# Control flow or logic flow in Java Script

We don't want the full code to alway run. So we make specific conditions that if that is true only then run the code.

```
if (condition) {
}
```

The code in if condition only get execute when condition is true.
you can also directly pass true or false.

now there are some operators to check conditions→
<, >, <=, >=, ==, !=, ===, !==

↓ ↓

This check the type as well

now there is else as well, it is used when you want something to execute when if condition is false.

```
const temprature = 41
if (temprature < 50){
    console.log("less than 50");
}else {
 console.log("temprature is greater than 50");
}
```

you can do the same using implicit scope but never prefer that because that make a messy code and is not a good practice →

```
const balance = 1000
if (balance > 500) console.log("test");
```

in the implicit scode the scope is justified using ";". You can add multiple lines as well.

```
if (balance > 500) console.log("test"),
console.log("test2");
```

Remember that never use this approach.

you can also check multiple conditions using else if →

```
const balance = 1000
if (balance < 500) {
        console.log(" less than 500");
}
else if (balance < 900) {
        console.log(" less than 900");
}
else {
   console.log(" less than 1200");
}
```

To check multiple conditions in one if condition →

```
const userLoggedIn = true
const debitCard = true

if (userLoggedIn && debitCard) {
        console.log("Allow to buy course")
}
```

&& → And

|| → or

## Switch conditional statement

Syntaxc →

```
switch (key) {
    case value:

        break;

    default:
        break;
}
```

eg →

```
const month = 3
switch (month) {
    case 1:
        console.log("January");
        break;
    case 2:
        console.log("Feburary");
        break;
    case 3:
        console.log("March");
        break;
    default:
        console.log("Please enter a valid month");
        break;
}
```

## Truthy and falsy

Till now we have been checking strictly that the value is true or false. There is another way of doing the same by using truthy or falsy values where just having value will mean true or false.

```
const userEmail = "v@vasu.ai"

if (userEmail) {
    console.log("Got user email");
}
else {
    console.log("Don't have user Email");
}
```

Output → Got user email

falsy values → false, 0, -0, BigInt 0n, "", null, undefined, NaN

Every thing except these are truthy values.

Some truth values → "0", "false", " " (string with space), [], {}, function() {} (Empty Function)

## To check if we have empty object or array →

(i)
```
const userEmail = []
if (userEmail.length === 0){
}
```

(ii)
```
const emptyObj = {}
if (Object.keys(emptyObj).length === 0) {
}
```

## Nullish Coalescing operator (??)

This operator is used to handle undefined and null values and is widely used when working with database. especially when you don't want any null value.

(i)
```
let val1;
val1 = 5?? 10
```

Output → 5

If there is no null value it's default behaviour is it will take the first value.

(ii)
```
let val1;
val1 = null ?? 10
```

Output → 10

(iii) val1 = undefined ?? 10

Output → 10

(iv) val1 = null ?? 10 ?? 20

Output → 10

Ternary Operator

Syntax → condition ? true : false

```
const iceTeaPrice = 100
iceTeaPrice >= 80 ? console.log("less than 80"):
        console.log("greater than 80")
```

Don't confuse it with nullish coalescing operator.