Module 01

Purwadhika
Digital Technology School

Full Stack Web Development

Conditional and loop statements

Job Connector Program

Outline

- Conditional statements
- Loop statements

Conditional statements

What are conditional statements?

Conditional statements are code expressions used to **tell the computer, which block of code to execute**.

In other words, conditional statements **determine the flow of your computer program**.

If statement

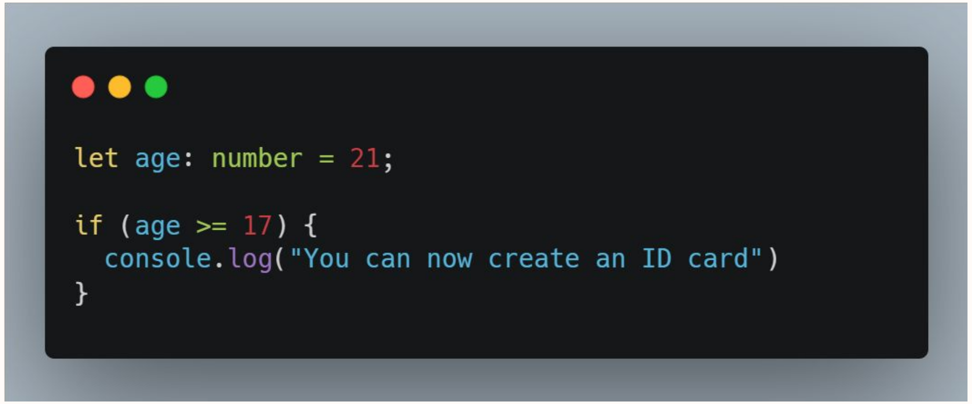
If statement are the basic foundation of conditional statements.

It takes a **condition** (which should result in a boolean), and **a block of code that executes when the condition's result is true**.

If statement

In this example, we have a variable called ***age*** which has a value of **21**. Below that, we have an ***if statement*** and a block of code right after it.

The condition of the ***if statement*** tells us that, **the variable *age* should be greater than equals to 17**, for the block of code to be executed.

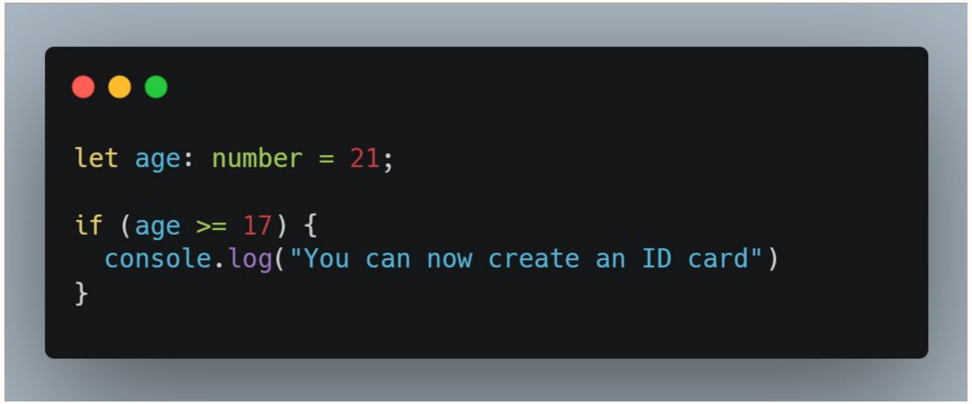


```
let age: number = 21;

if (age >= 17) {
  console.log("You can now create an ID card")
}
```

If statement

Now, since ***age*** has a value of **21** which is clearly greater than **17**, the condition will result in a **true (boolean)** and the block of code will now be executed.

So the code below will have an output of: **“You can now create an ID Card”**



```
let age: number = 21;

if (age >= 17) {
  console.log("You can now create an ID card")
}
```

Else statement

Now what happens **if *age* does not meet the required condition** ? What **if *age* is less than 17** ?

Surely, the code below will have no output. But it would be better if we were to give some kind of message to the user that their age isn't eligible. For this, we need **else statement**.

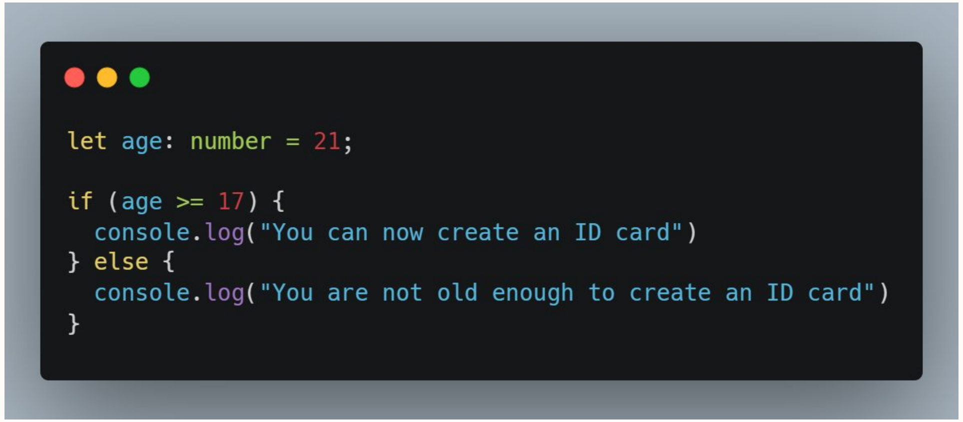
Else statement

An else statement will act as a **backup plan for if statements**. It does not require a condition, it only needs a block of code to execute.

The block of code of an else statement executes **when the condition of an if statement does not meet its requirement**.

Else statement

As you can see, we've added an else statement to the example below. This way, when ***age* is NOT greater than equals to 17**, the else statement's block of code will be executed. So the output of the code below will now be **"You are not old enough to create an ID card"**



```
let age: number = 21;

if (age >= 17) {
  console.log("You can now create an ID card")
} else {
  console.log("You are not old enough to create an ID card")
}
```

Else if statement

Now let’s take a look at a different case, let’s say we want to make a program to check if a student has a passing grade. In that case, **we’re gonna need to have multiple conditions**.

But currently, we can only make 2 possible outcomes. **This is where *else if statements* come to play**.

Else if statement

An ***else if statement*** is basically an ***if statement* combined with an *else statement***.

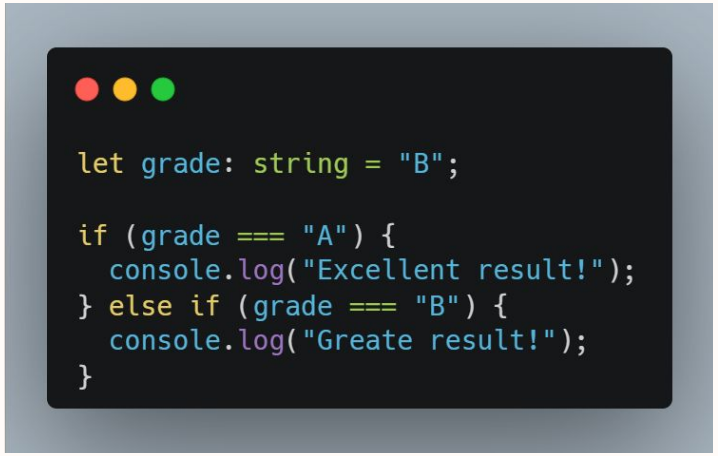
It will act as a **backup plan for an if statement, however it will also need a condition to be fulfilled**.

Else if statement

Take a look at the code, ***grade*** has a value of **"B"** which means **it will not meet the condition of the *if statement***.

Because the *if statement*'s code block isn't executed, it will **continue to check the condition of the *else if statement***.

The condition of the ***else if statement*** will result in a true (boolean). This means that **the code block will be executed**, and the output of the code will be: **"Great Result!"**

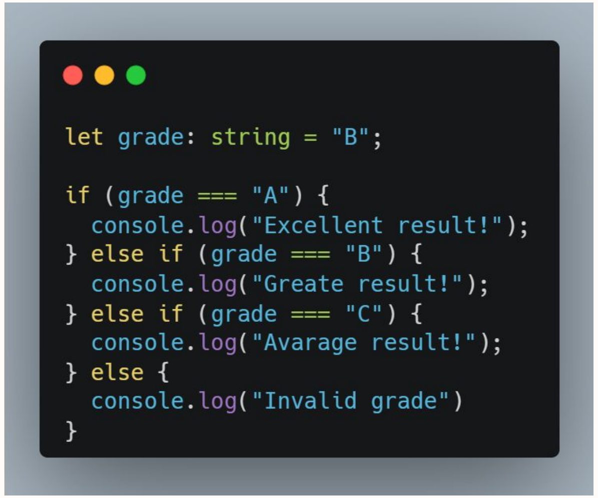


```
let grade: string = "B";

if (grade === "A") {
  console.log("Excellent result!");
} else if (grade === "B") {
  console.log("Greate result!");
}
```

Chaining conditions

We can also **chain together several *else if statements*** to create even more possible outcomes.

We can also **add an *else statement*** to handle cases where grade has a value of other than “A”, “B”, or “C”.



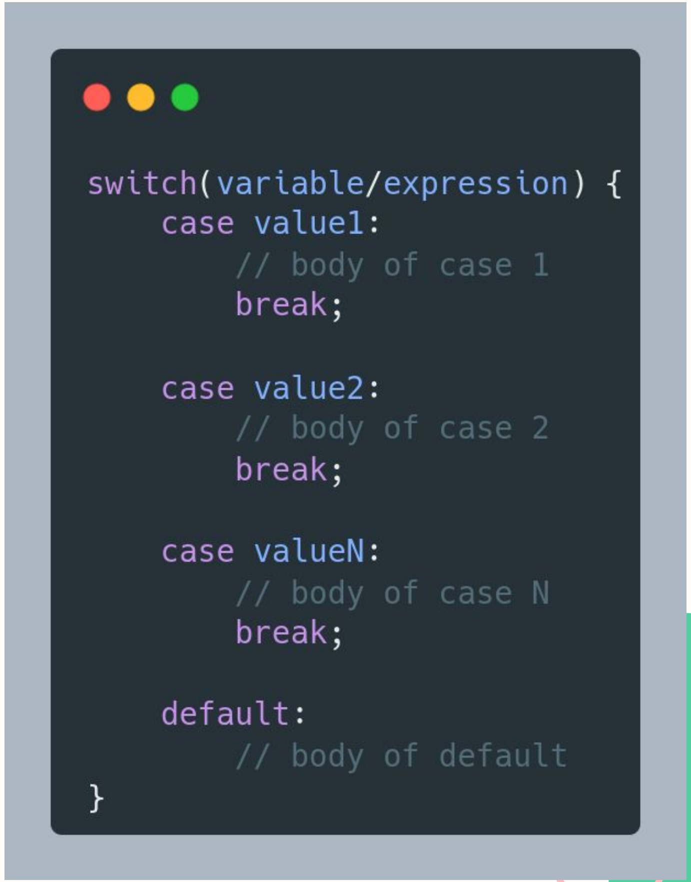
```
let grade: string = "B";

if (grade === "A") {
  console.log("Excellent result!");
} else if (grade === "B") {
  console.log("Greate result!");
} else if (grade === "C") {
  console.log("Avarage result!");
} else {
  console.log("Invalid grade")
}
```

Switch Case

The JavaScript switch statement is used in decision making.

The switch statement evaluates an expression and executes the corresponding body that matches the expression's result.

The syntax of the switch statement is:



```
switch(variable/expression) {
    case value1:
        // body of case 1
        break;

    case value2:
        // body of case 2
        break;

    case valueN:
        // body of case N
        break;

    default:
        // body of default
}
```

Switch Case

The switch statement evaluates a variable/expression inside parentheses ().

If the result of the expression is equal to value1:

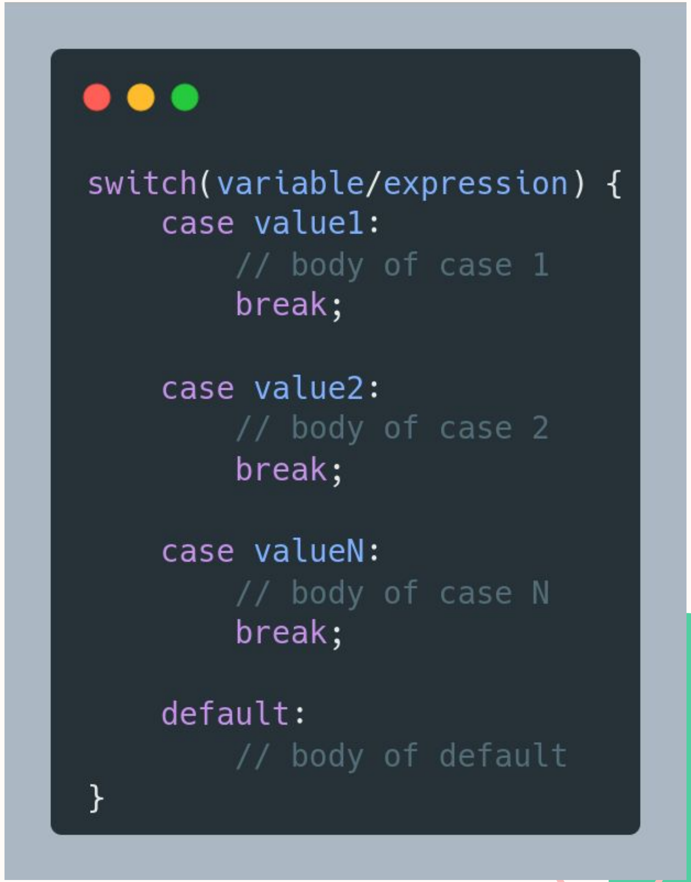
its body of case 1 is executed.

If the result of the expression is equal to value2:

its body of case 2 is executed.

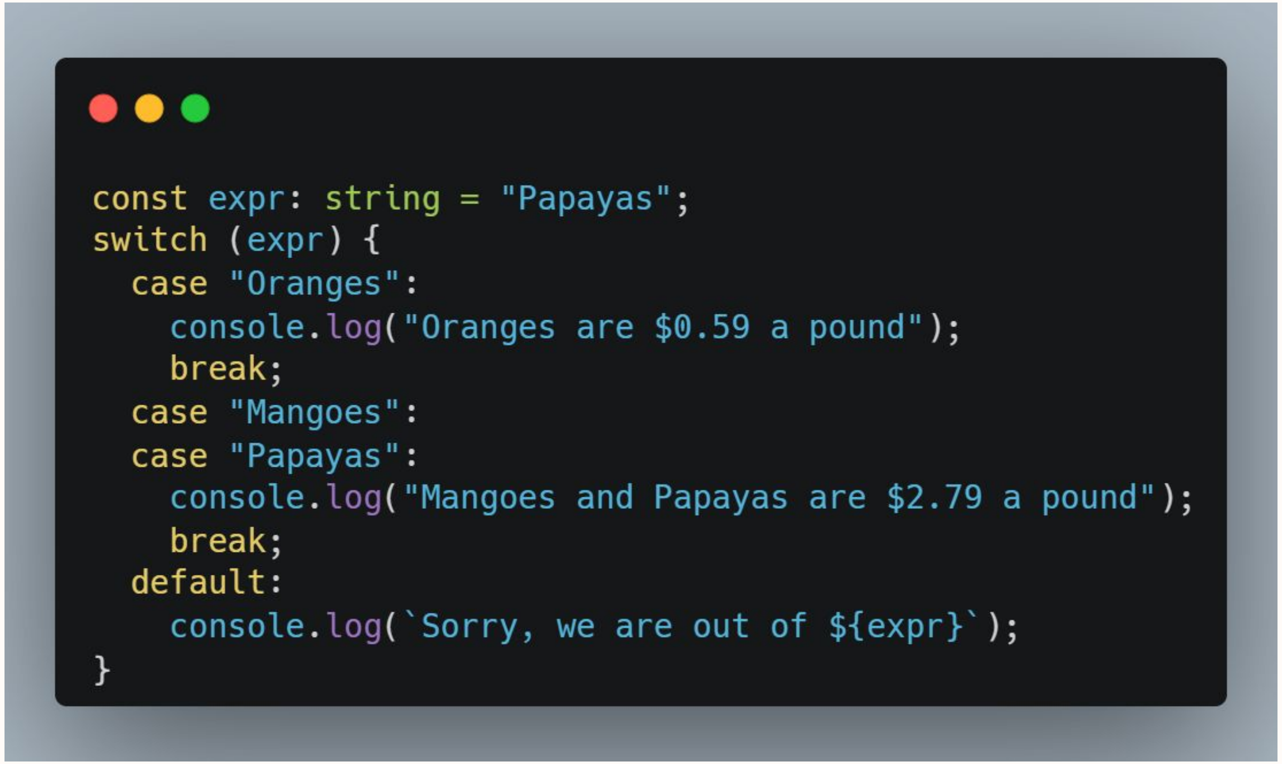
This process goes on. If there is no matching case,

the default body executes.



```
switch(variable/expression) {
    case value1:
        // body of case 1
        break;

    case value2:
        // body of case 2
        break;

    case valueN:
        // body of case N
        break;

    default:
        // body of default
}
```

Switch Case - Example



```
const expr: string = "Papayas";
switch (expr) {
  case "Oranges":
    console.log("Oranges are $0.59 a pound");
    break;
  case "Mangoes":
  case "Papayas":
    console.log("Mangoes and Papayas are $2.79 a pound");
    break;
  default:
    console.log(`Sorry, we are out of ${expr}`);
}
```

Difference Between if else and Switch Statement

Parameter	If-else	Switch
Definition	The if and else blocks are executed depending on the condition in the if statement	The switch statement has multiple cases, and the code block corresponding to that case is executed
Evaluation	Used for integer, character, pointer, floating-point type, or Boolean type.	Used for character expressions and integers.
Testing	Tests both logical expressions and equality	Tests only equality
Expression	Multiple statements for multiple decisions	Single statements for multiple decisions

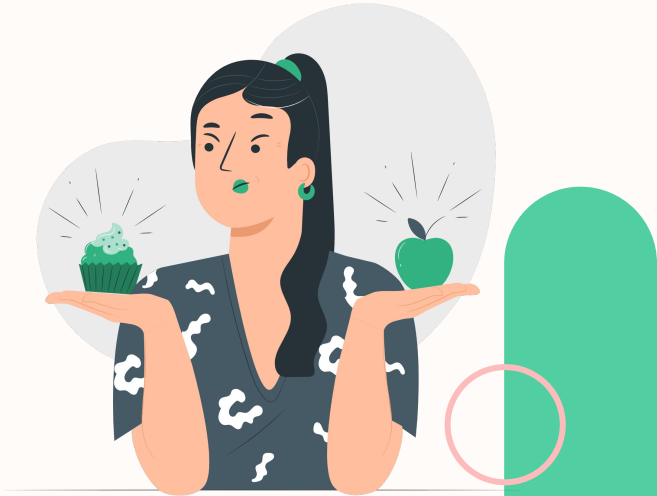
Difference Between if else and Switch Statement

Parameter	If-else	Switch
Default Execution	If the condition inside the if-statement is false, then the code block under the else condition is executed	If the condition inside switch statements does not match any of the cases, the default statement is executed.
Editing	Difficult to edit nested if-else statements.	Easy to edit.
Values	Based on constraint	Based on user

Truthy and falsy values

Falsy and truthy are terms used in programming to **determine values within a boolean context**.

For example in a boolean context, **1** is considered true which means **1 is a truthy value**. **0** in a boolean context is considered false which means **it is a falsy value**.



Truthy and falsy values

At first glance, this seem quite simple but Javascript can sometimes become confusing.

Here is a list of some falsy and truthy values that can sometimes be confusing.

If you're not sure whether a value is truthy or falsy, you can use an *if statement* and input your value as the condition.

If your value is truthy, then surely the if statement should execute the code in the block.

Falsy

- "" (empty string)
- 0
- null
- undefined
- NaN

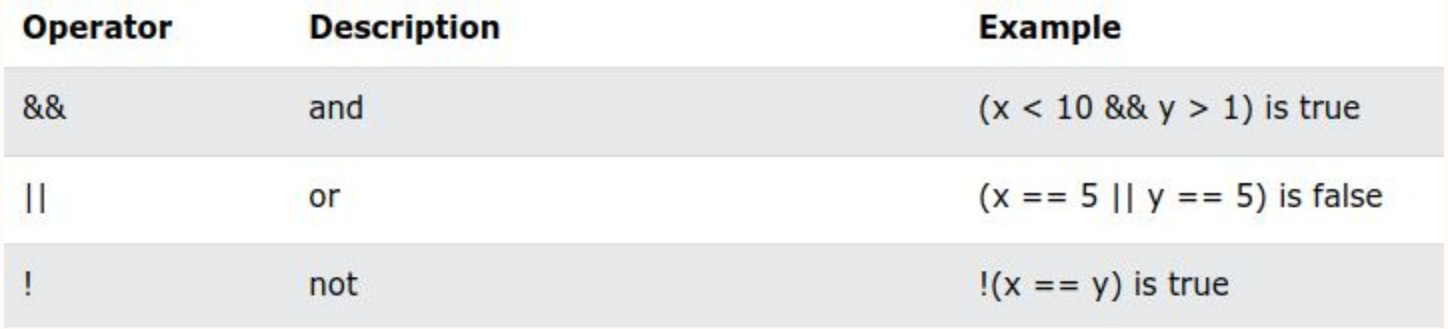
Truthy

- " " (blank character string)
- [] (empty array)
- {} (empty object)
- 1
- "1" (string)
- "0" (string)
- "false" (string)
- "true" (string)

Logical Operators

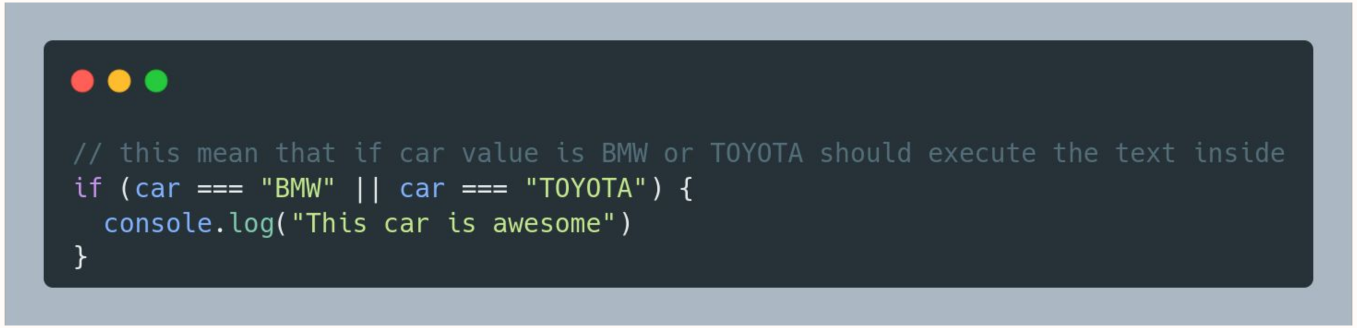
Logical operators are used to determine the logic between variables or values.

Given that x = 6 and y = 3, the table below explains the logical operators:



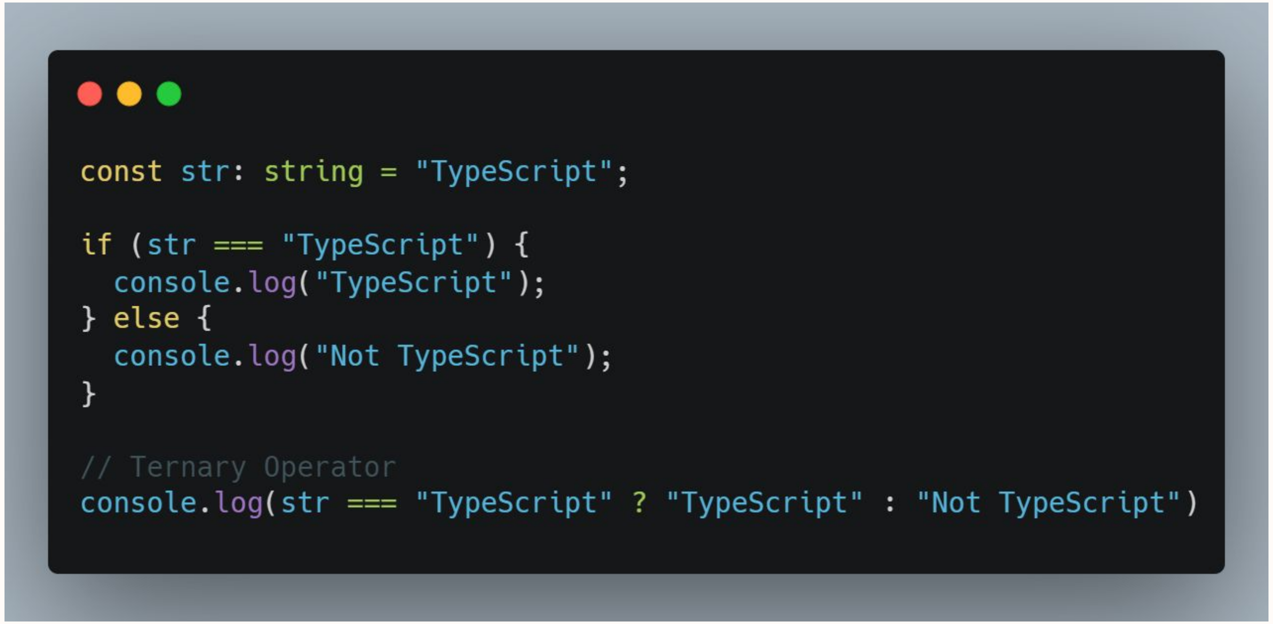
Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

Logical Operators Example



```javascript
// this mean that if car value is BMW or TOYOTA should execute the text inside
if (car === "BMW" || car === "TOYOTA") {
  console.log("This car is awesome")
}
```

Ternary Operator



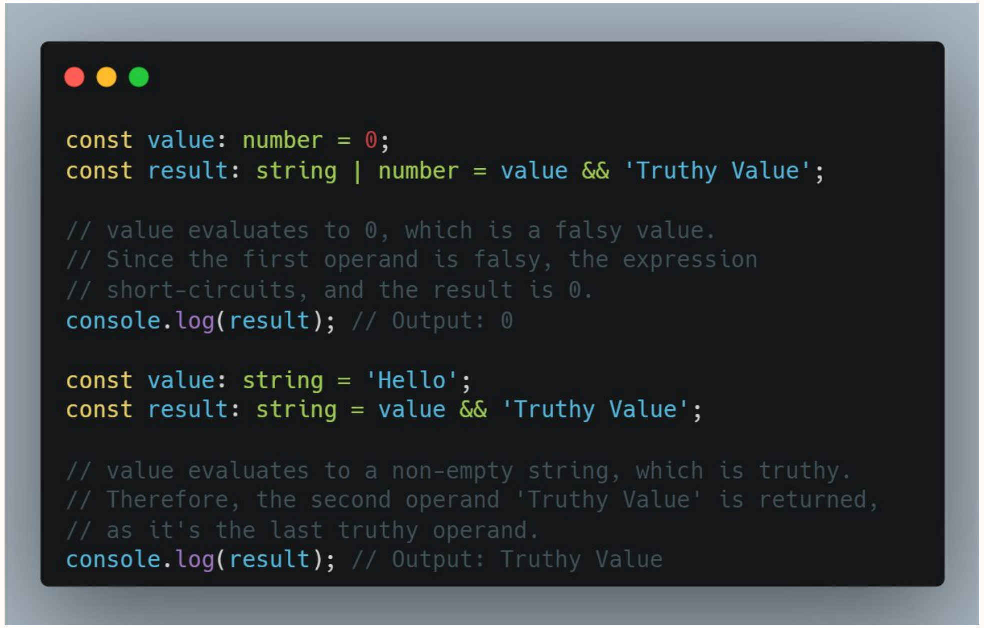
```
const str: string = "TypeScript";

if (str === "TypeScript") {
  console.log("TypeScript");
} else {
  console.log("Not TypeScript");
}

// Ternary Operator
console.log(str === "TypeScript" ? "TypeScript" : "Not TypeScript")
```

Short-Circuiting

Short-circuiting is a behavior exhibited by logical operators (&&, ||) where the evaluation of the second operand is skipped if the outcome can be determined by evaluating the first operand alone.

The && Operator

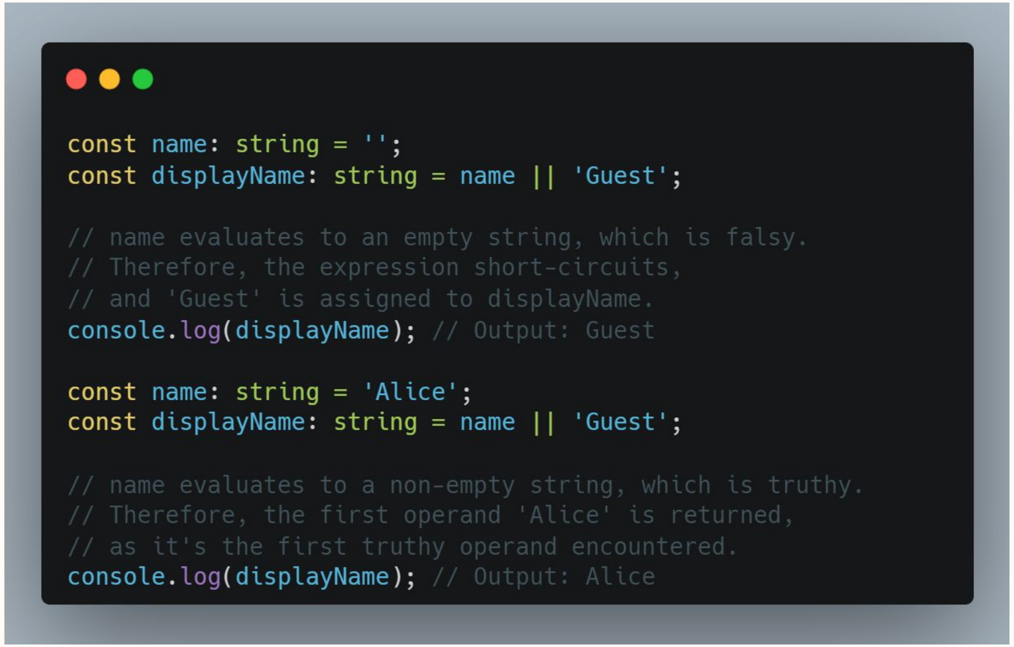
The && operator returns the first falsy operand, or the last truthy operand if all operands are truthy.



```
const value: number = 0;
const result: string | number = value && 'Truthy Value';

// value evaluates to 0, which is a falsy value.
// Since the first operand is falsy, the expression
// short-circuits, and the result is 0.
console.log(result); // Output: 0

const value: string = 'Hello';
const result: string = value && 'Truthy Value';

// value evaluates to a non-empty string, which is truthy.
// Therefore, the second operand 'Truthy Value' is returned,
// as it's the last truthy operand.
console.log(result); // Output: Truthy Value
```

The || Operator

The || operator returns the first truthy operand, or the last falsy operand if all operands are falsy.



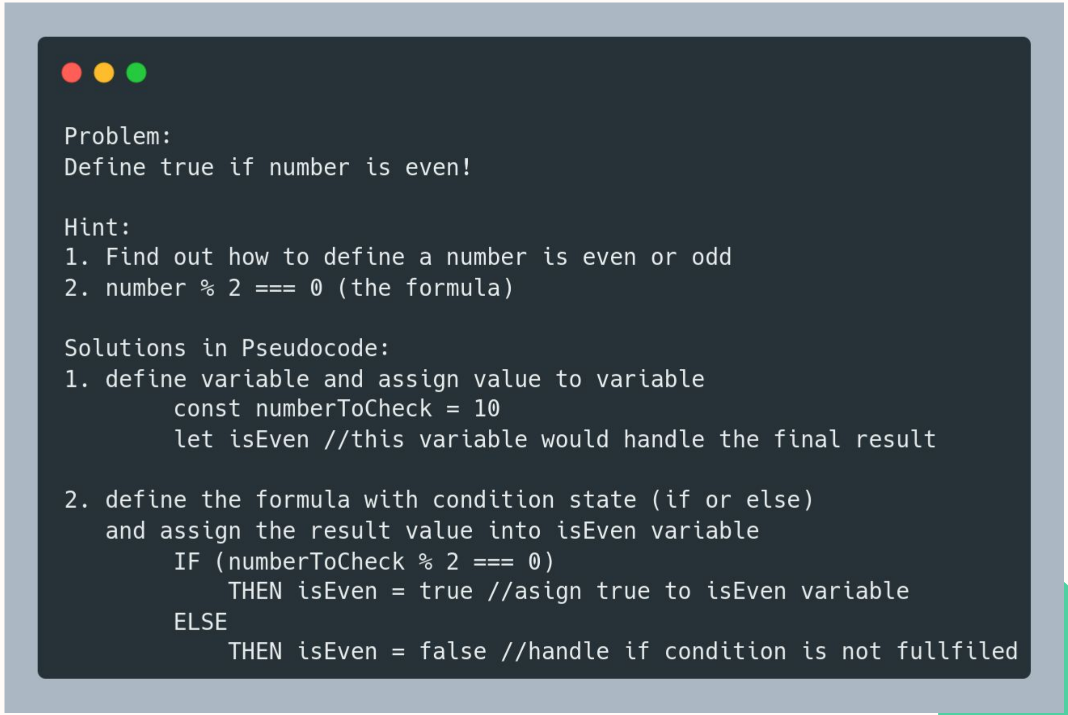
```typescript
const name: string = '';
const displayName: string = name || 'Guest';

// name evaluates to an empty string, which is falsy.
// Therefore, the expression short-circuits,
// and 'Guest' is assigned to displayName.
console.log(displayName); // Output: Guest

const name: string = 'Alice';
const displayName: string = name || 'Guest';

// name evaluates to a non-empty string, which is truthy.
// Therefore, the first operand 'Alice' is returned,
// as it's the first truthy operand encountered.
console.log(displayName); // Output: Alice
```

Pseudocode in Conditional Statement

Since you have learn about pseudocode in the last session. Here is example of pseudocode implemented in conditional statement.

Remember, pseudocode will help you to solve a problem with easier approach!

Don’t forget to convert this pseudocode into a programming code!



```
Problem:
Define true if number is even!

Hint:
1. Find out how to define a number is even or odd
2. number % 2 === 0 (the formula)

Solutions in Pseudocode:
1. define variable and assign value to variable
        const numberToCheck = 10
        let isEven //this variable would handle the final result

2. define the formula with condition state (if or else)
   and assign the result value into isEven variable
        IF (numberToCheck % 2 === 0)
            THEN isEven = true //asign true to isEven variable
        ELSE
            THEN isEven = false //handle if condition is not fullfiled
```

Loop statements

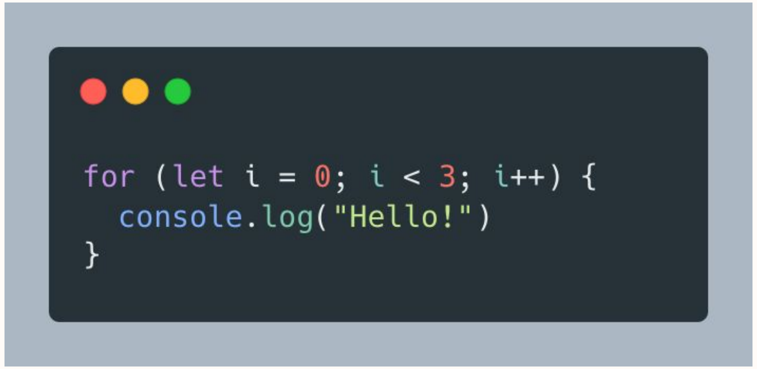
What are **loops**?

In simple terms, loops are a **sequence of commands or instructions that is repeatedly executed** until a certain condition is met.

For loop

A ***for loop*** consists of 3 statements in its conditions.

- The first statement is executed once before the execution of the code block, to initialize the iteration variable.
- The second statement defines the condition for executing the code block.
- The third statement is executed every time after the execution of the code block.



```
for (let i = 0; i < 3; i++) {
  console.log("Hello!")
}
```

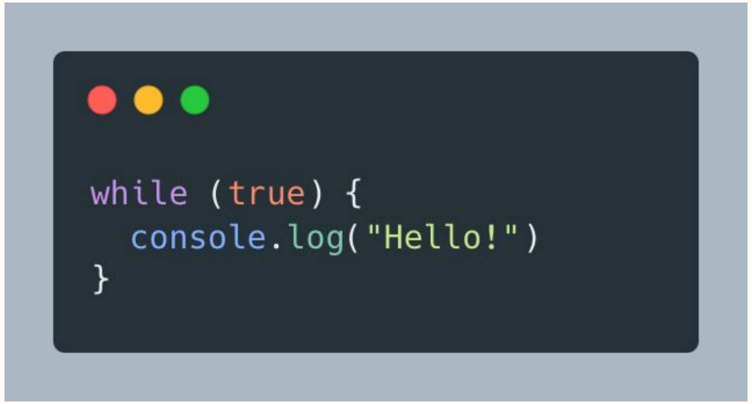
While loop

- While loops are basically ***if conditions* that are repeated.**
- As long as the condition is **true**, the loop will continue.

While loop

This loop will result in an infinite loop. Which means the loop will never stop.

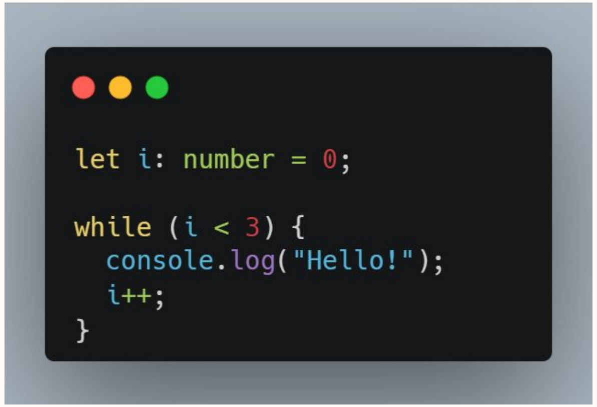
Keep in mind that when using loops, we should always set a condition so that the loop will eventually break/stop.



```
while (true) {
  console.log("Hello!")
}
```

While loop

This is how you should make a ***while loop statement***. In every iteration, the ***i*** variable will be incremented, therefore the condition will eventually result in a ***false*** boolean



```
let i: number = 0;

while (i < 3) {
  console.log("Hello!");
  i++;
}
```

Do … While loop

Do while loops are very similar to while loops.

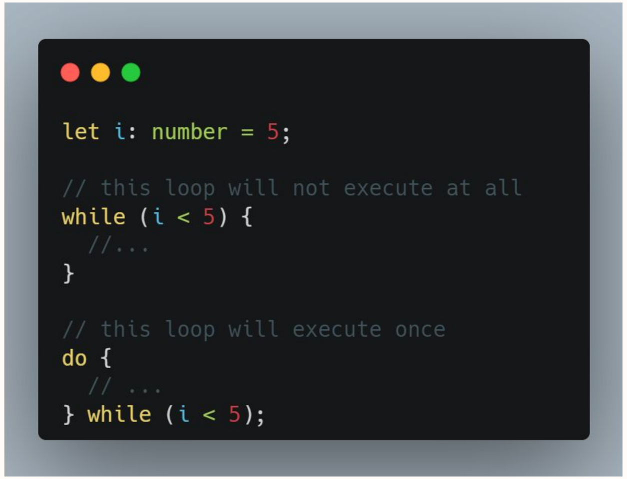
The only difference it has is that it only starts **checking the condition after the first code block execution.**

Do ... While loop

In this example, the **i** variable already has a value of **5**.

The while loop will not execute since the condition is checked before the code block execution, and the condition itself results in a false value.

However the do while loop will execute at least once, because the condition is checked only after the first code block execution

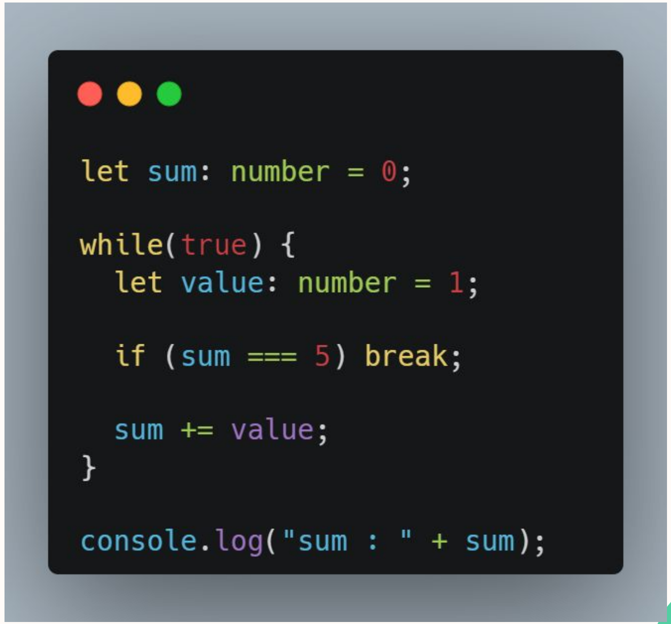


```
let i: number = 5;

// this loop will not execute at all
while (i < 5) {
  //...
}

// this loop will execute once
do {
  // ...
} while (i < 5);
```

Break

Normally, a loop exits when its condition becomes **falsy**. But we can force the exit at any time using the special **break** directive.

In this code, the loop will stop when the value of **sum** is 5.

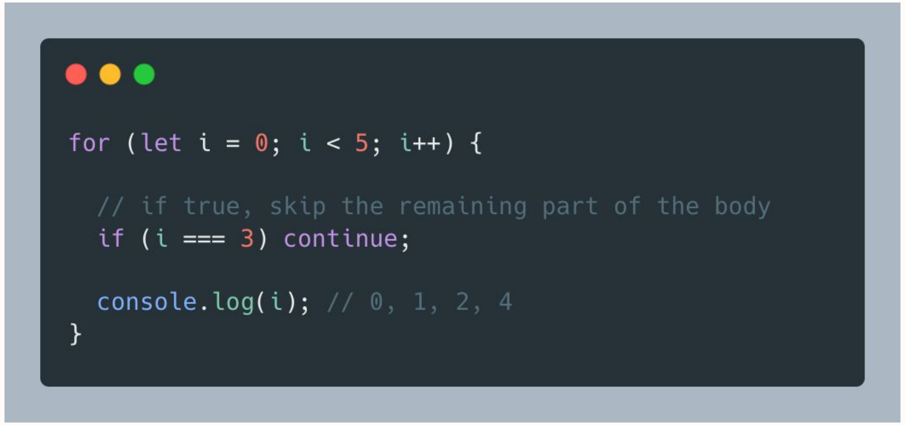


```
let sum: number = 0;

while(true) {
   let value: number = 1;

   if (sum === 5) break;

   sum += value;
}

console.log("sum : " + sum);
```

Continue

The continue directive is a "**lighter version**" of break. It doesn't stop the whole loop. Instead, it **stops the current iteration and forces the loop to start a new one** (if the condition allows).

We can use it if we're done with the current iteration and would like to move on to the next one.

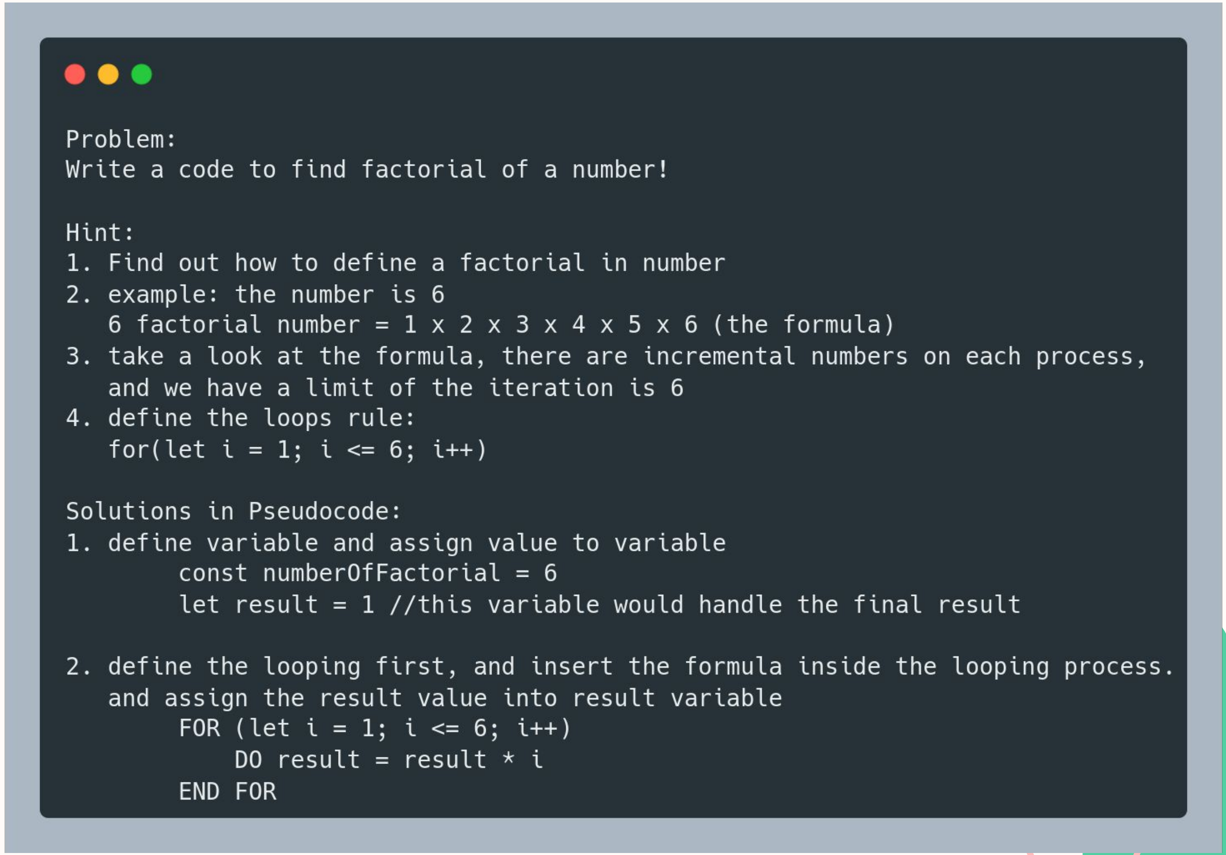


```
for (let i = 0; i < 5; i++) {

  // if true, skip the remaining part of the body
  if (i === 3) continue;

  console.log(i); // 0, 1, 2, 4
}
```

Pseudocode in Looping Statement

Check out this pseudocode in order to solve a problem that needs to implement looping statement.

Try to solve this problem with another looping such as WHILE!



```
Problem:
Write a code to find factorial of a number!

Hint:
1. Find out how to define a factorial in number
2. example: the number is 6
   6 factorial number = 1 x 2 x 3 x 4 x 5 x 6 (the formula)
3. take a look at the formula, there are incremental numbers on each process,
   and we have a limit of the iteration is 6
4. define the loops rule:
   for(let i = 1; i <= 6; i++)

Solutions in Pseudocode:
1. define variable and assign value to variable
       const numberOfFactorial = 6
       let result = 1 //this variable would handle the final result

2. define the looping first, and insert the formula inside the looping process.
   and assign the result value into result variable
       FOR (let i = 1; i <= 6; i++)
           DO result = result * i
       END FOR
```

Exercise

- Write a code to check whether the number is odd or even
  - Example: 25 → odd number, 2 → even number
- Write a code to check whether the number is prime number or not
  - Example: 7 → 7 is a prime number
  - Example: 6 → 6 is not a prime number
- Write a code to find the sum of the numbers 1 to N
  - Example: 5 → 1 + 2 + 3 + 4 + 5 = 15
  - Example: 3 → 1 + 2 + 3 = 6
- Write a code to find factorial of a number
  - Example: 4! → 4 x 3 x 2 x 1 = 24
  - Example: 6! → 6 x 5 x 4 x 3 x 2 x 1 = 720
- Write a code to print the first N fibonacci numbers
  - Example: 15 → 610

Thank You!