Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

Variáveis e tipos primitivos em JavaScript

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

Variáveis

Uma **variável** referencia a um espaço na memória do computador utilizado para **guardar informações** que serão usadas em seus programas.

```
<script type="JavaScript">
    var name = "Deadpool";
    var idade = 35;
</script>
```

Criação de Variáveis

```
var nomeVariavel = valor;
```

Instrução opcional para indicar a criação de uma variável

Regras para criação dos identificadores de variáveis

- Devem começar com **letra (a-z, A-Z)**, **cifrão ($)** e **sublinhado (_)**;
- **Não podem começar com dígitos**;
- É possível utilizar **acentos e símbolos em sua composição**;
- **Não podem conter espaços;**
- ***Case-sensitive***: JavaScript é uma linguagem sensível a maiúsculas e minúsculas, o que significa que `myVariable` e `myvariable` seriam considerados duas variáveis diferentes.
- **Não podem ser palavras reservadas** da linguagem, nem espaços em nomes de variáveis. Utilize **`camelCase`** ou **`snake_case`**.
- **Identificadores Unicode**: JavaScript suporta identificadores Unicode, o que significa que é possível usar caracteres especiais e letras de outros alfabetos, como 变量 (chinês para "variável"). No entanto, o uso de caracteres não-ASCII pode tornar o código mais difícil de ler e manter.

Exemplos de nomes **válidos**:

`base` **`area`** **`n1`** **`num_alunos`**

`iNumAlunos` **`_teste`**

Exemplos de nomes **inválidos**:

`1a` **`%aumento`** ***`num alunos`*** **`var`**

Variáveis no JavaScript

Case-sensitive: diferencia maiúsculas de minúsculas.

Utilizar nome de identificadores com algum **significado**.

JavaScript não diferencia números inteiros ou flutuantes.

Todos são do tipo **`number`**.

Tipos primitivos

primordiais

Tipagem dinâmica

`JavaScript` é uma linguagem de **tipagem dinâmica**.

Não é preciso declarar o tipo de uma variável antes de sua atribuição.

O tipo será automaticamente determinado quando o programa for processado.

Também é possível reatribuir uma mesma variável com um tipo diferente.

Tipos primitivos primordiais em JS

`7` `41` `-13` Inteiros

`0.5` `-37.2` `4.1` Fracionários (float)

`true`

`false`

`"CSTSI"`

`'JavaScript'`

`` `Rafa` ``

Tipos primitivos primordiais em JS

7 41 -13

0.5 -37.2 4.1

number

Number: JavaScript não diferencia inteiros de números de ponto flutuante. Todos são do tipo **number**.

```
var inteiro = -13
var flutuante = 124.13
```

Para delimitação de strings podemos usar **aspas duplas**, **aspas simples** ou **crase**.

Strings: representa sequências de caracteres.

“CSTSI”

‘JavaScript’

`Pelotas`

string

```
var texto = “JavaScript”
var texto2 = ‘tsi’
```

```
var teste = false
```

true

false

boolean

Tipos primitivos primordiais em JS

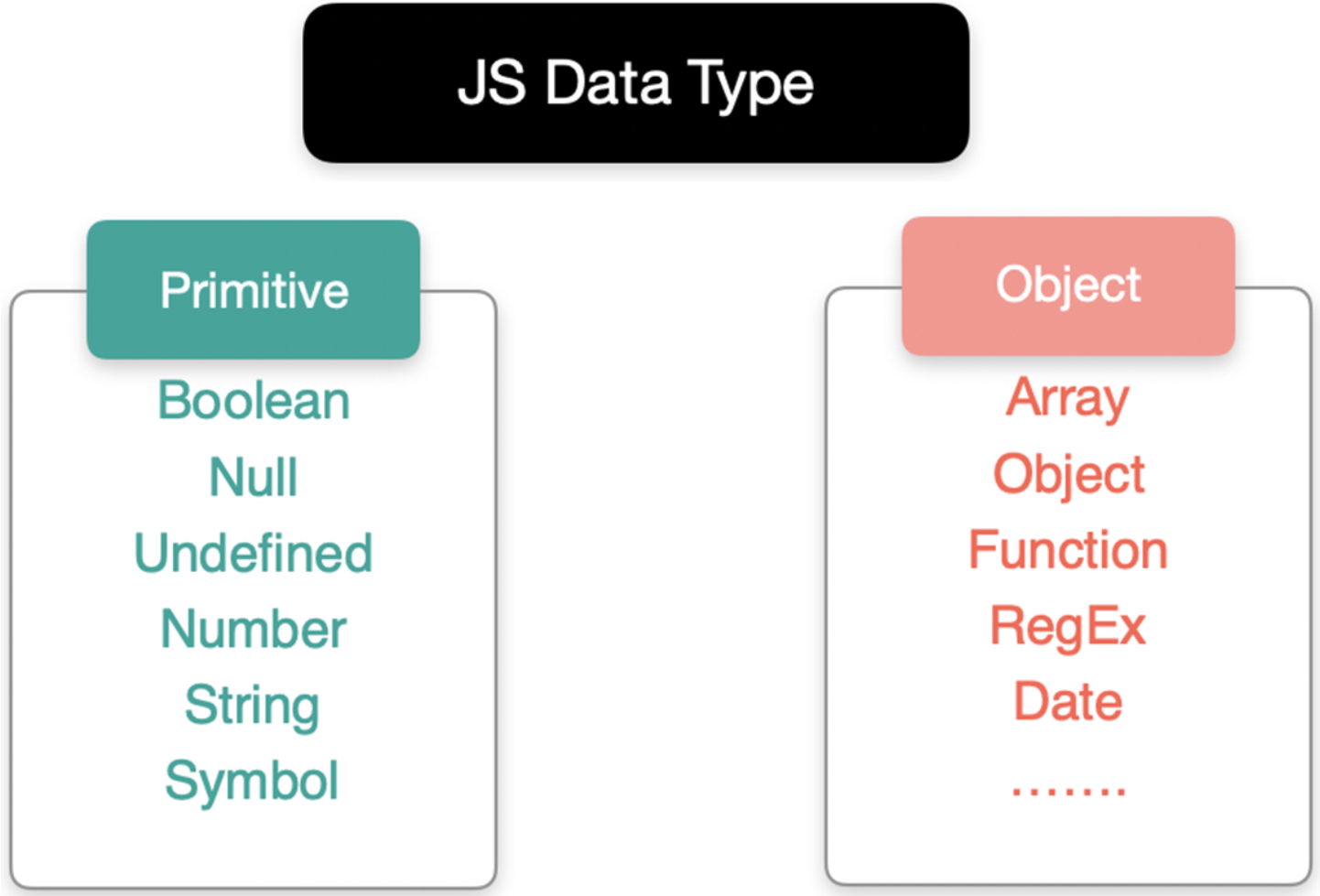
`undefined`: Representa uma **variável que foi declarada mas teve um valor atribuído**.
Exemplo: **`let x;`** (aqui o valor de x é `undefined`).

`null`: Representa a **ausência intencional de qualquer valor objeto**. É um valor especial que indica "nenhum valor" ou "valor desconhecido".
Exemplo: **`let y = null;`**

`Symbol`: Introduzido no ECMAScript 2015 (ES6), representa um identificador único e imutável. É usado frequentemente como chaves únicas para propriedades de objetos.
Exemplo: **`Symbol('description')`**

`BigInt`: Introduzido no ECMAScript 2020, representa números inteiros de precisão arbitrária. É usado para lidar com valores maiores que o máximo valor representável pelo tipo **`Number`**.
Exemplo: **`‘1234567890123456789012345678901234567890n’`**.

Tipos de dados no JavaScript



Comando typeOf

Permite descobrir qual o **tipo de dado** de determinada variável.

Útil, já que a linguagem permitir **atribuir dinamicamente um tipo de dado diferente do original** que a mesma havia sido declarada.

Isso acaba aumentando ou diminuindo o espaço ocupado pela variável na memória, a medida que o programa evolui.

Retornos de typeOf

Retorno	Descrição	Exemplo

undefined	Retornado quando a variável não foi inicializada.	`let x;`
		
		`console.log(typeof x); // "undefined"`

boolean	Retornado para valores booleanos (true ou false).	`let flag = true;`
		
		`console.log(typeof flag); // "boolean"`

number	Retornado para valores numéricos, incluindo inteiros e números flutuantes.	`let num = 42;`
		
		`console.log(typeof num); // "number"`

bigint	Retornado para valores do tipo BigInt.	`let bigIntValue = 1234567890123456789012345678901234567890n;`
		
		`console.log(typeof bigIntValue); // "bigint"`

string	Retornado para valores de texto.	`let text = "Hello, world!";`
		
		`console.log(typeof text); // "string"`

symbol	Retornado para valores do tipo Symbol.	`let sym = Symbol('description’);`
		
		`console.log(typeof sym); // "symbol"`

Retornos de `typeOf`

Retorno	Descrição	Exemplo

`object`	Retornado para valores que são objetos (incluindo arrays e null)	`let objeto = { name: 'Biro' };`
		`console.log(typeof objeto); // "object"`
		
		`let array = [1, 2, 3];`
		`console.log(typeof array); // "object"`
		
		`let nulo = null;`
		
		`console.log(typeof nulo); // "object"`
		`//(isso é um detalhe histórico e considerado um bug[1])`

`Function`	Retornado para funções (que são tecnicamente objetos).	`function minhaFuncao() {}`
		
		`console.log(typeof minhaFuncao); // "function"`

[1]O comportamento do `typeof` para `null` é uma consequência da implementação histórica do JavaScript e é considerado um erro ou uma limitação da linguagem que foi mantida por questões de compatibilidade. Para evitar confusão, é importante usar verificações específicas para null e entender as peculiaridades da linguagem.

Utilizando dados

Capturando os dados

Coloque o resultado da expressão a direita, dentro da variável nome

```
var nome = window.prompt('Qual seu nome?');
```

Usando os dados capturados

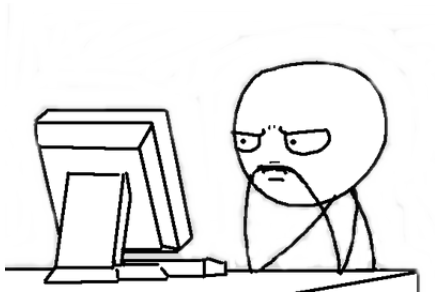
```
window.alert('Prazer te conhecer, ' + nome + '!!!');
```

Concatena as **strings** lidas.

Calculando expressões

```
<script>

    var num1 = window.prompt('Digite um número: ');
    var num2 = window.prompt('Digite outro número: ');

    var s = num1 + num2;

    window.alert('Soma dos valores: ' + s );

</script>
```

Exemplo



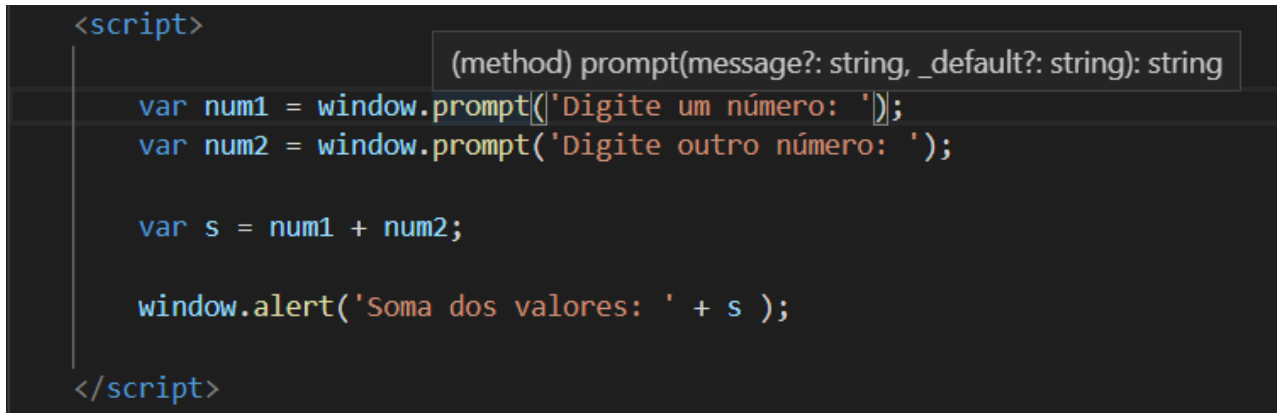
Calculando expressões

```
<script>

    var num1 = window.prompt('Digite um número: ');
    var num2 = window.prompt('Digite outro número: ');

    var s = num1 + num2;

    window.alert('Soma dos valores: ' + s );

</script>
```

⚠

number + number (adição)

string + string (concatenação)



Exemplo

Conversão (*cast*) de dados

`Number.parseInt(n)`

`Number.parseInt()` converte uma ***string*** **em um número um inteiro**.

`Number.parseFloat(n)`

`Number.parseFloat()` converte uma ***string*** **em um número real**.

```
<script>
    var num1 = Number.parseInt(window.prompt('Digite um número: '));
    var num2 = Number.parseInt(window.prompt('Digite outro número: '));

    var s = num1 + num2;
    window.alert('Soma dos valores: ' + s );
</script>
```

Exemplo

`Number.parseInt()` converte um argumento de ***string*** **e retorna um inteiro da raiz ou base específica**.

Cast de dados

Usando apenas **`Number(n)`**

Delega ao `JavaScript` a determinação se o número é inteiro ou real, para que faça as transformações necessárias.

```
<script>
    var num1 = Number(window.prompt('Digite um número: '));
    var num2 = Number(window.prompt('Digite outro número: '));

    var s = num1 + num2;

    window.alert('Soma dos valores: ' + s );
</script>
```

Exemplo

Não elimina as soluções anteriores, porque existem casos em que podemos **trabalhar com tipos de dados específicos**, dependendo da lógica da aplicação desenvolvida.

Cast de dados (Number -> String)

`String(s)` **Transforma qualquer coisa que pode ser convertida em `String`.**

`s.toString()` **Retorna uma `String` que representa o objeto especificado**

```
<script>
        var num1 = Number(window.prompt('Digite um número: '));
        var num2 = Number(window.prompt('Digite outro número: '));

        var s = num1 + num2;

        window.alert('Soma dos valores: ' + s.toString);
        window.alert('Soma dos valores: ' + String(s));
 </script>
```

Exemplo

Concatenação x Template String


```
<script>

        var num1 = Number(window.prompt('Digite um número: '));
        var num2 = Number(window.prompt('Digite outro número: '));

        var soma = num1 + num2;

        // Concatenação
        window.alert('Soma de ' + num1 + " e " + num2 + 'é ' + soma);









</script>
```

Concatenação x Template String

Método usado nas versões mais recentes do JavaScript.

***Template* Strings são *strings* que permitem expressões embutidas.**

Outra forma de criar `strings` e tornar o código um pouco mais legível.

```
<script>

        var num1 = Number(window.prompt('Digite um número: '));
        var num2 = Number(window.prompt('Digite outro número: '));

        var soma = num1 + num2;

        // Concatenação
        window.alert('Soma de ' + num1 +  ' e '  + num2 +  ' é ' + soma);


        // Template String
        window.alert(`Soma de ${num1} e ${num2} é ${soma}`);

</script>
```

[Exemplo]

Placeholders

Utiliza **crases** para delimitar.

Formatando String

Métodos e propriedades úteis para se trabalhar com strings.

```
var s = "IFSul";

s.length;       // atributo que retorna o número de caracteres da string

s.toUpperCase(); // método que muda todas as letras da string para MAIÚSCULO

s.toLowerCase(); // método que muda todas as letras da string para minúsculo
```

Formatando números

Métodos e propriedades úteis para se trabalhar com `strings`.

```
var n = 1302.1;

alert("A: " + n);

alert("B: " + n.toFixed(2));    // método para setar a quantidade de números depois da vírgula

alert("C: " + n.toFixed(2).replace('.', ',’));

alert(n.toLocaleString('pt-BR', {style: 'currency', currency: 'BRL'}));
```

Essa página diz
1302.1

Essa página diz
1302.10

Essa página diz
1302,10

Essa página diz
R$ 1.302,10

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

Variáveis e tipos primitivos em JavaScript

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas