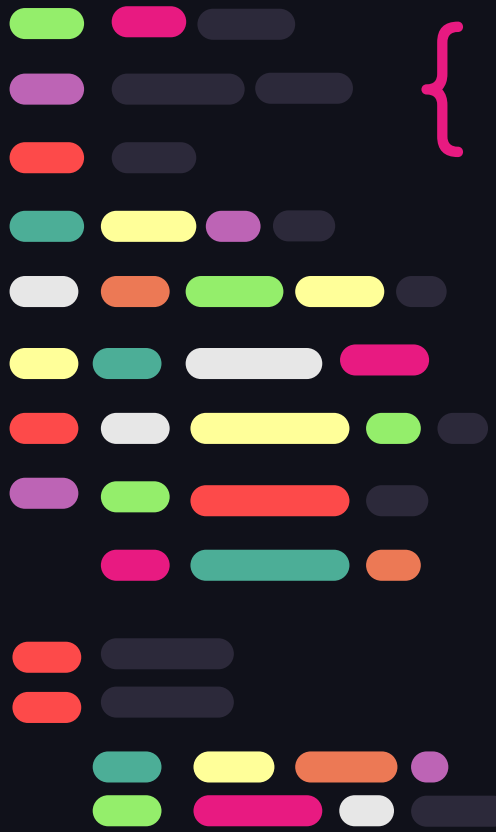


Javascript
2 - Operadores

< Por Isis =) >

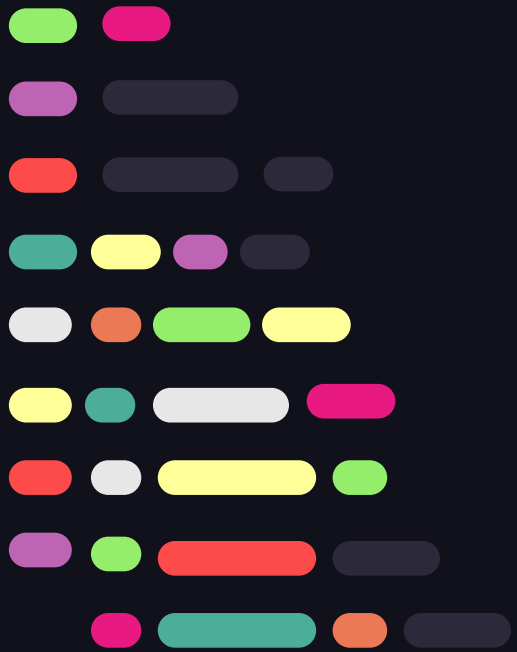
01 { ..

Operadores aritméticos

< Para realizar cálculos! Yey! >

} ..

Operadores aritméticos



- Adição (+)
- Subtração (-)
- Multiplicação (*)
- Divisão (/)
- Exponenciação (**)
- Resto da divisão (%)
- Incremento (somar 1) (++)
- Decremento (subtrair 1) (--)

Qual a ordem?

Os operadores aritméticos seguem a mesma ordem de prioridade da matemática. Multiplicações e divisões são executadas antes de somas e subtrações, por exemplo.

Para mudar a ordem de execução, utilizamos parênteses.

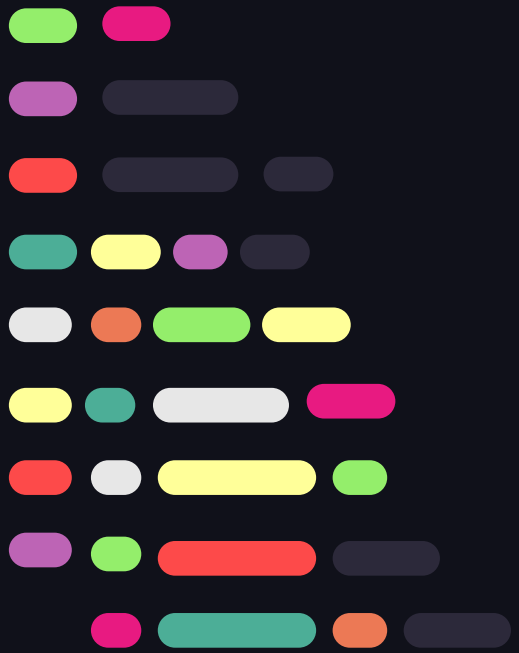
Exemplo:

```
let a = (10 + 1) * 5;
```

Neste caso, a = 55.

Você pode usar quantos parênteses quiser, mas cuidado para não deixar nenhum sem fechar.

Incremento



O incremento é um operador muito útil para contagem.

Exemplo:

```
let a = 2;

a++;
```

A passa a ser 3 após o incremento. Vamos usar bastante quando estudarmos repetição.

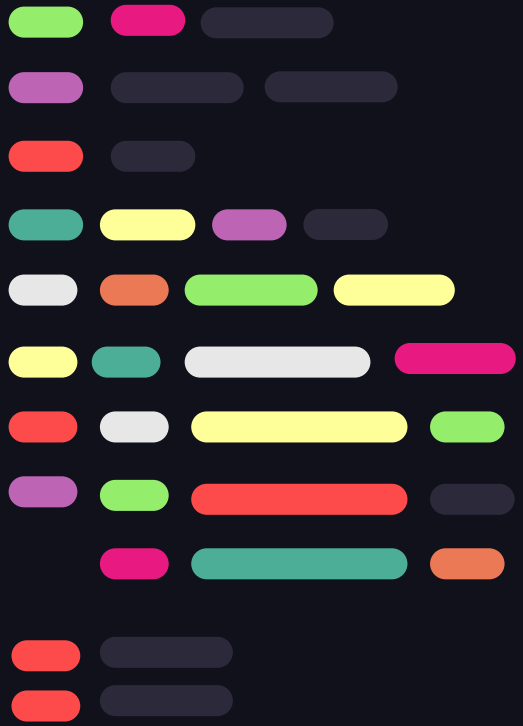
02 { ..

Operadores de atribuição

< Para atribuir valores a variáveis >

} ..

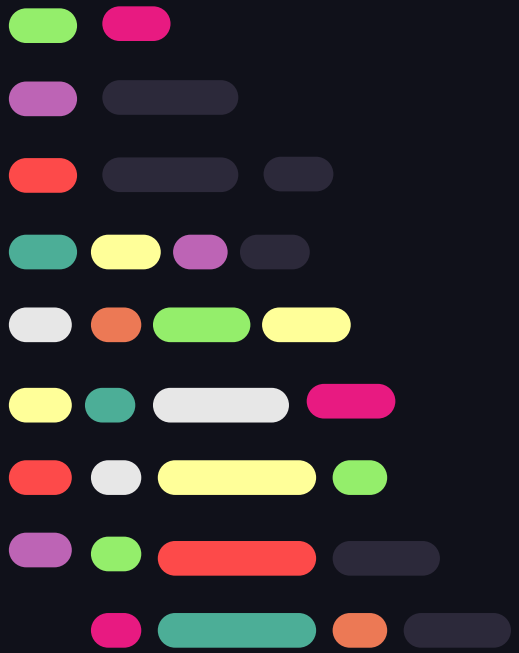
Mais de um tipo



O tipo mais comum de operador para atribuição é o `=`.

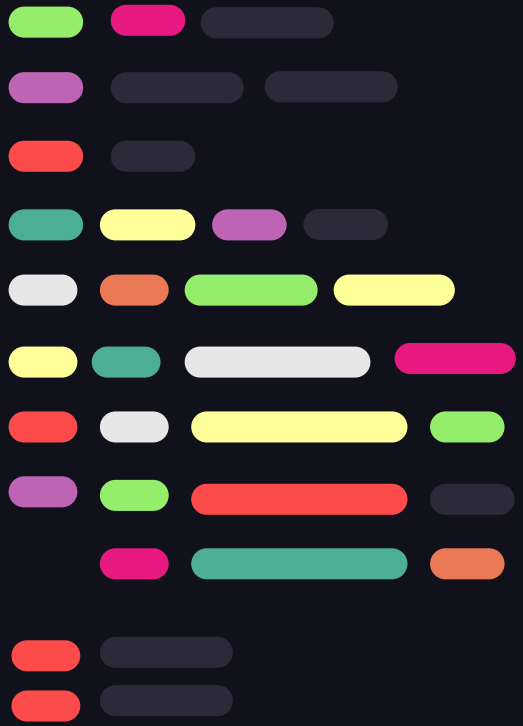
Por si só, ele é bastante suficiente, mas existem alguns outros operadores, que se combinam com as operações aritméticas.

Operadores de atribuição



= : recebe o valor
+= : soma o valor
-= : subtrai o valor
*= : multiplica o valor
/= : divide pelo valor
%= : resto da divisão pelo valor
**= : exponenciação pelo valor

Exemplo de uso



```
let a = 2;

a += 5;

A variável a passa a ser 7.

a /= 2;

A variável a passa a ser 2.5.
```

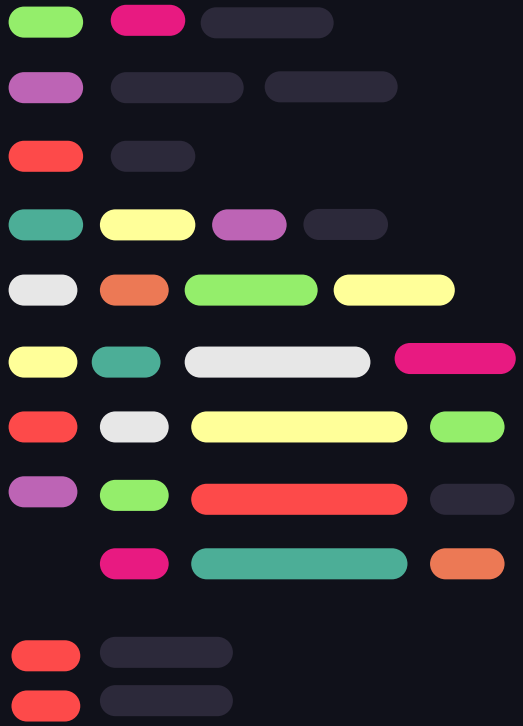
03 { ..

Operadores de comparação

< Para comparar dois elementos >

} ..

Como funcionam



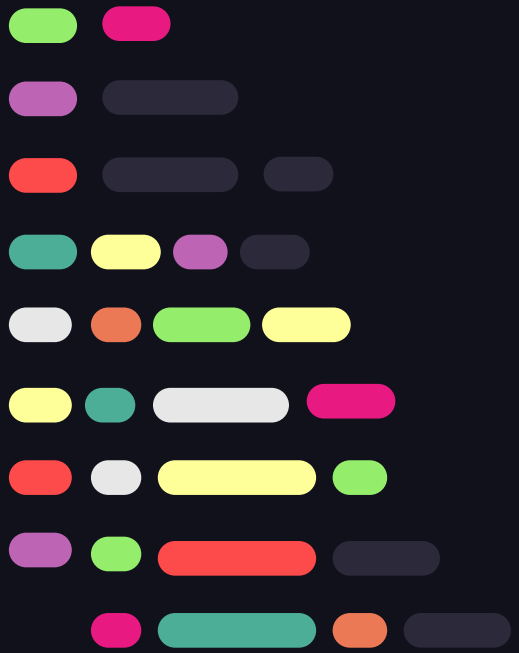
Operadores de comparação interagem entre dois dados e retornam verdadeiro se a comparação for válida ou falso do contrário.

Por exemplo:

2 é igual a 3 (falso)

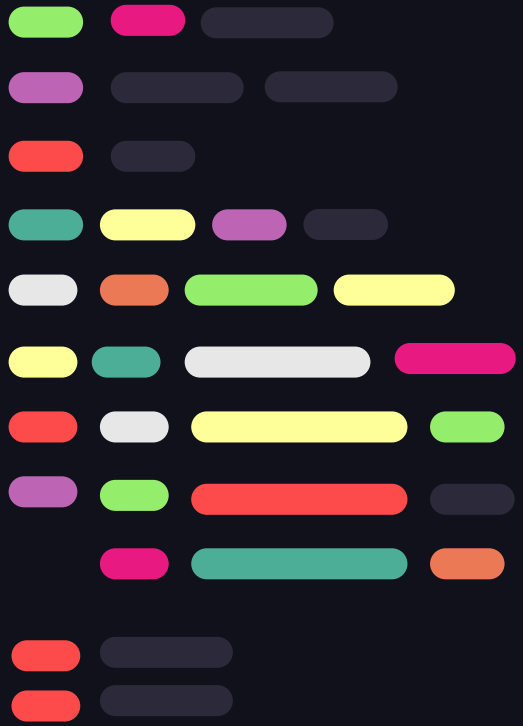
2 é menor que 3 (verdadeiro)

Operadores de comparação



- Igual a (==)
- Diferente de (!=)
- Maior que (>)
- Maior ou igual a (>=)
- Menor que (<)
- Menor ou igual a (<=)
- Igual e mesmo tipo de dado (===)
- Diferente em valor e tipo de dado (!==)

Exemplo de uso



```
let a = 2 == 3; (false)
let b = 2 == 2; (true)
let c = 2 != 3; (true)

let a = 2 < 3; (true)
let b = 2 > 3; (false)

let a = 3 < 3; (false)
let b = 3 <= 3; (true)
```

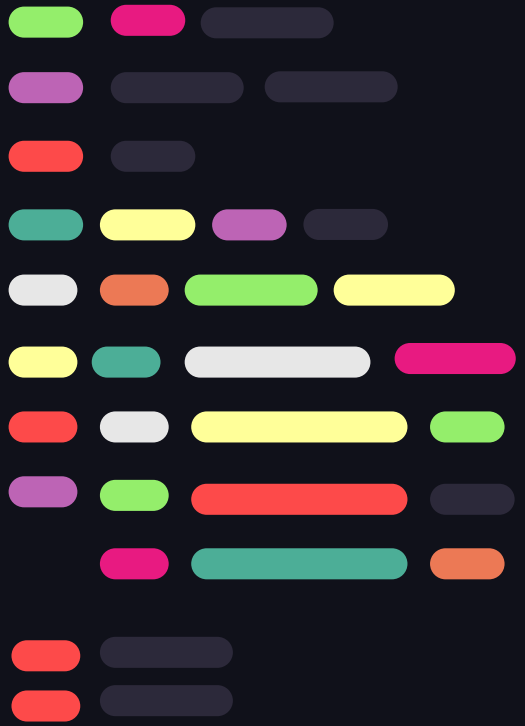
04 { ..

Operadores lógicos

< Para formular ideias mais complexas >

} ..

Como funcionam

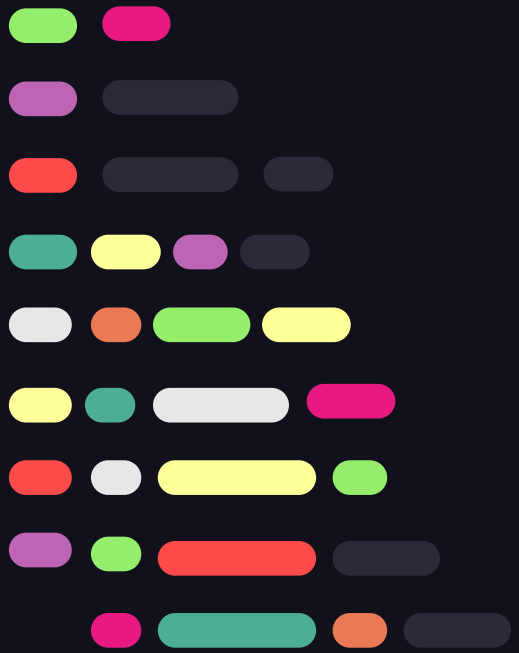


Operadores lógicos interagem entre sentenças booleanas e retornam uma nova booleana resultante.

Exemplo:

Se A for verdadeiro e B for verdadeiro, retorna verdadeiro. Se A ou B for falso, retorna falso.

Operadores lógicos



- E (&&) : verdadeiro apenas se os dois forem verdadeiros
- OU (||) : verdadeiro se ao menos um for verdadeiro
- NÃO (!) : o contrário. Verdadeiro se falso ou falso se verdadeiro. Envolve apenas uma sentença.

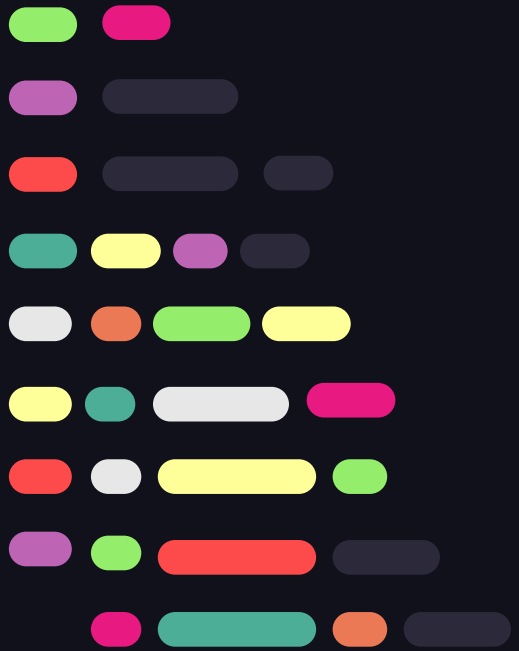
Tabela verdade E (&&)

{

	A = V	A = F
A = V	V	F
A = F	F	F

}

Exemplo



```
let a = true && true; (true)

let b = true && false; (false)

let c = false && false; (false)
```

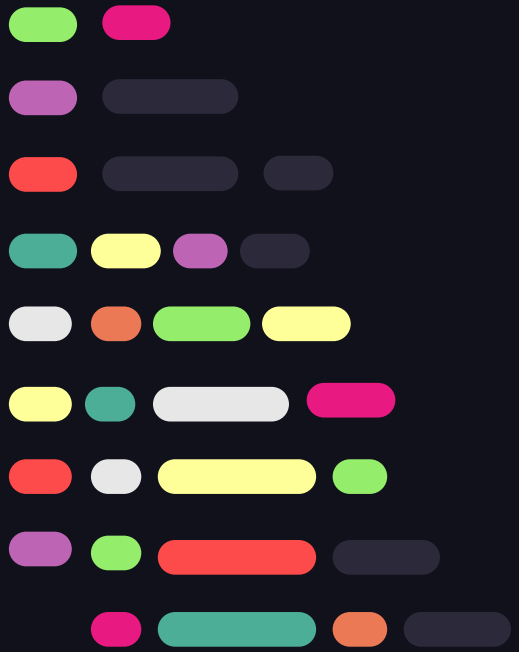
Tabela verdade OU (||)

{

	B = V	B = F
A = V	V	V
A = F	V	F

}

Exemplo



```
let a = true || true; (true)

let b = true || false; (true)

let c = false || false; (false)
```

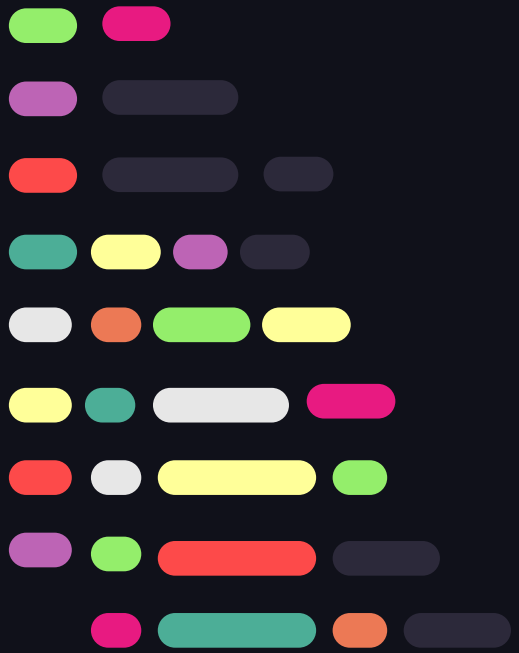
Tabela verdade NÃO(!)

{

A	!A
V	F
F	V

}

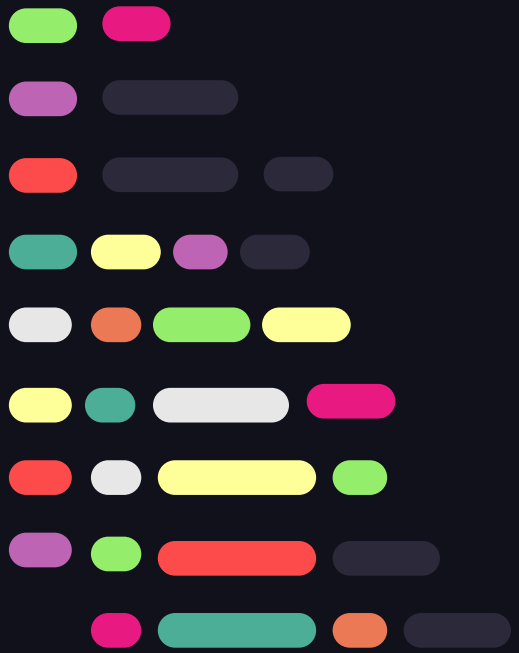
Exemplo



```
let a = !(true); (false)

let b = !(false); (true)
```

Combinando operadores

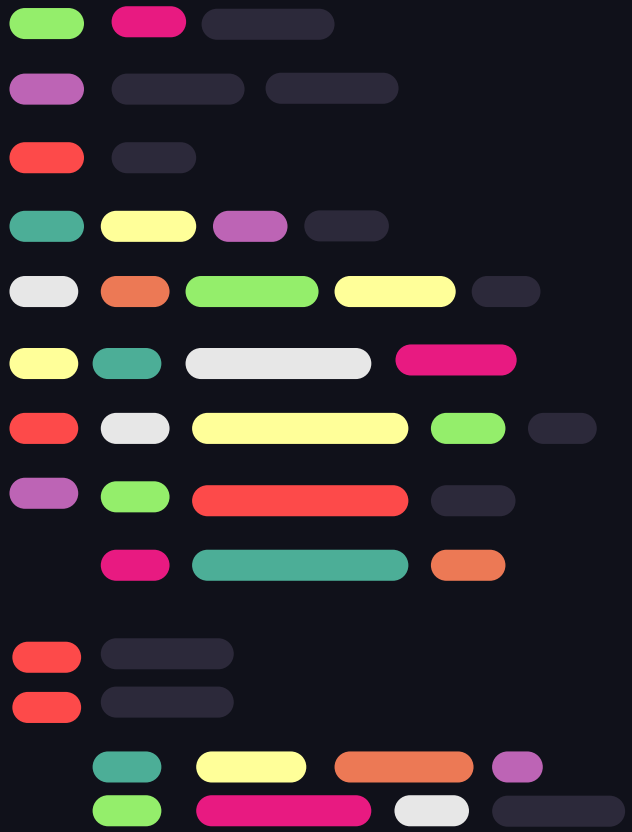


```
let a = 2, b = 3;

let c = a == 2 || a == 3; (true)

let d = a < b && a  > 1; (true)

let e = !d; (false)
```



Bom... }

< A partir daqui as coisas começam a ficar um pouco complicadas. É bom garantir que estes conceitos estão bem fixados, ok? >

*