



# Design Web

Prof. Romerito Campos

# Plano de Aula

**Objetivo:** conhecer a estrutura léxica da linguagem, tipos/variáveis/valores e declarações condicionais e de laço

## Conteúdo

- Estrutura Léxica

# JavaScript

- JavaScript é
  - Alto-nível
  - Dinâmica
  - Interpretada
  - Suporta os estilos Orientado a objetos e Funcional
- **JavaScript não é Java**

# JavaScript

- O core da linguagem dá suporte a
  - tipos números e textuais, arrays, sets e maps
  - não inclui funcionalidade padrão para **input/output**
- O ambiente original da linguagem foi o **Browser**
  - O input é obtido de mouse, teclado e *requisições HTTP*
- Desde 2010, após o surgimento o [Node](#), O java script pode acessar recursos do sistema operacional (Ler e Escrever arquivos, usar a rede etc).

# JavaScript

## Estrutura Léxica

# Estrutura Léxica - Case sensitive

- Case sensitive, espaços e quebras de linha (line breaks)
- Comentários
- Literais
- Identificadores e palavras reservadas
- Unicode
- Ponto-Vírgula opcionais (*Optional semicolons*)

# Estrutura Léxica

- **Variáveis**, palavras reservadas e identificadores devem ser escritos de maneira consistente.

```
//definindo variável (explicação a seguir)
let a = 10;
//vai ocorrer um erro!!!
if (A === 10) {
    // Uncaught ReferenceError: B is not defined
}
```

**Uncaught ReferenceError: B is not defined**

# Estrutura Léxica

- Variáveis, **palavras reservadas** e identificadores devem ser escritos de maneira consistente.

```
//uso do IF ao invés de if  
IF (10 === 10) {  
    /**Uncaught ReferenceError: B is not defined**  
}
```

**Uncaught ReferenceError: IF is not defined**

# Estrutura Léxica

- Comentários: Há três formas de comentar o código.

```
// comentário
```

```
/*comentário*/
```

```
/*
```

```
* comentário multi-linha
```

```
*
```

```
*/
```

# Estrutura Léxica

- **Literais:** valores que podem ser aplicados diretamente no código

```
12  
1.2  
"Hello World"  
'Hello'  
true  
false  
null
```

# Estrutura Lexica

- **Identificadores e palavras reservadas**
- Um **identificador** é utilizado para nomear constantes, variáveis, funções, propriedades de objetos, classes e etc.

```
//identificadores  
function teste () {  
    ///  
}  
  
let variavel = 10;  
const nome = "romerito";
```

- `teste`, `variavel` e `nome` são identificadores
- podemos acessar o conteúdo da variável e da constante pelo seus identificadores
- podemos executar a função pelo seu identificador: `teste()`

# Estrutura Lexica

- **Unicode:** os programas em JavaScript são escritos utilizando o conjunto de caracteres Unicode
- Isso permite utilizar símbolos e palavras, por exemplo, latinas na definição de identificadores

```
//exemplos de uso de Unicode  
let café = "moca";  
caf\u00e9 = "duplo moca";
```

**Abra o console do navegador e digite:**

```
console.log ("\u{1F600}")
```

# Estrutura Lexica

- **Unicode**

- evite utilizar símbolos especiais ou acentos
- Javascript não normaliza as definições de identificadores para evitar conflitos

```
//ambas as palavras com o espace \ são café  
const caf\u{e9} = 10;  
const cafe\u{301} = 10;
```

# Estrutura Lexica

- **Ponto-Vírgula Opcional (Optional Semicolons)**
  - ; é utilizado para separar declarações
  - quebras de linha pode ser tratadas como ;

```
//utilização explícita de ;  
let a = 3;  
let b = 3;
```

```
// JS trata a quebra de linha como;  
let a = 3  
let b = 3
```

# Estrutura Lexica

- **Ponto-Vírgula Opcional (Optional Semicolons)**

- ; é utilizado para separar declarações
- quebras de linha pode ser tratadas como ;

```
let a
a
=
3
console.log(a);
```

- Este exemplo funciona apenas rodando um script como o código `node script.js`.
- JavaScript compreende como `let a; a=3; console.log(3)`.

# Estrutura Lexica

- **Ponto-Vírgula Opcional (Optional Semicolons)**

```
//considere que existe 'x', 'f', 'a' e 'b'  
let y = x + f  
(a+b).toString()
```

- A declaração acima pode indicar duas linhas de código, mas acaba sendo interpretada como uma única linha de código.
- O '(' aparece como sendo parte da execução de `f()`. Javascript enxerga f como função: `let y = x + f(a+b).toString()`

# Referências

FLANAGAN, D. **JavaScript - The Definitive Guide**. 7. ed. Sebastopol, CA, USA: O'Reilly Media, 2020.