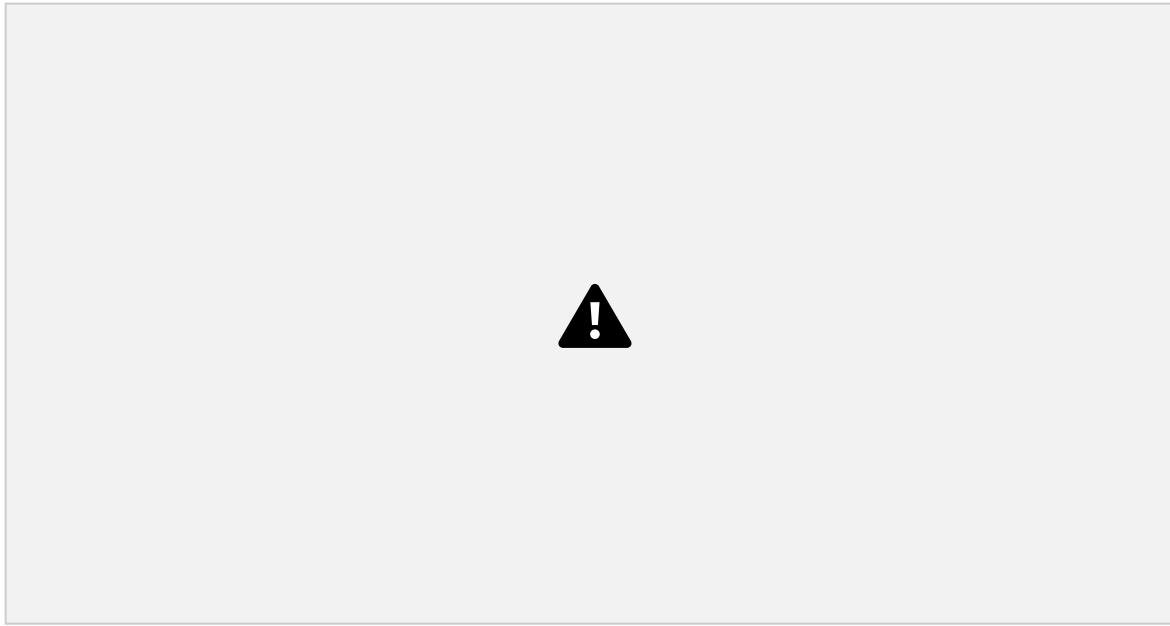
Tipos Null e

undefined



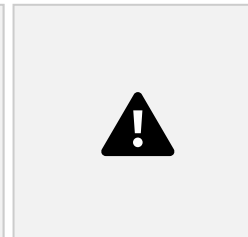
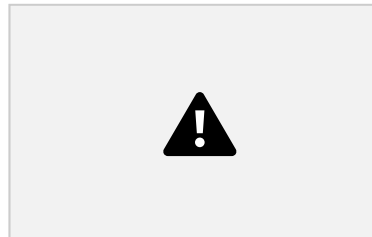
- **Undefined**

- Significa que **uma variável foi declarada, mas ainda não recebeu um valor**



Tipos Null e

undefined





- **Quando ocorre**

Undefined?

- Variável declarada, mas não inicializada
- Função sem retorno explícito
- Acesso a uma propriedade inexistente





Tipos Null e

undefined



null e Undefined?

- **Diferença entre**



Conversão implícita

- Acontece quando o próprio interpretador tenta **converter automaticamente:** ▪ **Tipos diferentes para completar uma operação**

✓ **Ex:** soma de um **número** com uma **string** contendo o **primeiro caractere sendo número**.

– **O resultado vai ser gerado uma string**

✓ **Ex:** multiplicação de um número com uma string contendo o **primeiro caractere sendo número**.

– **Dar number**



Exemplos

Conversão implícita

- **String + Número = tudo vira string**



- **String -/* Número → vira número**





Exemplos

Conversão implícita

- **String -/* Número** → **vira número**



Booleanos em operações



Conversão Explícita

- **Conversão explícita**

- parseInt(), parseFloat()
- toString(), String()
- Number()

✓ Não funciona se tiver o primeiro caractere junto com numero

- Mais sobre conversão

✓ https://www.w3bai.com/pt/js/js_type_conversion.html



Conversão Explícita

- **Exemplos Para Número**

Number() Operador unário +



`parseInt()` e `parseFloat()`



Conversão Explícita

- **Hexadecimal, Decimal e Binário**



Conversão Explícita

- **Hexadecimal <-> Decimal**



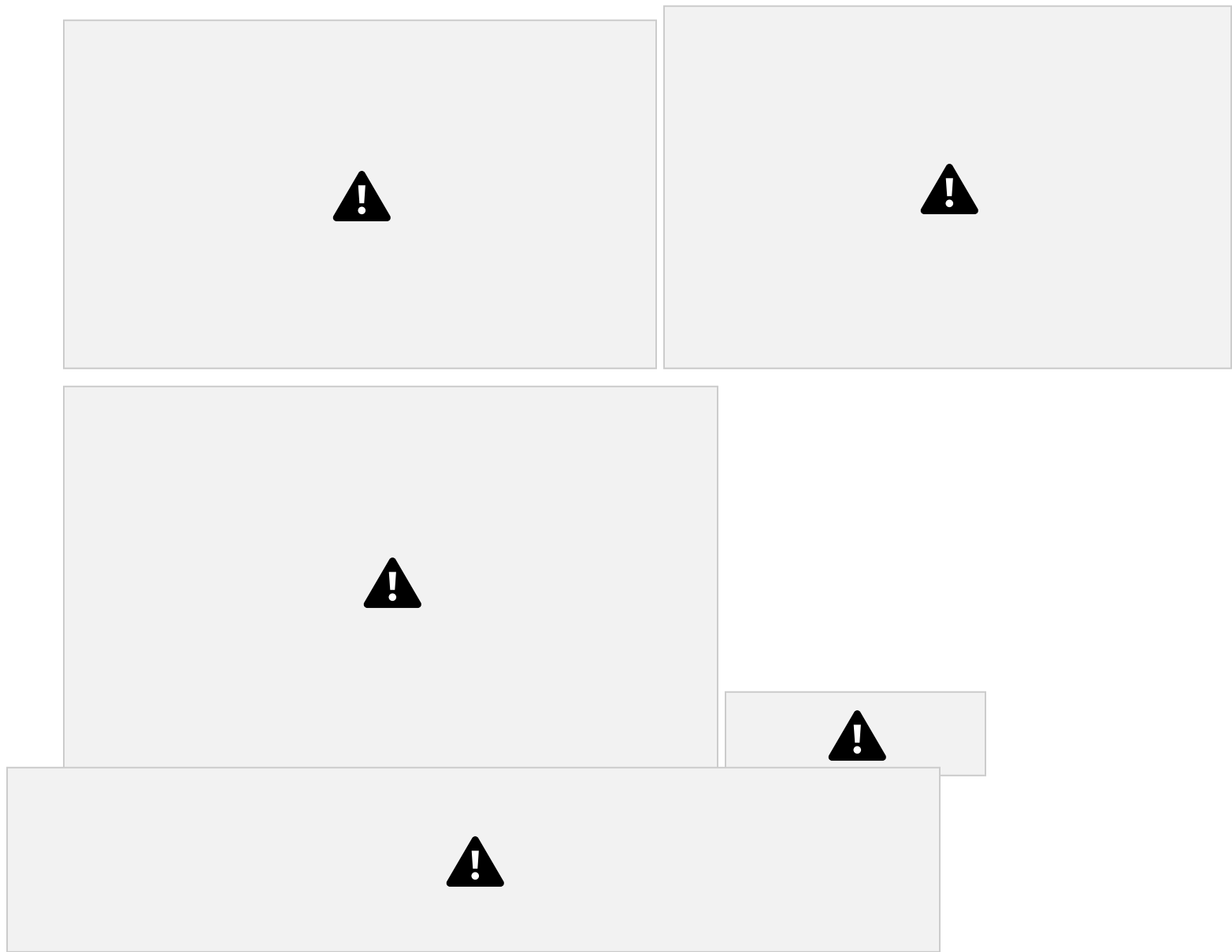
Conversão Explícita

- **Binário <-> Decimal**



Conversão Explícita

- **Exemplos Para String**



Conversão Explícita

- **String() versus toString()**



Conversão Explícita

- **Exemplos para Boolean**





Resumo Conversão Explícita

