Fundamentos do Desenvolvimento de Software

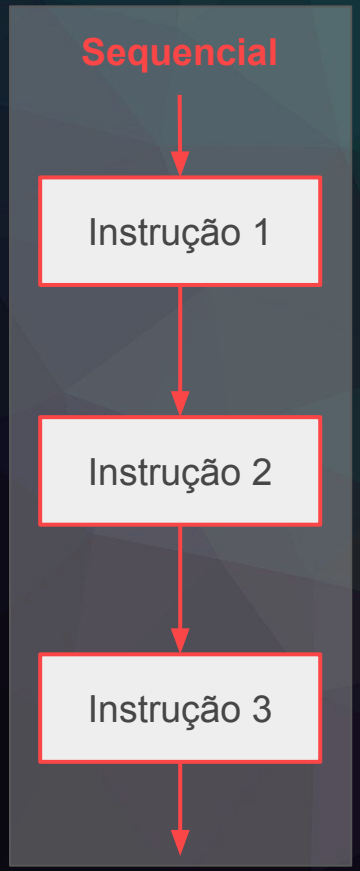
Programação Web com JavaScript I

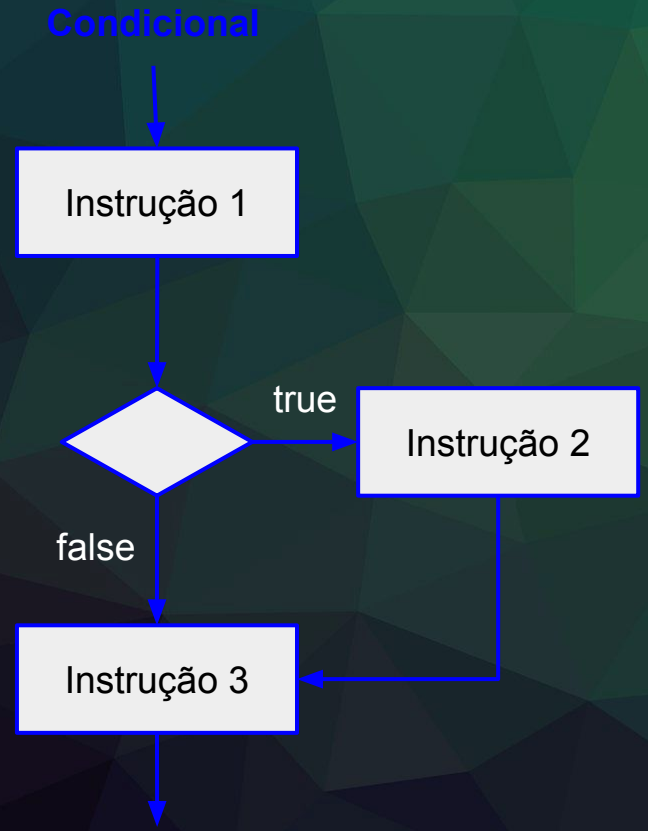
Agenda

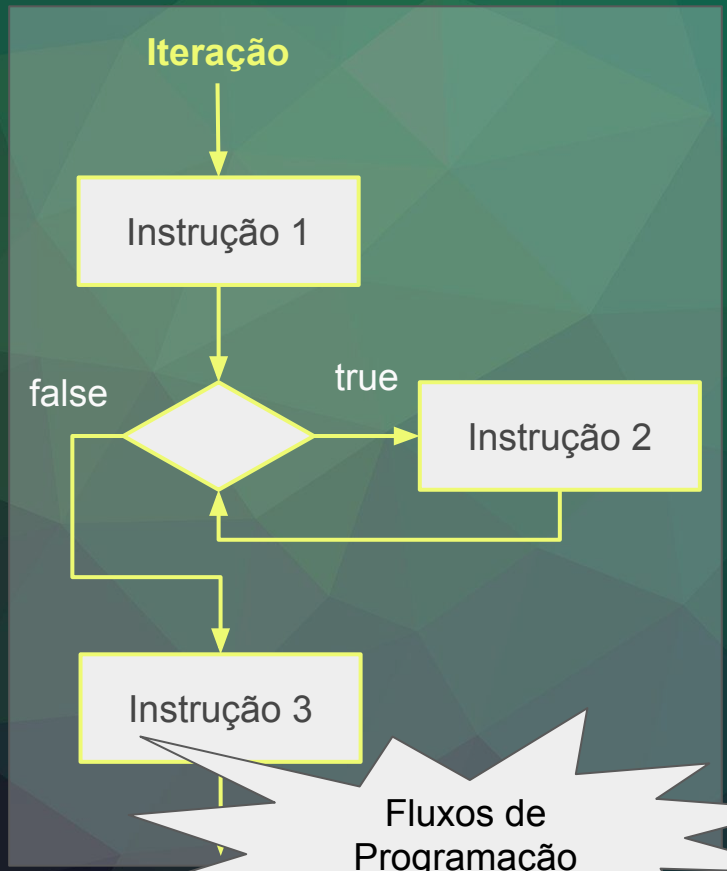
Etapa 4: Tipos de dados em Javascript.

- Estrutura de Controle Condicional.
- Operadores Relacionais.
- Praticando Códigos.

Estrutura de Controle Condicional







Fluxos de Programação

1

```
if(expressao_logica)
    1 instrução executada se a expressão for verdadeira
```

2

```
if(expressao_logica)
    1 instrução executada se a expressão for verdadeira
else
    1 instrução executada se a expressão for falsa
```

index.html | script.js 5

script.js > ...

```
1  //Entrada
2  let numero1 = Number(prompt(Digite um número));
3  let numero2 = Number(prompt(Digite outro número diferente));
4
5  //Processamento e Saída
6  if(numero1 > numero2)
7      alert(`O número ${numero1} é maior que ${numero2}`);
8  else
9      alert(`O número ${numero2} é maior que ${numero1}`);
```

3

```
if(expressao_logica) {
    instruções executadas se a expressão for verdadeira
}
```

4

```
if(expressao_logica) {
    instruções executadas se a expressão for verdadeira
} else {
    instruções executadas se a expressão for falsa
}
```

index.html | script.js

script.js > ...

```
1   //Entrada
2   let numero1 = Number(prompt("Digite um número"));
3   let numero2 = Number(prompt("Digite outro número diferente"));
4
5   //Processamento e Saída
6   if (numero1 > numero2) {
7     alert(`O número ${numero1} é maior que ${numero2}`);
8   } else {
9     alert(`O número ${numero2} é maior que ${numero1}`);
10  }
```

```
if(expressao_logica) {

    instruções executadas se a expressão for verdadeira

} else if(outra_expressao_logica) {

    instruções executadas se a expressão for falsa e
    deseja testar outra coisa

} else if(mais_uma_expressao_logica) {

    instruções executadas se as expressões anteriores
    forem falsas e deseja testar outra coisa

} else {

    instruções executadas se todas as expressões
    anteriores forem falsas

}
```

5

index.html | script.js

script.js > ...

```
1   //Entrada
2   let numero1 = Number(prompt("Digite um número"));
3   let numero2 = Number(prompt("Digite outro número"));
4
5   //Processamento e Saída
6   if (numero1 > numero2) {
7     alert(`O número ${numero1} é maior que ${numero2}`);
8   } else if (numero2 > numero1) {
9     alert(`O número ${numero2} é maior que ${numero1}`);
10  } else {
11    alert("Os números são iguais");
12  }
```

Operadores Relacionais

Um booleano ou **Boolean**, em ciência da computação, é um tipo de dado lógico que pode ter apenas um de dois valores possíveis: **verdadeiro** ou **falso**.

Expressões compostas por operadores relacionais terminam em **Boolean**.

Fluxos de execução condicionais dependem de expressões / variáveis que sejam definidos como **Boolean**.

Chamado **Boolean** em homenagem a **George Boole**, que definiu um sistema de lógica algébrica pela primeira vez na metade do século XIX.

igual → ==

igual estrito → ===

diferente → !=

diferente estrito → !==

maior que → >

maior ou igual que → >=

menor que → <

menor ou igual que → <=

Uma expressão com esses operadores têm como resultado um boolean.

O **operador de igualdade** (==) verifica se seus dois operandos são iguais, retornando um resultado booleano.

O **operador de igualdade** tenta converter e comparar operandos, mesmo que tenham tipos diferentes, ao contrário do **operador de igualdade estrita**.

```
1  console.log(1 == 1);
2  // expected output: true
3
4  console.log('hello' == 'hello');
5  // expected output: true
6
7  console.log('1' ==  1);
8  // expected output: true
9
10 console.log(0 == false);
11 // expected output: true
```

O **operador de igualdade estrita** (===) verifica se seus dois operandos são iguais, retornando um resultado booleano.

O **operador de igualdade estrita** compara os dados de forma literal: sempre considera operandos de tipos diferentes como dados diferentes.

```
1  console.log(1 === 1);
2  // expected output: true
3
4  console.log('hello' === 'hello');
5  // expected output: true
6
7  console.log('1' ===  1);
8  // expected output: false
9
10 console.log(0 === false);
11 // expected output: false
```

JavaScript tem dois conjuntos de operadores de igualdade: **===** e **!==** e seus gêmeos malignos **==** e **!=** .

Os bons funcionam da maneira que você esperaria.

Se os dois operandos forem do mesmo tipo e tiverem o mesmo valor, então **===** produz true e **!==** produz false.

Os gêmeos malignos fazem a coisa certa quando os operandos são do mesmo tipo, mas se forem de tipos diferentes, **eles tentam coagir os valores**. As regras pelas quais eles fazem isso são complicadas e esquecíveis.

Do livro “JavaScript: The Good Parts” de Douglas Crockford

x	y	==	===
undefined	undefined	true	true
null	null	true	true
true	true	true	true
false	false	true	true
'foo'	'foo'	true	true
0	0	true	true
+0	-0	true	true
+0	0	true	true
-0	0	true	true
0	false	true	false
""	false	true	false
""	0	true	false
'0'	0	true	false
'17'	17	true	false
[1, 2]	'1,2'	true	false
new String('foo')	'foo'	true	false
null	undefined	true	false

Praticando Códigos

Crie um programa que leia um número inteiro e mostre se o mesmo é positivo, negativo ou zero.

Crie um programa que leia um número inteiro e mostre se o mesmo é par ou ímpar.