1.1.1. Declarando variáveis - JS

Javascript

- Javascript é uma linguagem dinamicamente tipada, o que significa que as variáveis não precisam de uma declaração de tipo, como em Java.
- Javascript também é uma linguagem fracamente tipada, o que significa que existe uma conversão implícita de tipos. Por exemplo, o código `1 + "1"` gera erros em linguagens fortemente tipadas, mas em JS o resultado é a string `"11"`

Let keyword

```
let numberOfSeasons = 6;
let numberOfEpisodes = 12;
let episodeTime = 45;
let commercialTime = 5;

let totalShowTime = numberOfSeasons*numberOfEpisodes*
(episodeTime+commercialTime)
console.log(totalShowTime)
```

- Observação: o criador do Javascript tentou copiar a aparência geral do Java e de linguagens C-like. Isso pode ser observado, por exemplo, pela convenção de usar camelCase para nomes de variáveis e pelo término de linhas com ponto-e-vírgula.

Const keyword

```
const hoursPerDay = 24;
const minutesPerHour = 60;
const secondsPerMinute = 60;
```

```
const secondsInADay =
hoursPerDay*minutesPerHour*secondsPerMinute
console.log(secondsInADay)
```

- A funcionalidade do `const` é similar ao `final` em Java: declarar uma variável imutável, que não pode ser modificada depois de sua criação.

> **Typescript**
>
> A linguagem Typescript é um superset de JS, que tenta forçar o programador a usar um sistema estaticamente tipado. Essa pequena diferença fez Typescript se tornar uma linguagem default para programação web em muitas empresas e frameworks, justamente por melhorar a eficiência de programação por diminuir bugs obscuros gerados pela ausência e conversão implícita de tipos.

```
let num:number = 4;
console.log(num)
```

var keyword

```
function example() {
    var x = 10;
    if (true) {
        var x = 20; // This redeclares the same variable
        console.log(x); // Outputs: 20
    }
    console.log(x); // Outputs: 20
}
```

- `var` permite a redeclaração de variáveis, sem erros

- `var` possuem o escopo global/funcional, o que pode acarretar bugs difíceis de compreender
- No desenvolvimento moderno, `let` e `const` são mais utilizados por terem um comportamento mais previsível do que var

1.1.2. Declarando variáveis - Java

- Java é uma linguagem estaticamente tipada, o que significa que as variáveis precisam de uma declaração explícita de tipo.
- Java também é uma linguagem fortemente tipada, o que significa que não existe uma conversão implícita de tipos.

> **Mas não há conversão de tipos no print?**
>
> Algo que me deixou confuso quando eu li a definição acima é que o print do Java não necessita a conversão de números para textos, diferentemente de Python, por exemplo.

```
public class HelloWorld {
        public static void main(String[] args) {
                int num = 7;
                System.out.println("Meu número favorito é
" + num);
        }
}
```

> **Note**
>
> Porém esse fenômeno é chamado de type casting ou type coercion, e não é uma violação da regra, pois é específico

para o print. A violação ocorre quando variáveis podem ser declaradas misturando tipos, como no exemplo abaixo.

```
let a = "1"
let b = 1
let c = a+b

console.log(c)
```

- Java, diferentemente de Python e JS, é uma linguagem compilada. Sua filosofia de criação foi de criar um código que funcionaria em várias máquinas e sistemas operacionais possíveis, usando o JVM apropriado para cada situação.

1.1.3. Declarando variáveis - Python

Python

- Python é uma linguagem dinamicamente tipada, o que significa que as variáveis não precisam de uma declaração de tipo, como em Java.
- Python, diferentemente de JS, é uma linguagem fortemente tipada, o que significa que existe uma conversão implícita de tipos. Por exemplo, o código `1 + "1"` não gera erros em linguagens fracamente tipadas, como JS, mas em Python, essa operação gera um erro.

Declaração de variáveis

```
x = 5
# ou
x:int = 5
print(x)
```

- Apesar de ser dinamicamente tipada, o Python possui um sistema de `type hints`, que permite que o programador sugira qual é o tipo que a variável deve conter. Isso não é reforçado pela linguagem, mas pode ser um lembrete útil durante a escrita de códigos mais complexos.

Declaração de constantes

```
PI:float = 3.14159
raio:float = float(input("Digite o raio: "))
perimetro:float = 2*PI*raio
print(perimetro)
```

- No Python, não existe um tipo que force o programador a manter uma variável constante, porém, por convenção no Python, o programador usa SCREAMING_SNAKE_CASE para indicar que aquela variável não deveria ser modificada.

1.2.1. Primitivos - JS

```
// Number
var myNumber = 42;

// String
var myString = "Hello, world!";

// Boolean
var myBoolean = true;

// Undefined
var myUndefined;

// Null
var myNull = null;
```

```
// Symbol
var mySymbol = Symbol("unique");

console.log("Number:", myNumber);
console.log("String:", myString);
console.log("Boolean:", myBoolean);
console.log("Undefined:", myUndefined);
console.log("Null:", myNull);
console.log("Symbol:", mySymbol);
```

- Strings em JS podem ser criadas com aspas duplas ou simples. Por convenção de outras linguagens, aspas duplas são normalmente utilizadas para textos com mais de um caractere e aspas simples para textos de um caractere.
- Todo número em JS é um `float` ou `number`. Isso é extremamente confuso.
- Os booleanos são `true` e `false`, com letras minúsculas, assim como em Java

1.2.2. Primitivos - Java

```
public class PrimitiveTypesExample {
    public static void main(String[] args) {
        byte myByte = 127;
        short myShort = 32000;
        int myInt = 2147483647;
        long myLong = 9223372036854775807L; // Note the
'L' at the end to indicate it's a long literal
        float myFloat = 3.14f; // Note the 'f' at the end
to indicate it's a float literal
        double myDouble = 2.71828;
        char myChar = 'A';
```

```
        boolean myBoolean = true;

        System.out.println("byte: " + myByte);
        System.out.println("short: " + myShort);
        System.out.println("int: " + myInt);
        System.out.println("long: " + myLong);
        System.out.println("float: " + myFloat);
        System.out.println("double: " + myDouble);
        System.out.println("char: " + myChar);
        System.out.println("boolean: " + myBoolean);
    }
}
```

- Strings não são primitivos em Java, diferentemente de JS. Em Java, strings são objetos!

1.2.3. Primitivos - Python

- Não existem primitivos em Python, apenas objetos!

```
# int
my_int = 42

# float
my_float = 3.14

# bool
my_bool = True

# str
my_str = "Hello, world!"

# NoneType
my_none = None
```

```
print("int:", my_int)
print("float:", my_float)
print("bool:", my_bool)
print("str:", my_str)
print("NoneType:", my_none)
```

1.2.4. Primitivos - Tabela

- As representações de 'primitivos' entre JS, Java e Python são muito similares de maneira geral:

Tipo Primitivo	**Java**	**JavaScript**	**Python**
int	`int myInt = 42;`	`let myInt = 42;` ou `let meuInteiro = 42;`	`my_int = 42`
float	`float myFloat = 3.14f;`	`let myFloat = 3.14;`	`my_floaat = 3.14`
boolean	`boolean myBool = true;`	`let myBool = true;`	`my_bool = True`
char	char letter = 'a';	Não aplicável	Não aplicável
string	String text = "some text";	`let myString = "Olá, mundo!";`	`myString = "Olá, mundo!"`
undefined	Não aplicável	let myUndefined;`	Não aplicável (use `None`)
null	Não aplicável	`let myNull = null;`	`meu_nenhum = None`