Control Flow

By Gajanan Kharat - CODEMIND Technology

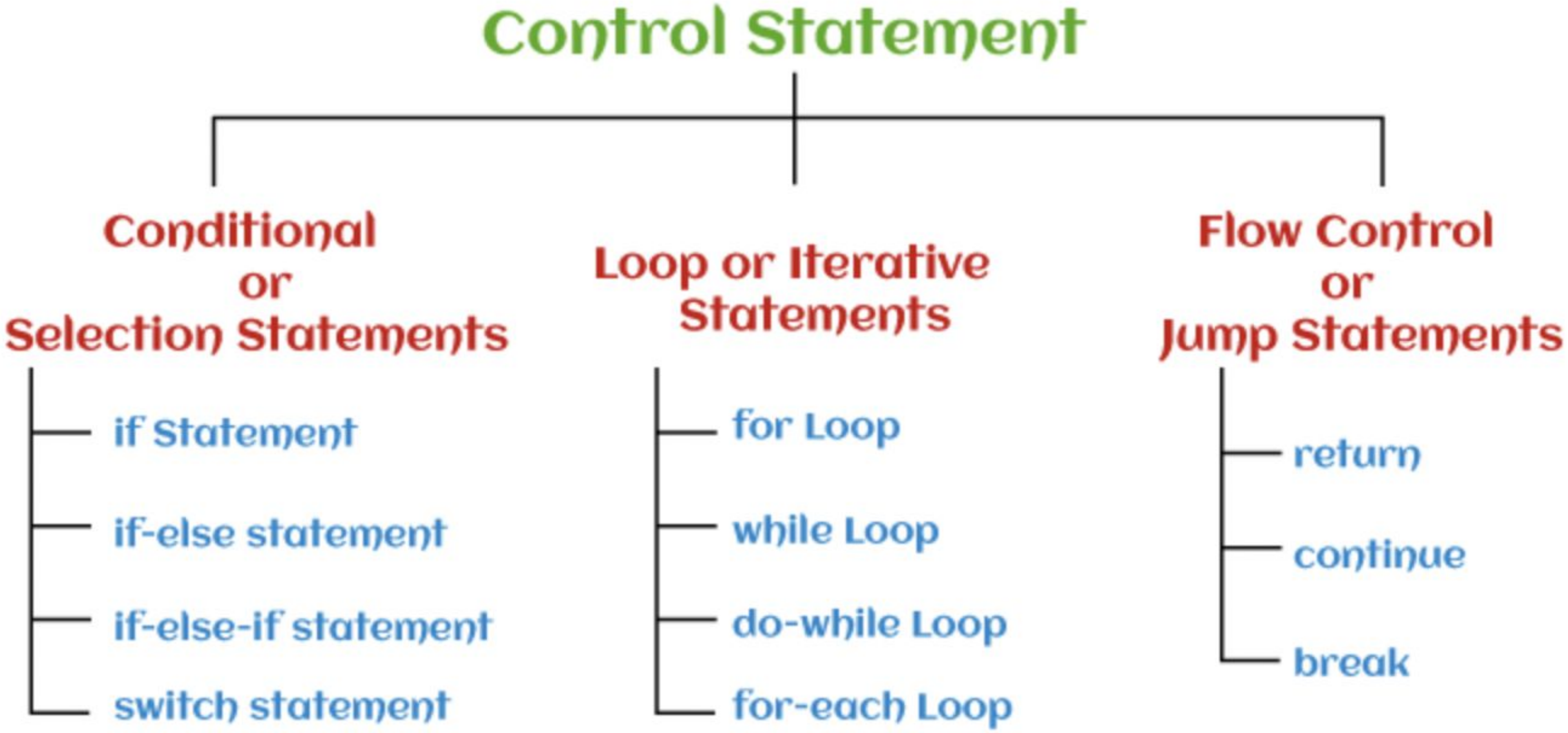
Contact us 966 5044 698
966 5044 598

Control Flow

Control flow in JavaScript is how our JS engine runs code from top to bottom.

It starts from the first line and ends at the last line, unless it hits any statement that changes the control flow of the program such as loops, conditionals, or functions

Control Flow statements



Control Statement

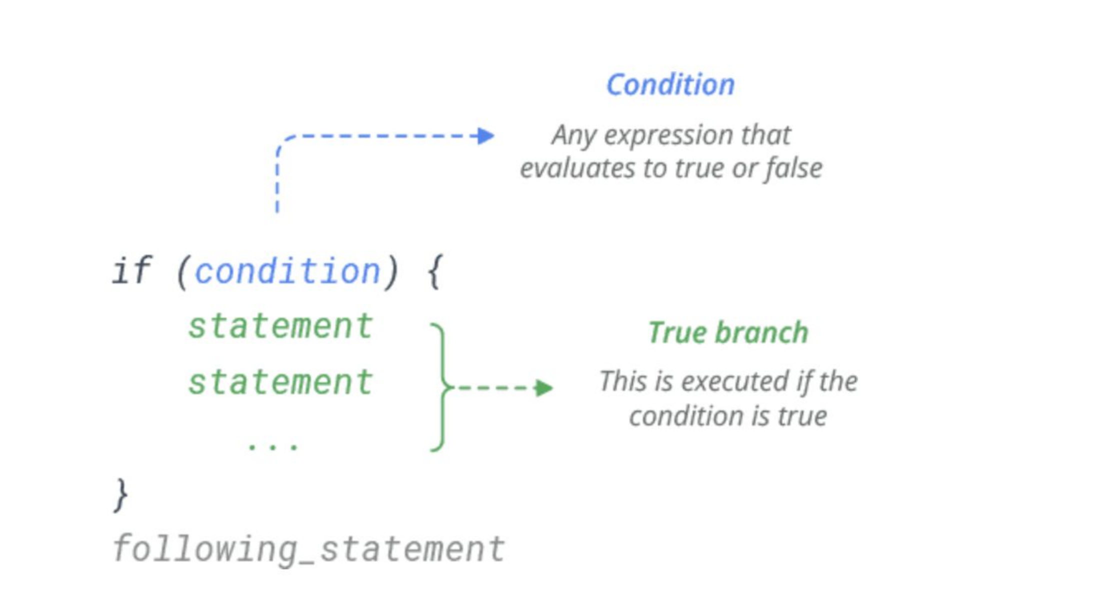
- Conditional or Selection Statements
  - if Statement
  - if-else statement
  - if-else-if statement
  - switch statement
- Loop or Iterative Statements
  - for Loop
  - while Loop
  - do-while Loop
  - for-each Loop
- Flow Control or Jump Statements
  - return
  - continue
  - break

if statement: Syntax and Flow chart

Condition can be checked using operators like

==, ===, !=, <, <=, >, &&, || and many more

Syntax:



```
if (condition) {
    statement
    statement
      ...
}
following_statement
```

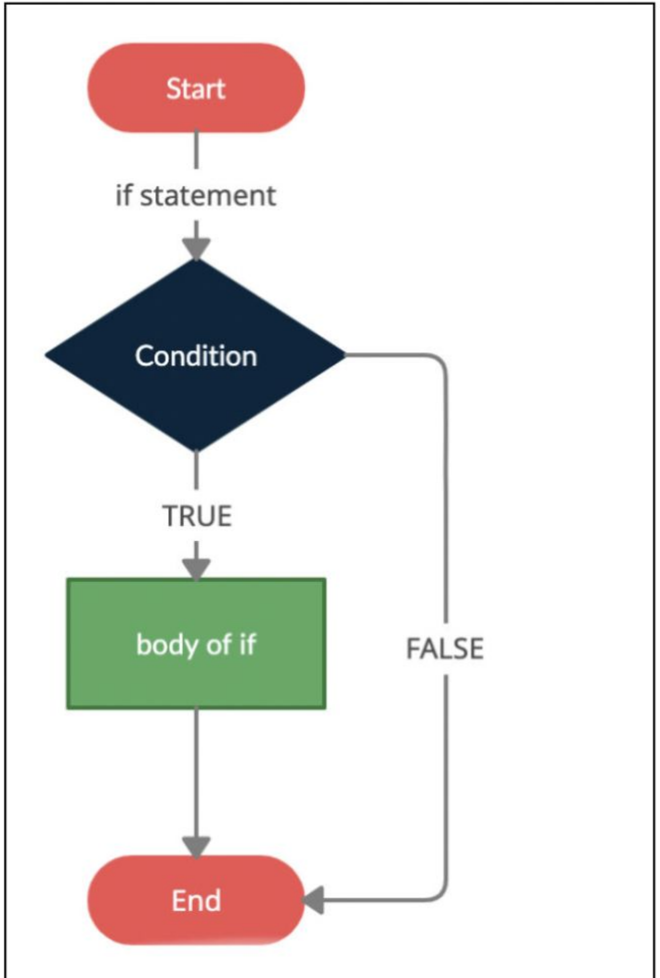
Condition

Any expression that evaluates to true or false

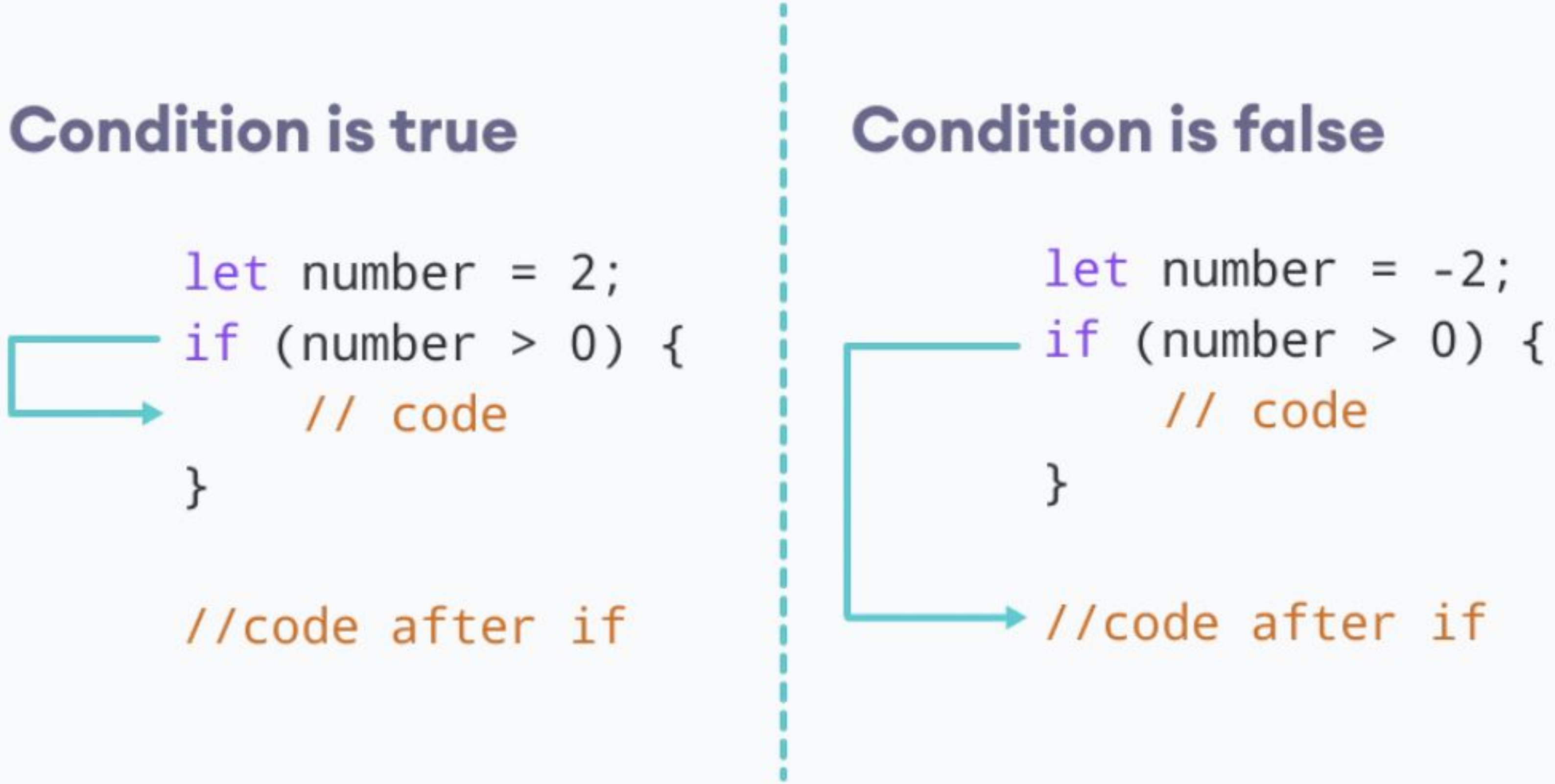
True branch

This is executed if the condition is true

Flow Chart



Conditional execution with true and false



Condition is true

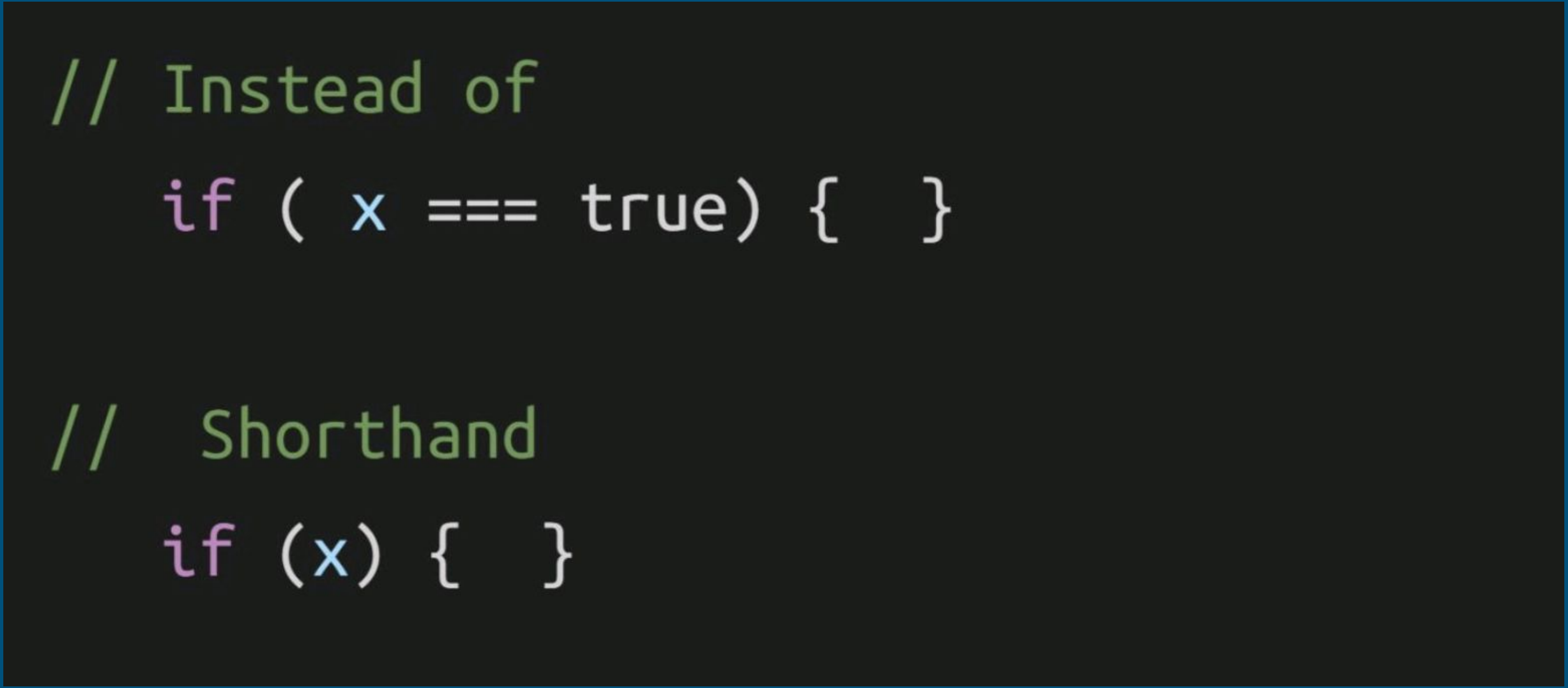
```
let number = 2;
if (number > 0) {
    // code
}

//code after if
```

Condition is false

```
let number = -2;
if (number > 0) {
    // code
}

//code after if
```

if statement

- If stmt without { } braces
  - In this case just one line is allowed inside if stmt body
  - Ex →
- Nested if stmt: We can write if statement inside if statement



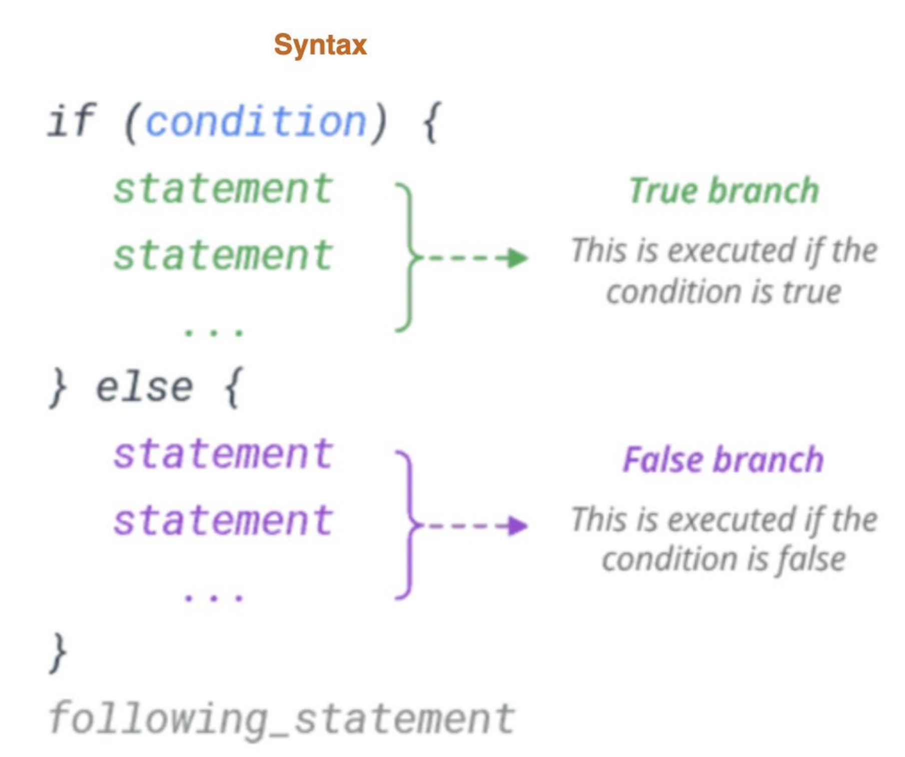
```
// Instead of
   if ( x === true) {  }


//  Shorthand
   if (x) {  }
```

Assignment 01: Please make sure to write function for each step and use the if block Of if else block

- File Name → 06_ifElseHandsOn.js

1. Create a function to check passed argument number value is even or odd and return the result as EVEN or ODD. Ex. 45, 70, 67, 98
2. Check if person is eligible for vote or not Ex→ age: 18, 20, 17, 40 → Ignore please
3. Check if string contains more than 10 character or not Ex → “JavaScript - ES6”
4. Check if string starts with “Java” Ex→ “JavaScript Language”

If else statement



Syntax

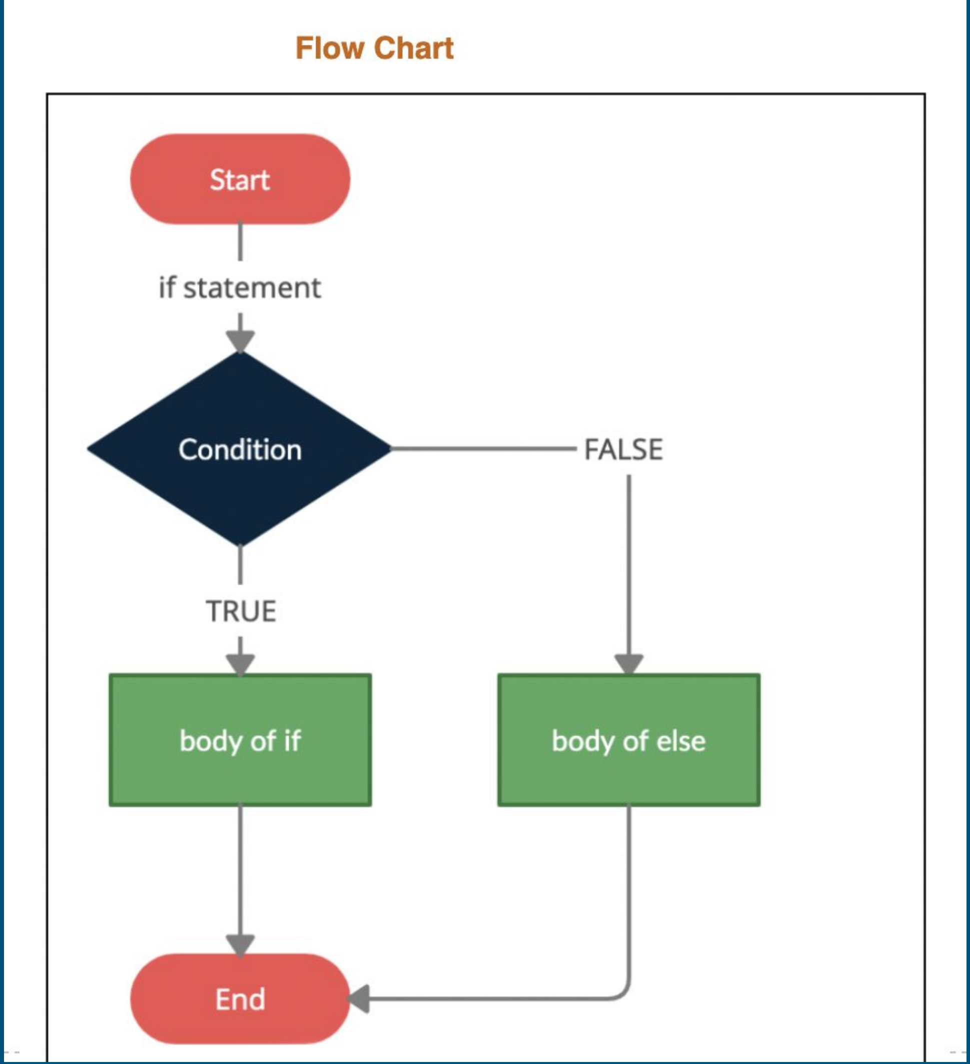
```
if (condition) {
    statement
    statement
      ...
} else {
    statement
    statement
      ...
}
following_statement
```

True branch

This is executed if the condition is true

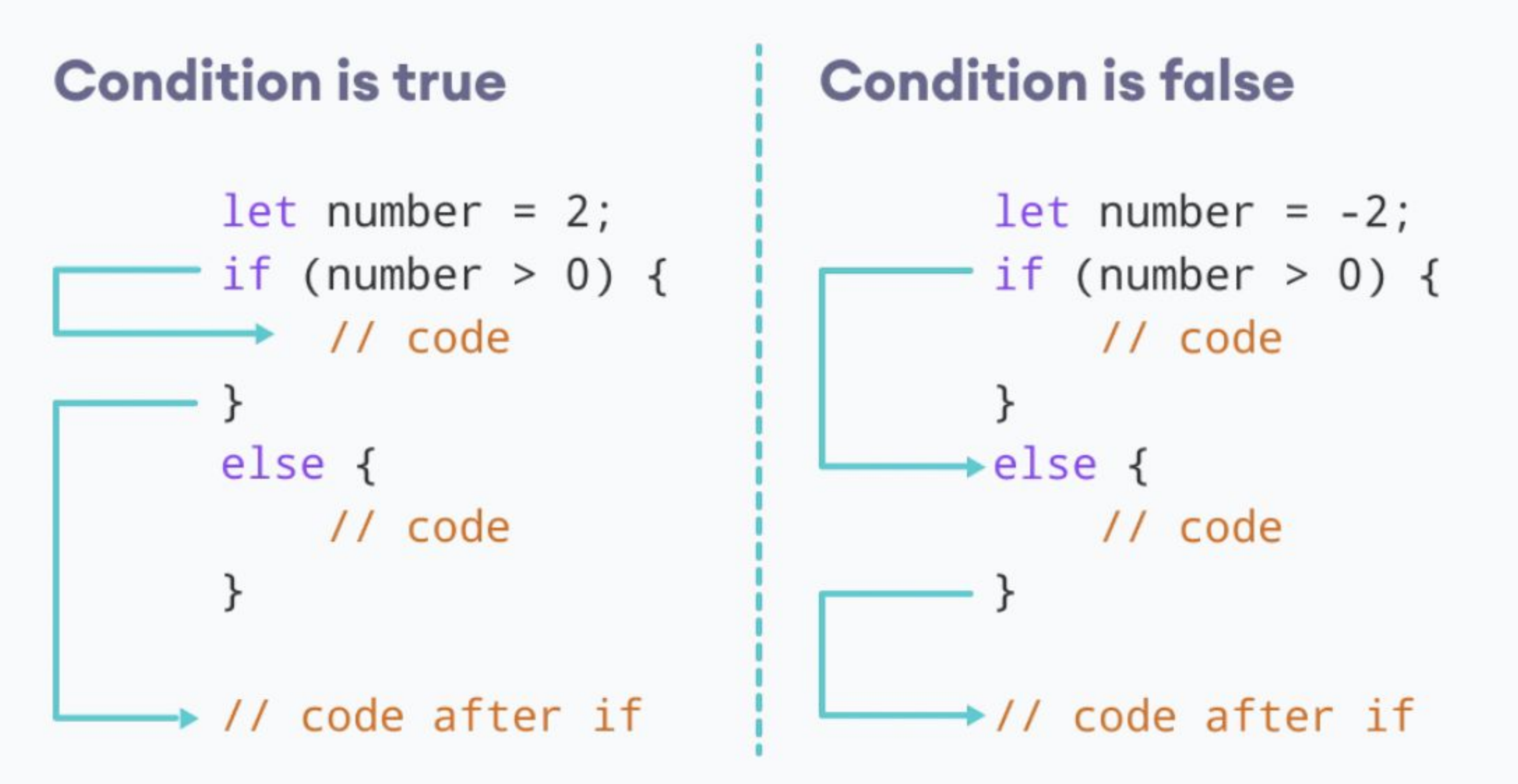
False branch

This is executed if the condition is false



Flow Chart

Example with both the condition flow



Condition is true

```
let number = 2;
if (number > 0) {
    // code
}
else {
    // code
}

// code after if
```

Condition is false

```
let number = -2;
if (number > 0) {
    // code
}
else {
    // code
}

// code after if
```

Assignment 02: Only use if else statement and commit the code, File→06_ifelse_grade.js

1. Write a FE and store into variable voteEligibe with one arg—> age to check whether he or she is eligible for voting or not, Then accordingly display message on console, don’t return the value please.

   Ex. 45, 17, 8, 20, -10, 200, 0, undefined, null

   Note:

   - If age value is 0 or negative or greater than 130 then display message → ‘In valid data’
   - If value is less than 18 then it should be → not eligible for vote
   - If value is greater than equal 18 then it should be → eligible for vote

Assignment 03: Only use if else statement and commit the code, File→06_grade.js

Design a grade system with function name as → gradeCalculation with argument marks and no return value

a. If marks is greater than equal to 90 then log the message→ Funtastic marks : ${marks}, Your grade is A+

b. If marks is greater than equal to 75 and less than 90 then log the msg → Excellent marks ${marks}, your grade is A

c. If marks is greater than equal to 50 and less than 75 then log the msg → Good marks ${marks}, your grade is B

d. If marks is greater than equal to 35 and less than 50 then log the msg → Marks is ${marks}, your grade is C, Need improvement

e. If marks is 0 or less than 0 or greater than 100 or not in valid number format ex. “Seventy” then log the msg→ Please provide the valid marks

f. Invoke function for values as shown → gradeCalculation(98), gradeCalculation(80), gradeCalculation(90), gradeCalculation(0), gradeCalculation(150), gradeCalculation(-7) , gradeCalculation(35), gradeCalculation(29), gradeCalculation(64), gradeCalculation(49), gradeCalculation(“91”), gradeCalculation(“Eighty”), gradeCalculation(undefined), gradeCalculation(null)

Assignment 03: if else statement File-06_interview.js

Fun expression with no return value to check TCS interview eligibility such as, If Graduation score is greater than equal to percentage 70 OR HSC score is greater than equal percentage 80 OR SSC score is greater than percentage 90 then only

Note: Not required to handle the invalid data

1.1. Function args → gradScore, hscScore, sscScore, candidateName

1.2. Congrates {candidate_name} you are eligible for TCS interview. Else Unfortunately you are not eligible for interview

1.3. Invoke Fun Expr with values as

  1.3.1. 80, 86, 90, "your name"

  1.3.2. 70, 65, 55, "your frd name"

  1.3.3. 60, 79, 88, " your other frd name "

Assignment : Design a grade system with gradeSystem(), Negative marks are not allowed

1. Function gradeSystem( ) with one argument ‘score’
2. Score less than 35 then log msg ‘You are Failed’
3. Score more than equal to 35 then log msg ‘You are Passed’
4. Score is greater than equal 35 and less than 60 then log msg Passed and GRADE is ‘C’
5. Score is greater than equal to 60 and less than 75 then log msg Passed and GRADE is ‘B’
6. score is greater than equal to 75 and less than equal to 90 then log msg Passed and GRADE is ‘A”
7. Score is greater than equal to 90 and less than equal to 100 then log msg Passed and GRADE is ‘A+’
8. Log message ‘Invalid Input’ on console for unhappy path scenarios like
   - 8.1. Invalid score like -ve score or more than 100
   - 8.2. Invalid input like in string format “Fourty Five Score”
   - 8.3. Invalid input null or undefined

Invoke the gradeSystem() function for below values as shown

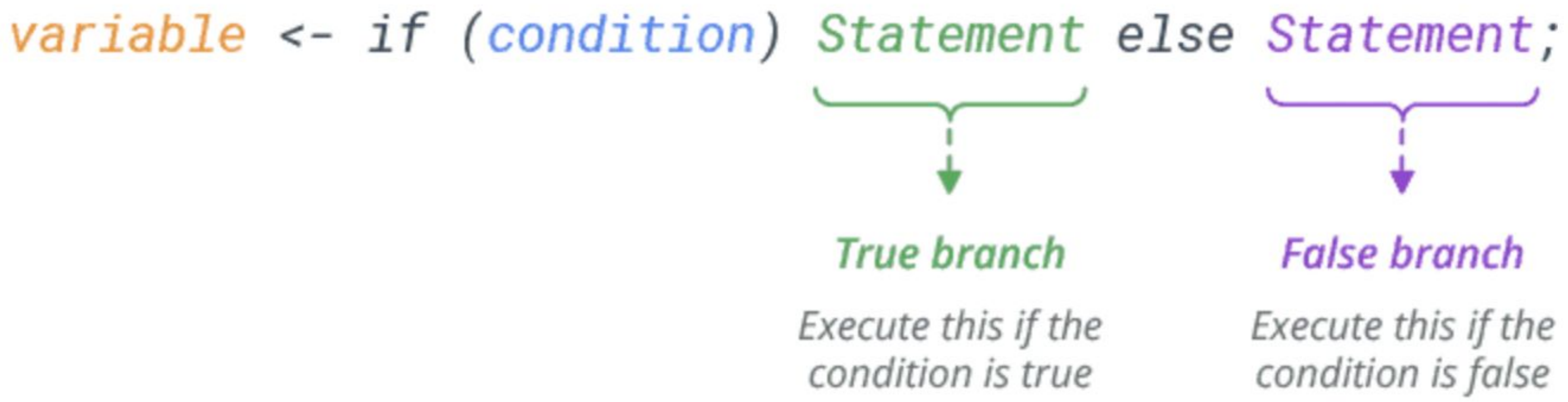
```
gradeSystem(66);          gradesystem(“fifty Five”);     gradeSystem(undefined);
gradeSystem(13);          gradeSystem(35);               gradeSystem(-20);
gradesystem(“ ”);         gradeSystem(75);               gradeSystem(55);
gradesystem(98);          gradeSystem(null);             gradeSystem(82);
```

Assignment 01: if else stmt, Pls have the separate function for each main step

1. A Function to check the number is even or odd using if else statement and log the result as it is on console [any value with number data type is valid]
   1.1. Ex → 2, 45, 13, 0, “70”
2. Implement a function with three args to check the greatest number amongst three numbers using if else if statements and call the same function for following inputs. Note: Don’t forget to check the valid input number then only do the comparison. In case invalid inputs, please ignore it and consider the valid numbers that +ve or -ve
   2.1. Ex → 56, 70, 80
   2.2. Ex. → -20, -90, 0

Ternary Operator or Conditional Operator

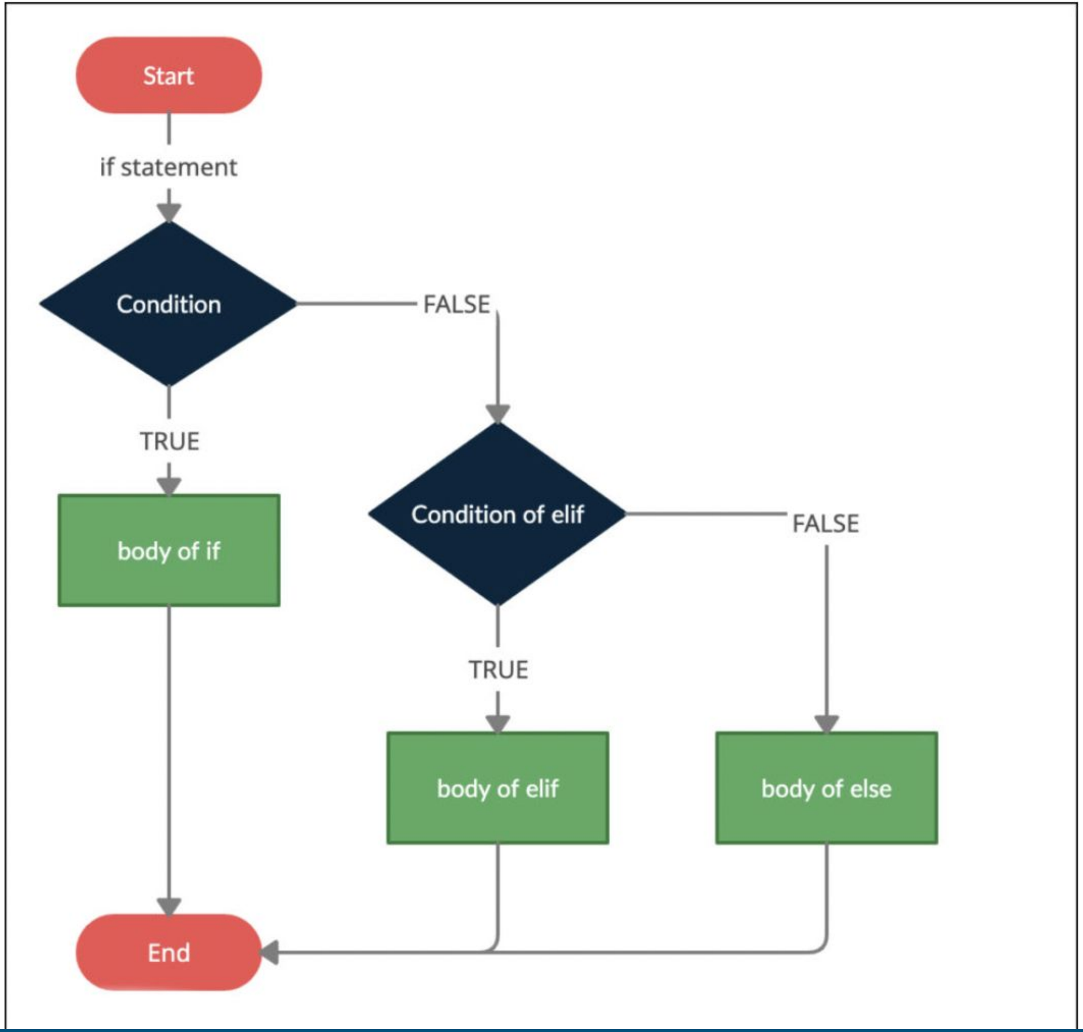
Ternary operator is similar to if else statement



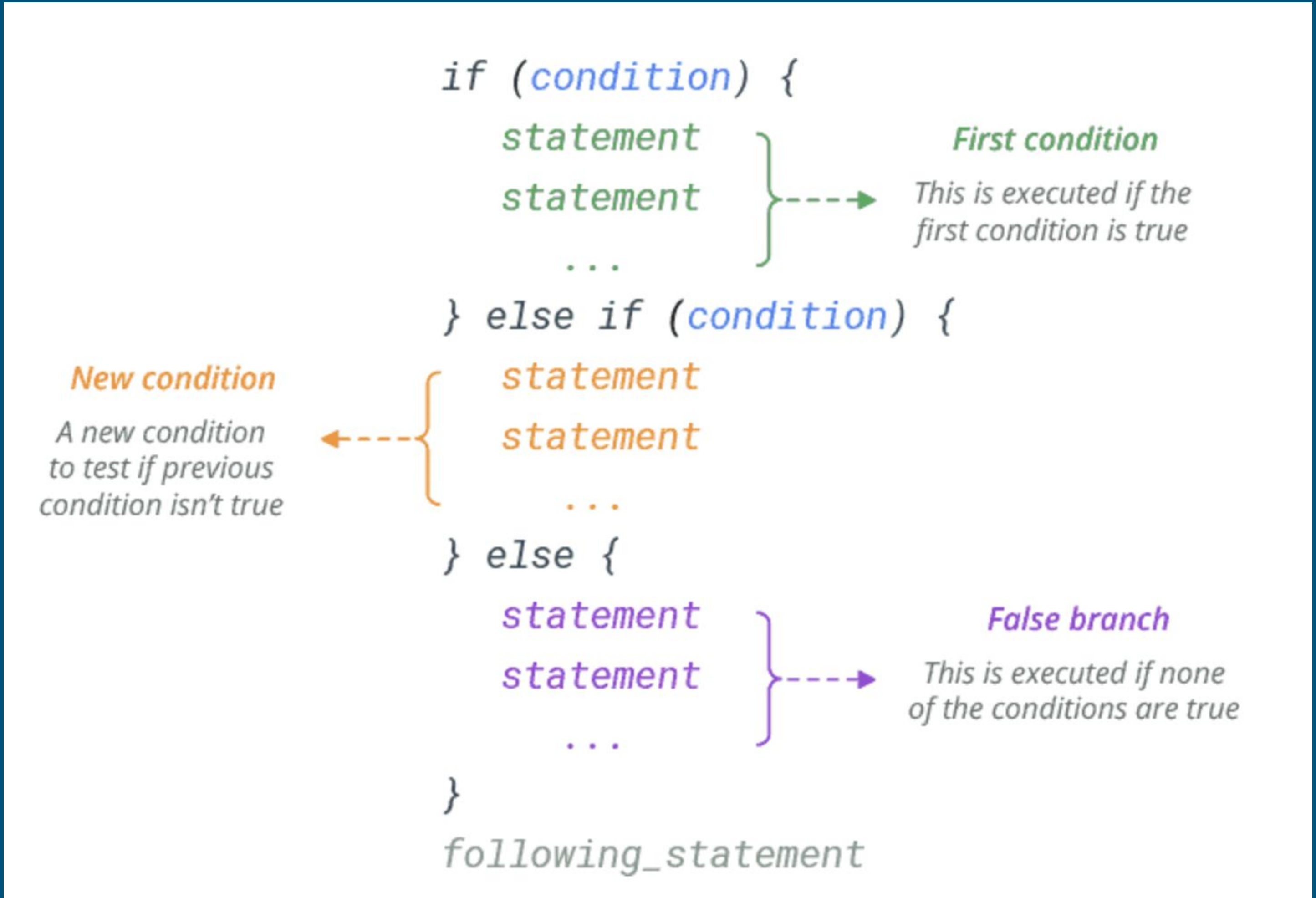
```
variable <- if (condition) Statement else Statement;
```

True branch
Execute this if the condition is true
False branch
Execute this if the condition is false

If else if else statement



Syntax



```
if (condition) {
    statement
    statement
      ...
} else if (condition) {
    statement
    statement
      ...
} else {
    statement
    statement
      ...
}
following_statement
```

First condition

This is executed if the first condition is true

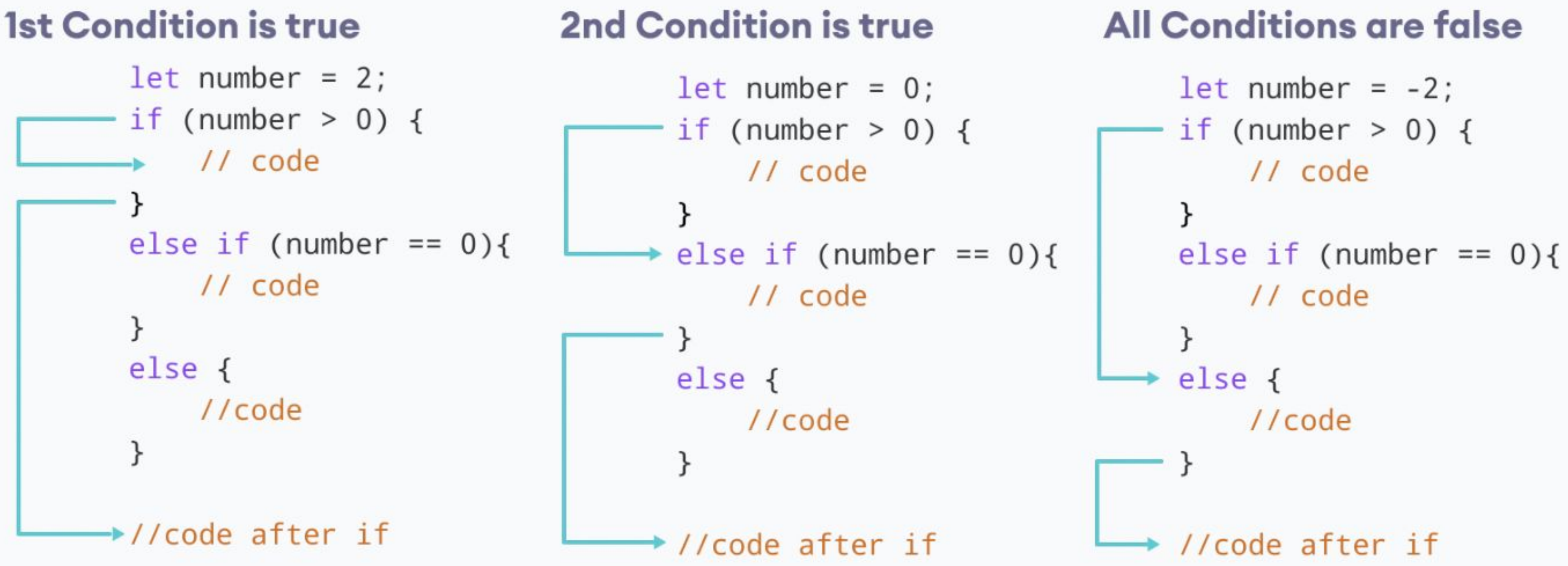
New condition

A new condition to test if previous condition isn't true

False branch

This is executed if none of the conditions are true

Example with all possible condition execution



1st Condition is true

```
let number = 2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

2nd Condition is true

```
let number = 0;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

All Conditions are false

```
let number = -2;
if (number > 0) {
    // code
}
else if (number == 0){
    // code
}
else {
    //code
}

//code after if
```

Assignment 02: File→ 06_marriageEligibility.js

1. Please design a marriage eligibility checker system to decide whether he or she is eligible for marriage or not based on two input values gender and age

   1.1 “Male”, 17

   1.2 “Male”, 25

   1.3 “Female”, 28

   1.4 “Female”, 16

   1.5 “Other”, 35

   1.6 “Other”, 41

   Note: Consider the age criteria defined by our Indian constitution as per the old act. Male >=21, Female >=18

Switch case statement

-A switch stmt can replace multiple if checks

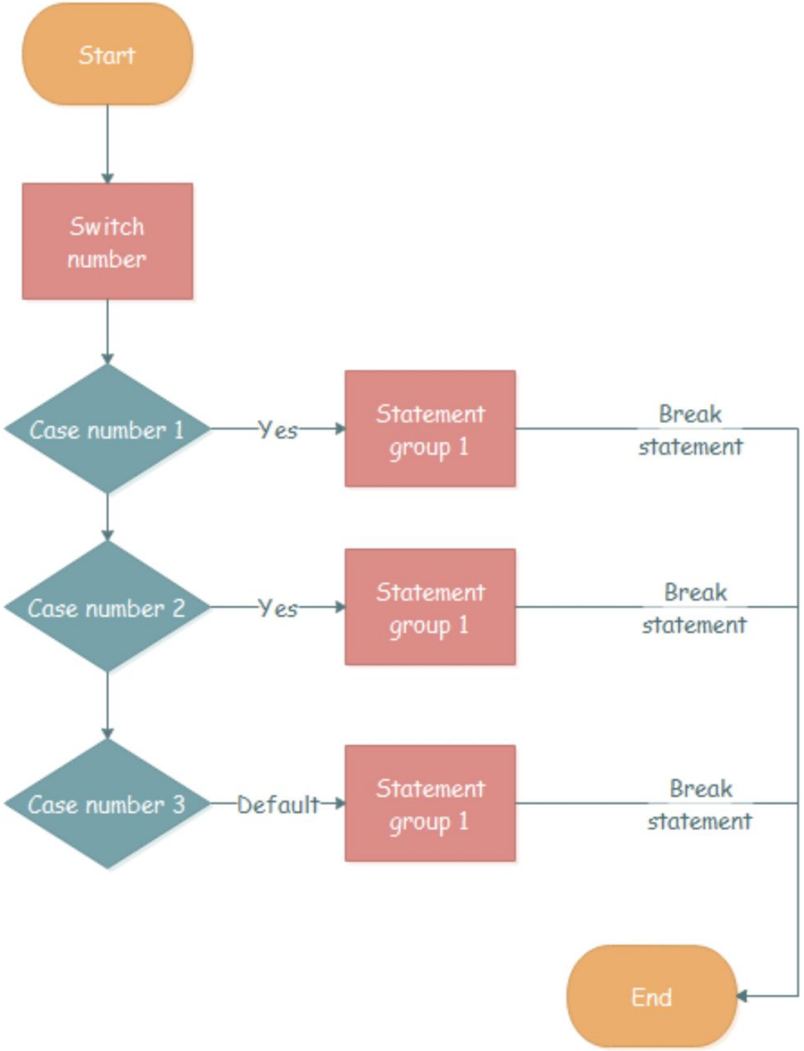
-It gives a more descriptive way to compare a value

With multiple variants

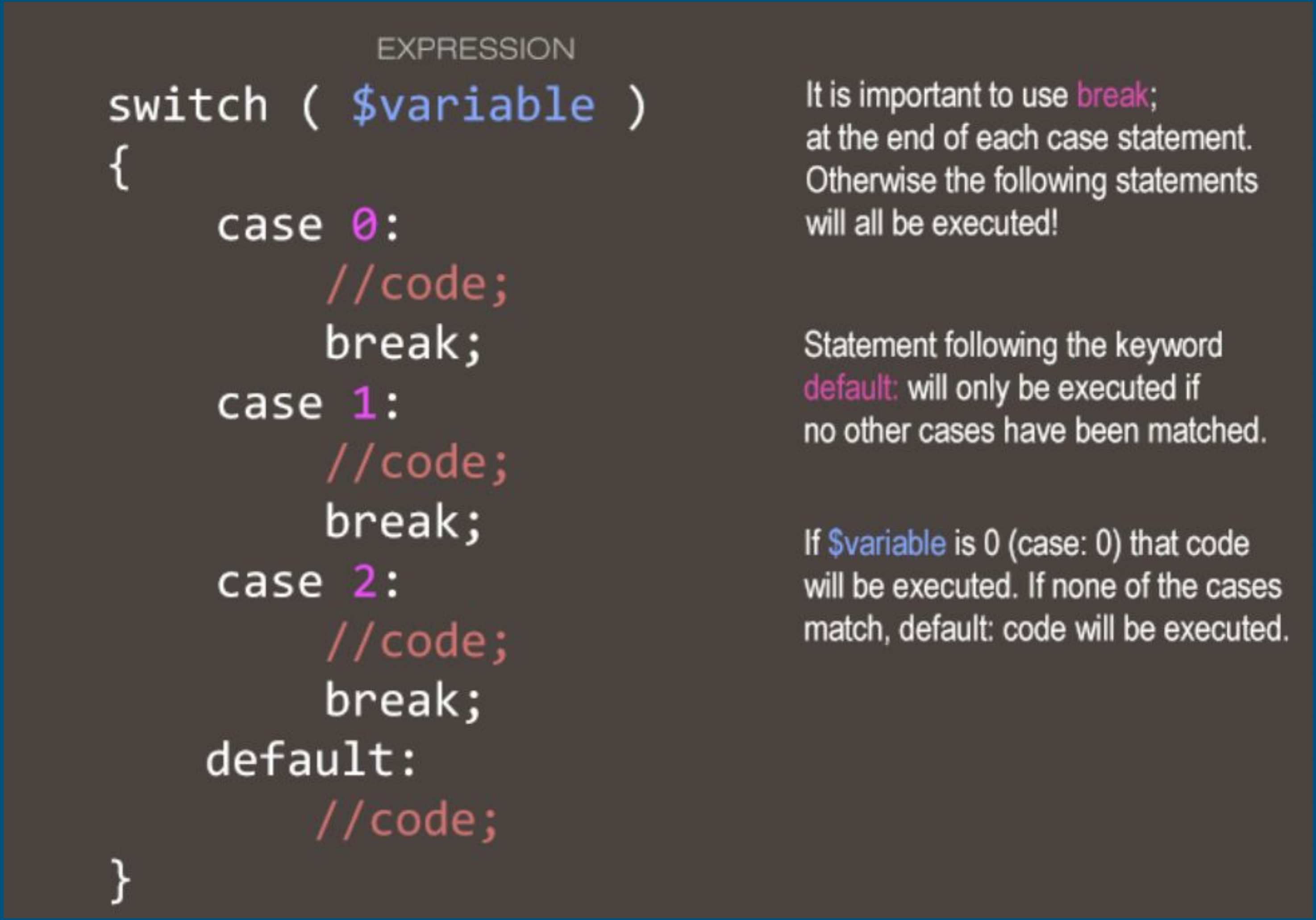
-The switch has one or more case blocks and

an optional default

-Break is imp, otherwise will check all conditions



Syntax



EXPRESSION

```
switch ( $variable )
{
    case 0:
        //code;
        break;
    case 1:
        //code;
        break;
    case 2:
        //code;
        break;
   default:
       //code;
}
```

It is important to use break; at the end of each case statement. Otherwise the following statements will all be executed!

Statement following the keyword default: will only be executed if no other cases have been matched.

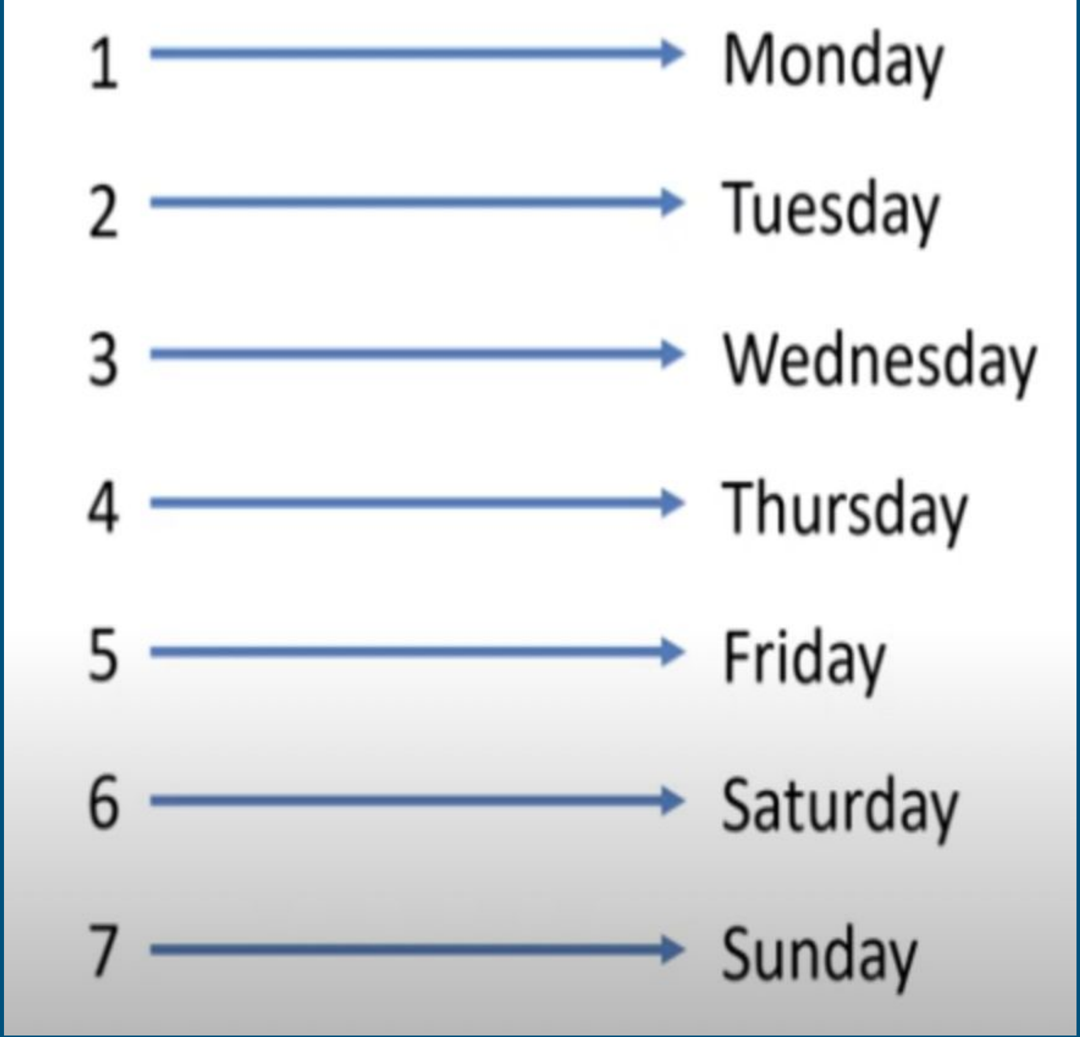
If $variable is 0 (case: 0) that code will be executed. If none of the cases match, default: code will be executed.

Assignment

Given a day in number and then make

Make a decision and log on console

- Week day
- Working or non working day



1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Sunday

Tricky Points

Case statements without break statement

Different values (types) that we can pass in switch( )

Switch case without default

Assignment 01: File→ 06_switchCaseMonth.js

Create Normal Function with name →monthOfYear( ) with one arg → month. Inside this write a switch case and pass month in switch

1. Given a month number then log the name of the month

   1 —- January

   2 —-- February

   ……………………..

   12 —-- December

Don;t forget to add the default case.

Invoke function monthOfYear(0) for monthOfYear(2), monthOfYear(5), monthOfYear(12), monthOfYear(15), monthOfYear(100) , monthOfYear(null) , monthOfYear(undefined)

Assignment 03: Design a **function expression** using if else to check given year is leap year or not

Please understand the leap year what it is first before function implementation.

Function expression with name leapYear and one argument that is year

Don’t forget to check valid and invalid input based on that log msg on console [Any value with number data type is valid]

Input values: Call the same function for all these below values

→ 2020, 1999, 1600, 2022, 1945, null, “Twenty Twenty”, undefined, NaN, 1750

Note: I may ask any one to explain the code and leap year concept as well in session

Real time Example

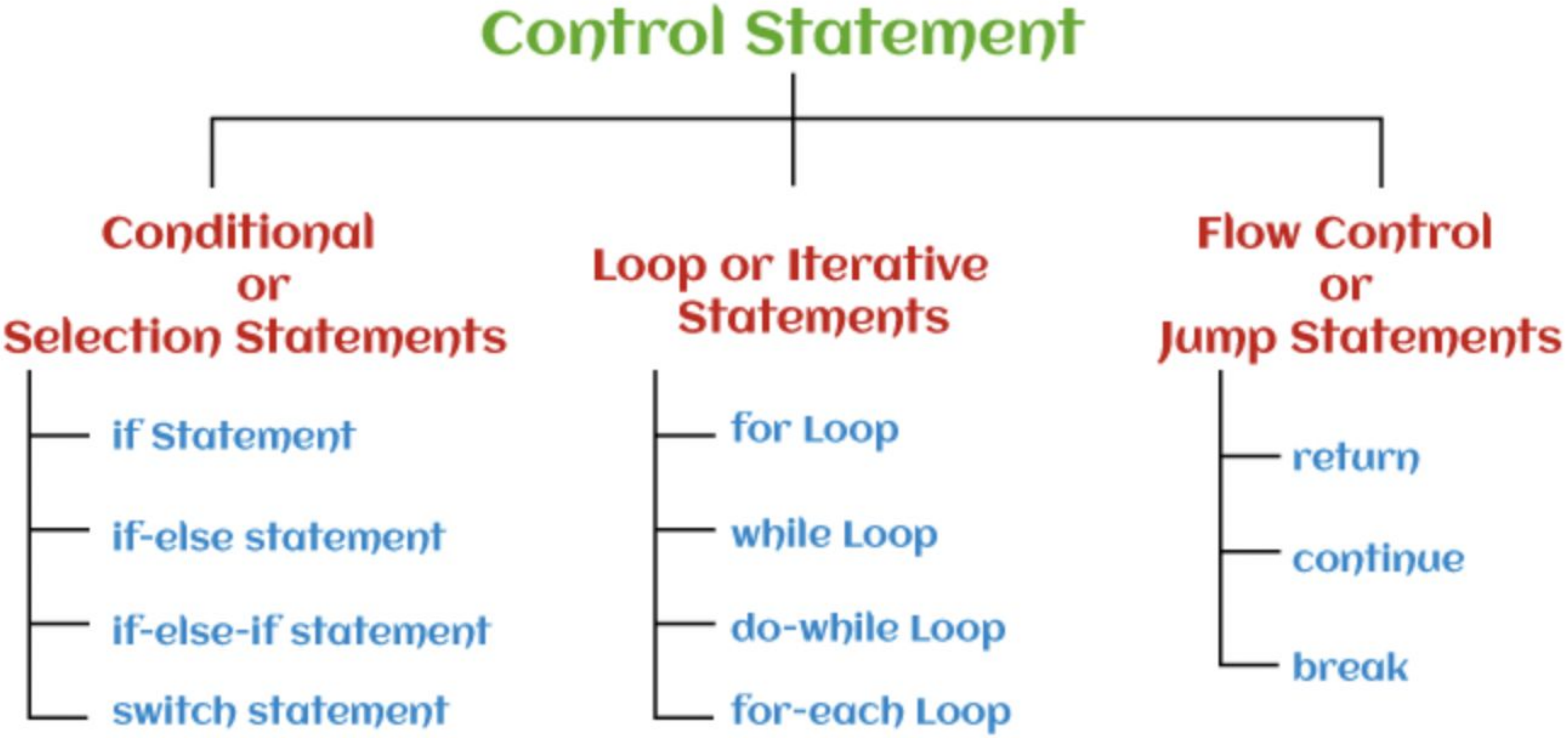
Journey Mumbai to Pune expressway

Example:

Print 1 to 100 on console

Print table of 5

Control Flow statements



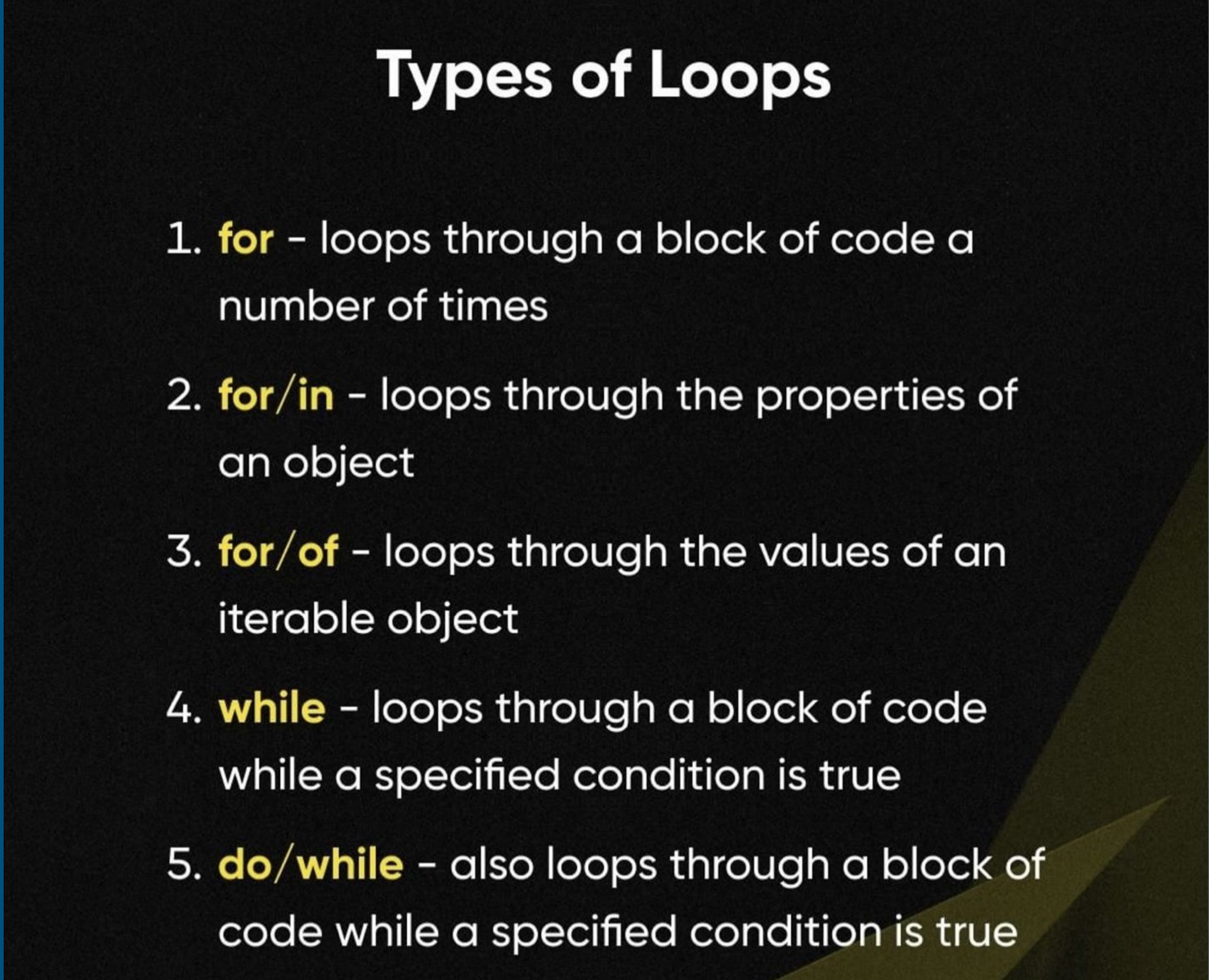
Control Statement

Conditional or Selection Statements
- if Statement
- if-else statement
- if-else-if statement
- switch statement

Loop or Iterative Statements
- for Loop
- while Loop
- do-while Loop
- for-each Loop

Flow Control or Jump Statements
- return
- continue
- break

Loop statements



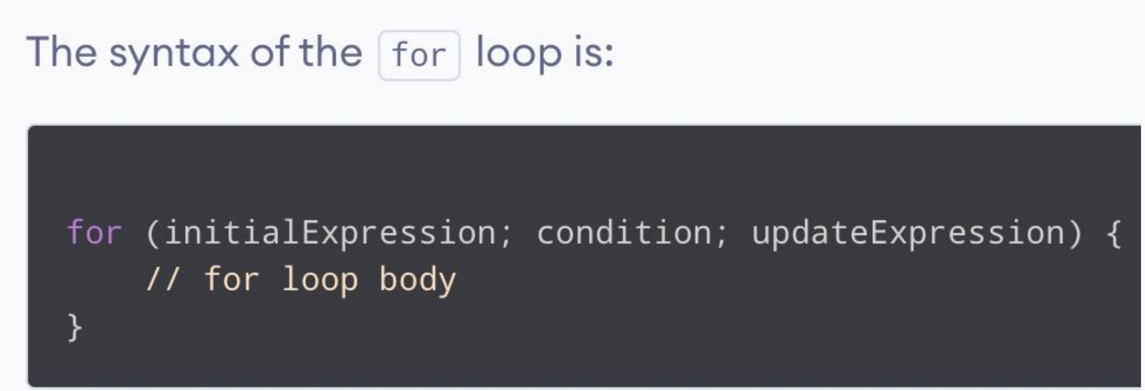
Types of Loops

1. **for** - loops through a block of code a number of times
2. **for/in** - loops through the properties of an object
3. **for/of** - loops through the values of an iterable object
4. **while** - loops through a block of code while a specified condition is true
5. **do/while** - also loops through a block of code while a specified condition is true

For loop flow chart

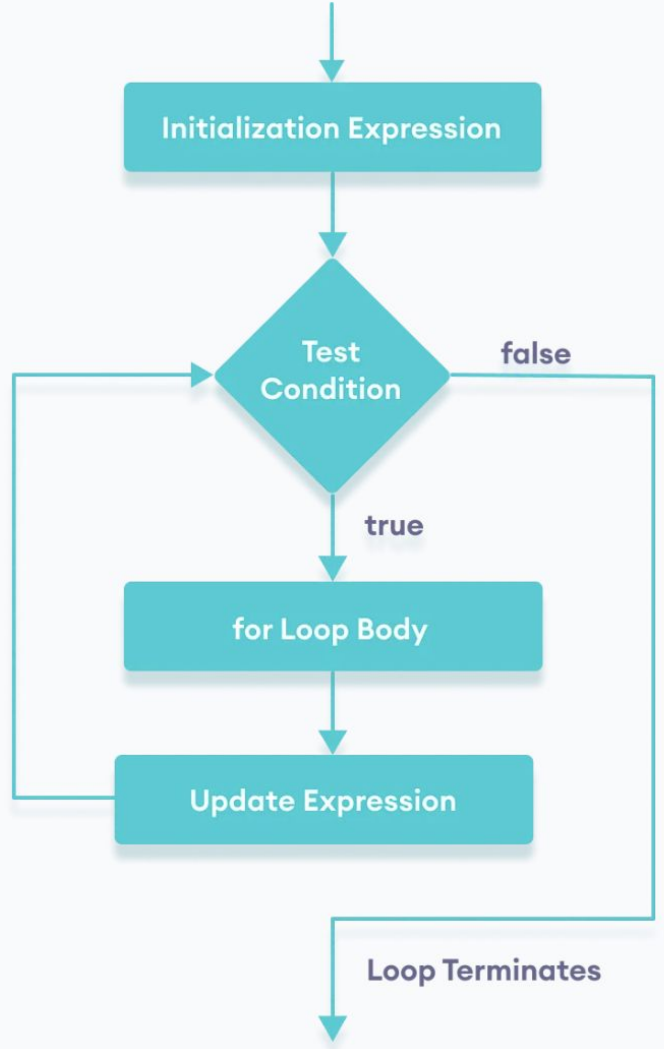
Loops are used to repeat a block of code

Ex → print numbers from 1 to 10



The syntax of the `for` loop is:

```
for (initialExpression; condition; updateExpression) {
    // for loop body
}
```



Assignments 01: 06_for_Assignment01.js

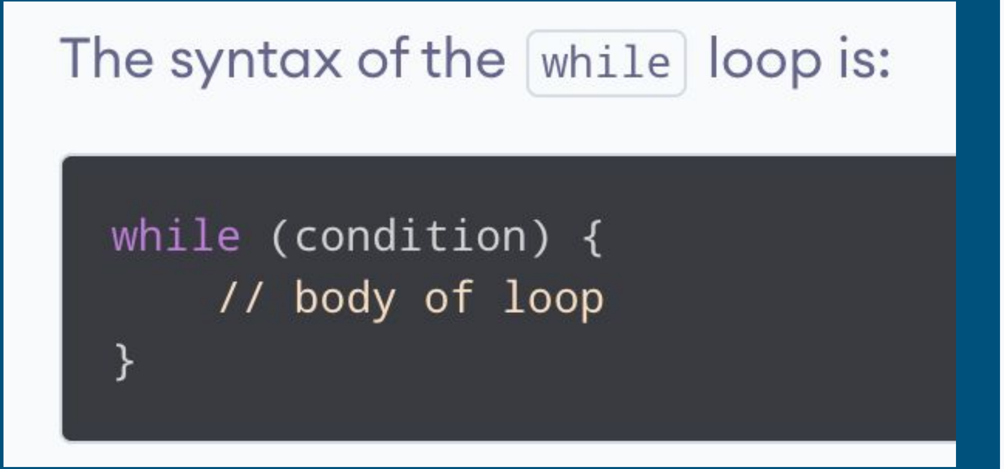
1. WAP to print numbers from 5 to 15 by incrementing 1
2. WAP to print numbers from 50 to 40 by decrementing by 1
3. WAP to find first 15 odd numbers
4. WAP to find first 10 even numbers
5. WAP to print table of 5 like → 5 10 15 20 25 …… 50
6. WAP to print table of 10 like 10 20 30 40 …… 100
7. WAP to print table of 10 in reverse order like → 100 90 80 70 …… 10

Assignments 01: 06_while_Assignment01.js, Using while loop only

1. WAP to print numbers from 5 to 15 by incrementing 1
2. WAP to print numbers from 50 to 40 by decrementing by 1
3. WAP to find first 15 odd numbers
4. WAP to find first 10 even numbers
5. WAP to print table of 5 like → 5 10 15 20 25 …… 50
6. WAP to print table of 10 like 10 20 30 40 …… 100
7. WAP to print table of 10 in reverse order like → 100 90 80 70 …… 10

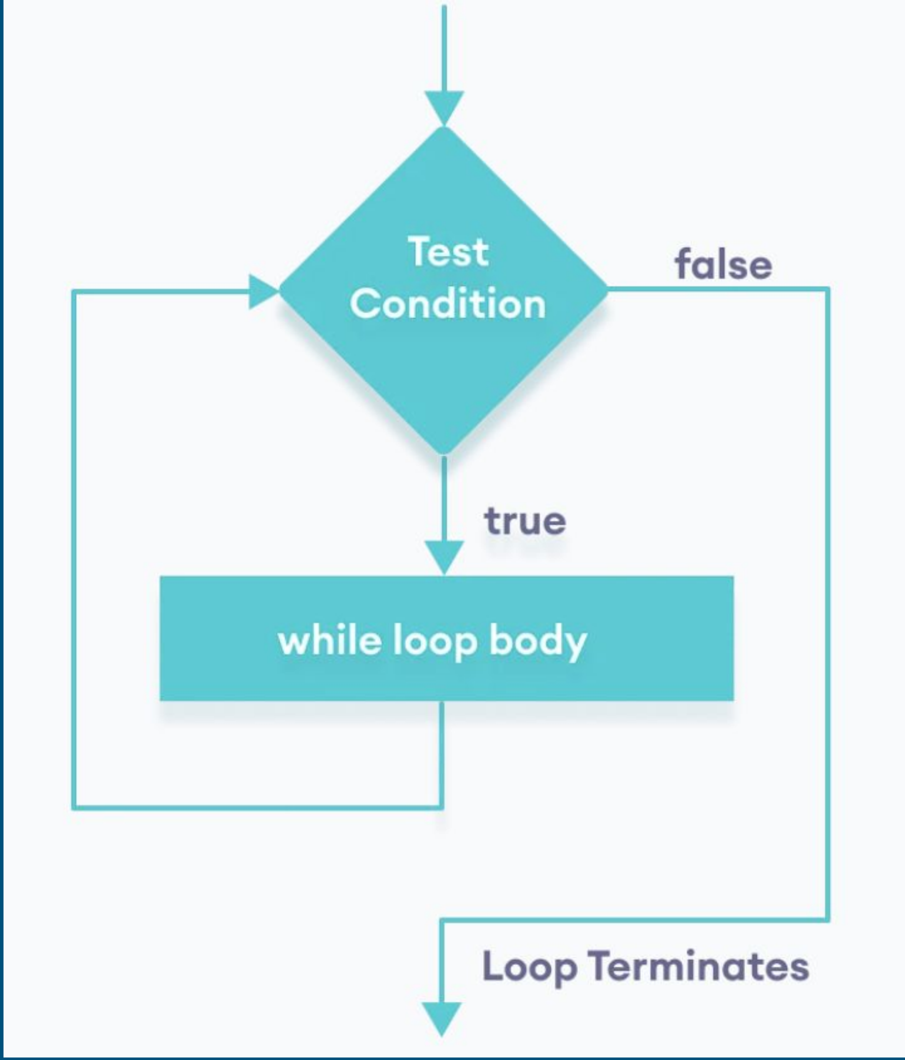
While loop

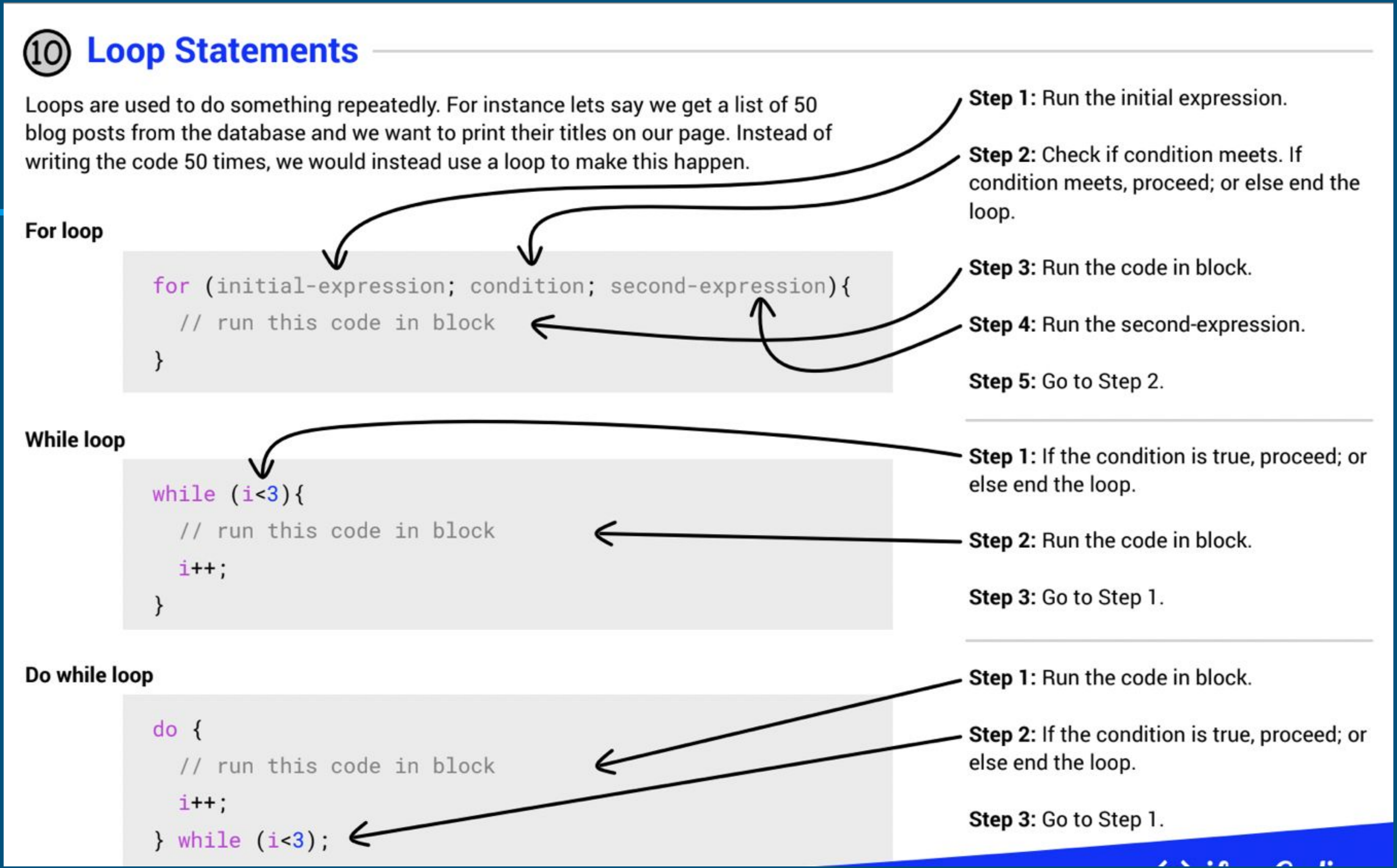
Ex → print numbers from 1 to 10



The syntax of the `while` loop is:

```
while (condition) {
    // body of loop
}
```





⑩ Loop Statements

Loops are used to do something repeatedly. For instance lets say we get a list of 50 blog posts from the database and we want to print their titles on our page. Instead of writing the code 50 times, we would instead use a loop to make this happen.

For loop

```
for (initial-expression; condition; second-expression){
  // run this code in block
}
```

Step 1: Run the initial expression.

Step 2: Check if condition meets. If condition meets, proceed; or else end the loop.

Step 3: Run the code in block.

Step 4: Run the second-expression.

Step 5: Go to Step 2.

While loop

```
while (i<3){
  // run this code in block
  i++;
}
```

Step 1: If the condition is true, proceed; or else end the loop.

Step 2: Run the code in block.

Step 3: Go to Step 1.

Do while loop

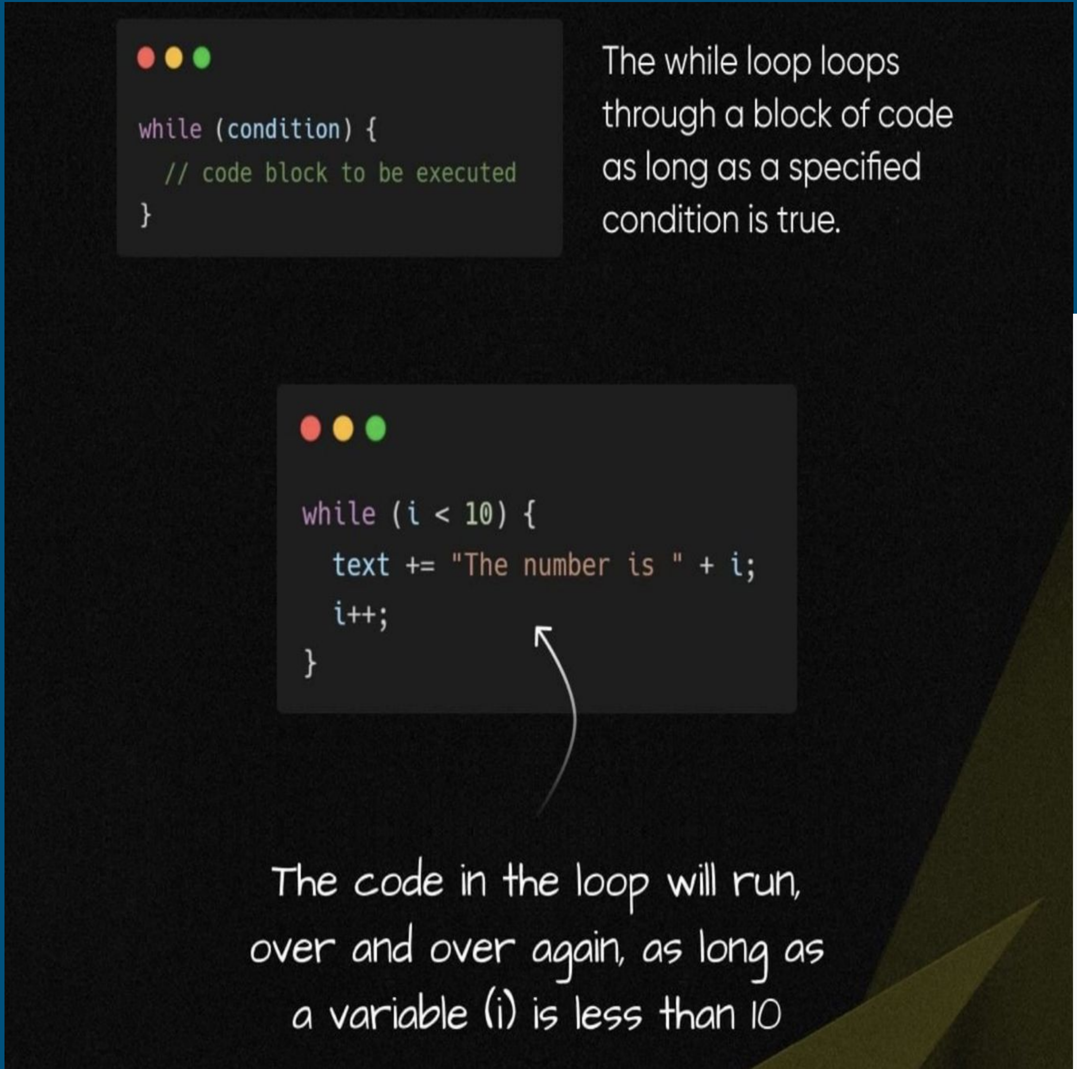
```
do {
  // run this code in block
  i++;
} while (i<3);
```

Step 1: Run the code in block.

Step 2: If the condition is true, proceed; or else end the loop.

Step 3: Go to Step 1.

While loop

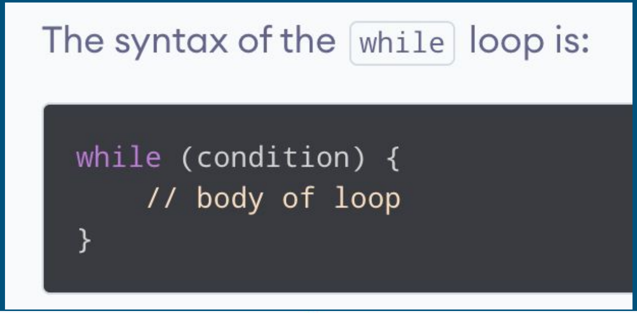


```
while (condition) {
  // code block to be executed
}
```

The while loop loops through a block of code as long as a specified condition is true.

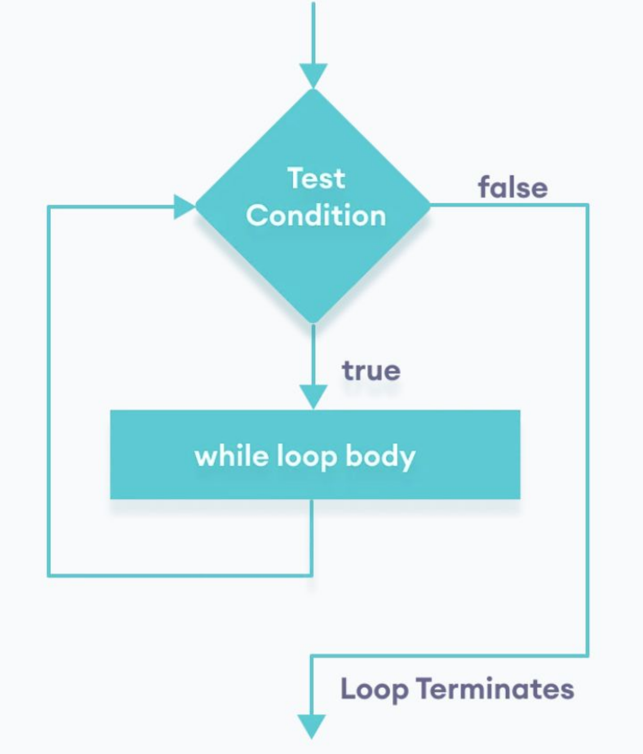
```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

The code in the loop will run, over and over again, as long as a variable (i) is less than 10

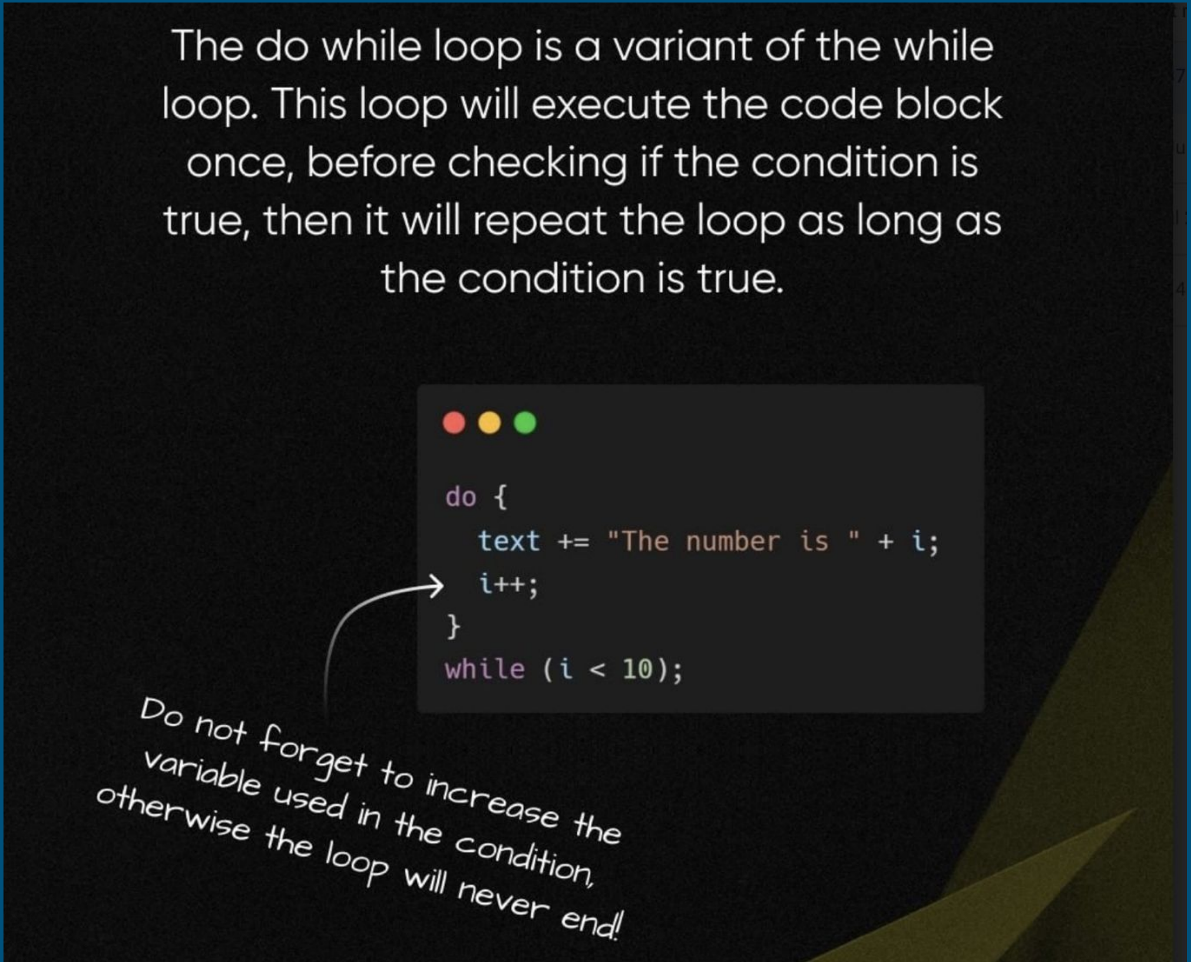


The syntax of the `while` loop is:

```
while (condition) {
    // body of loop
}
```



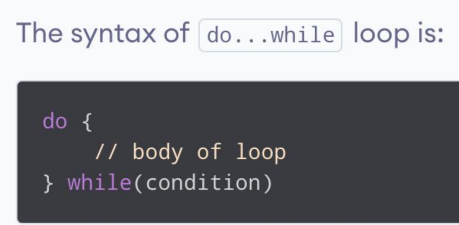
do while loop



The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

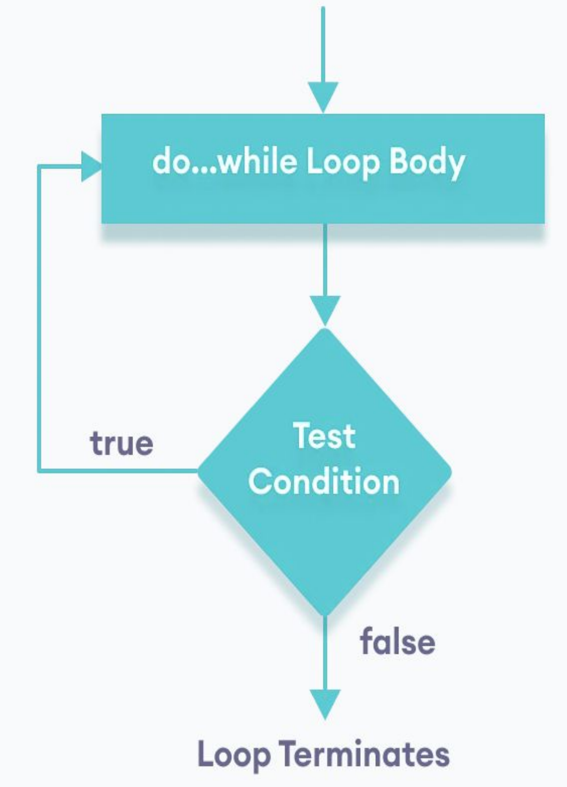
```
do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```

Do not forget to increase the variable used in the condition, otherwise the loop will never end!

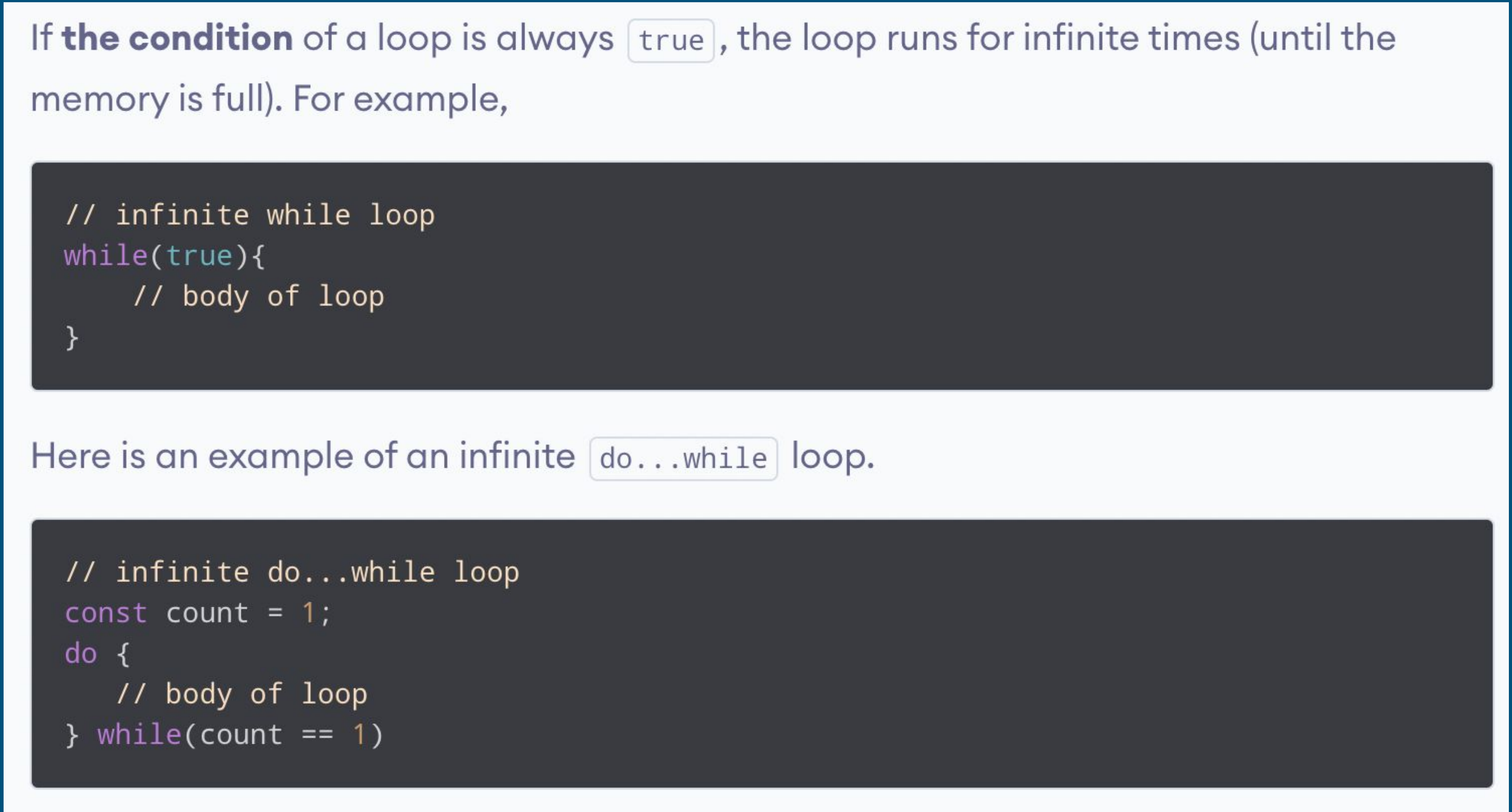


The syntax of `do...while` loop is:

```
do {
    // body of loop
} while(condition)
```



Infinite loop



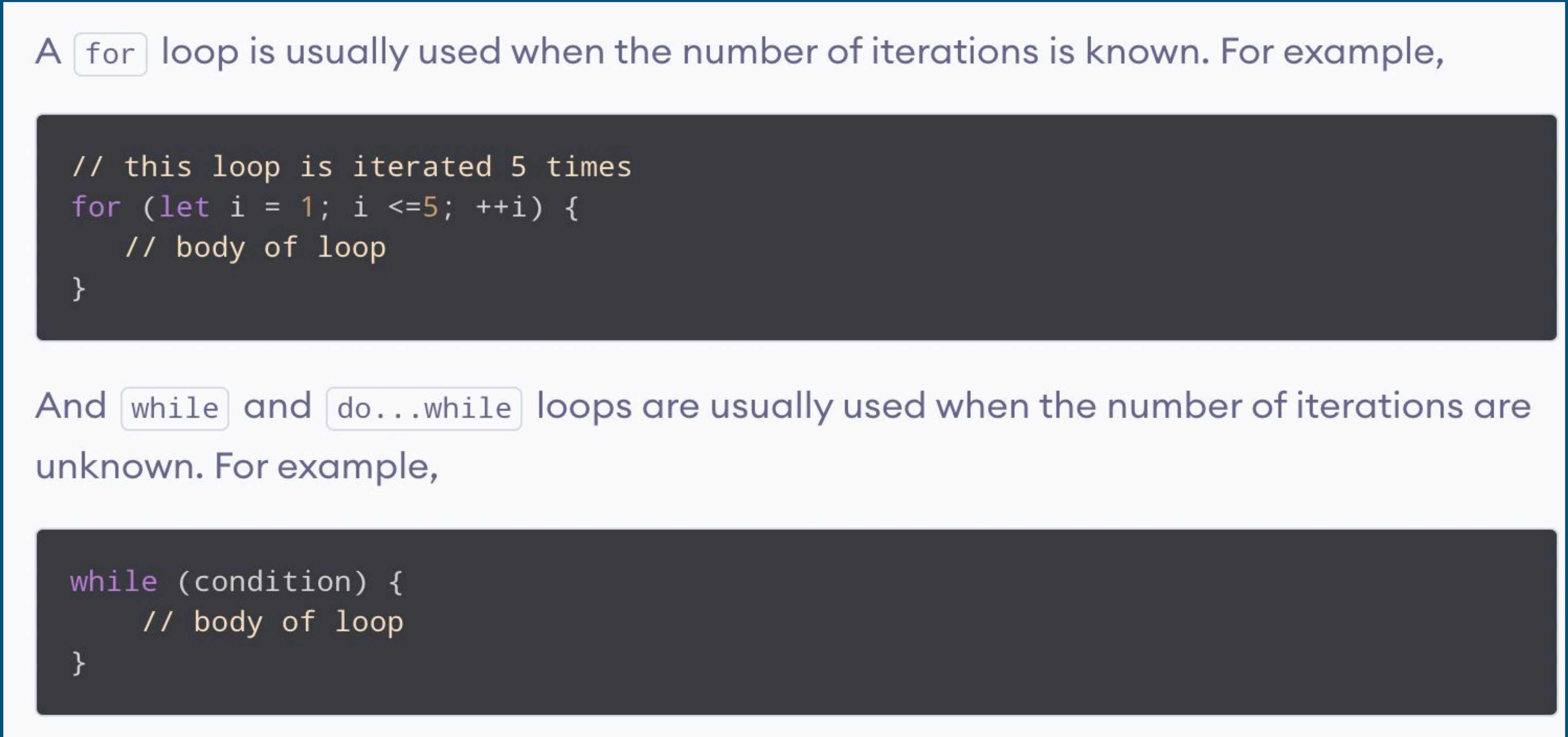
If **the condition** of a loop is always `true`, the loop runs for infinite times (until the memory is full). For example,

```
// infinite while loop
while(true){
    // body of loop
}
```

Here is an example of an infinite `do...while` loop.

```
// infinite do...while loop
const count = 1;
do {
   // body of loop
} while(count == 1)
```

Diff. between for loop and while loop



A `for` loop is usually used when the number of iterations is known. For example,

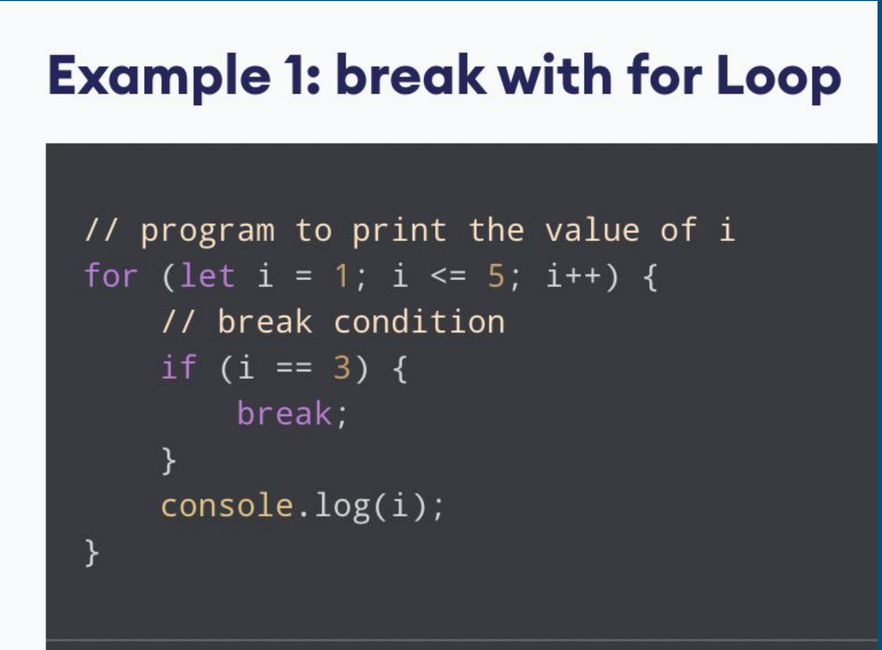
```
// this loop is iterated 5 times
for (let i = 1; i <=5; ++i) {
   // body of loop
}
```

And `while` and `do...while` loops are usually used when the number of iterations are unknown. For example,

```
while (condition) {
    // body of loop
}
```

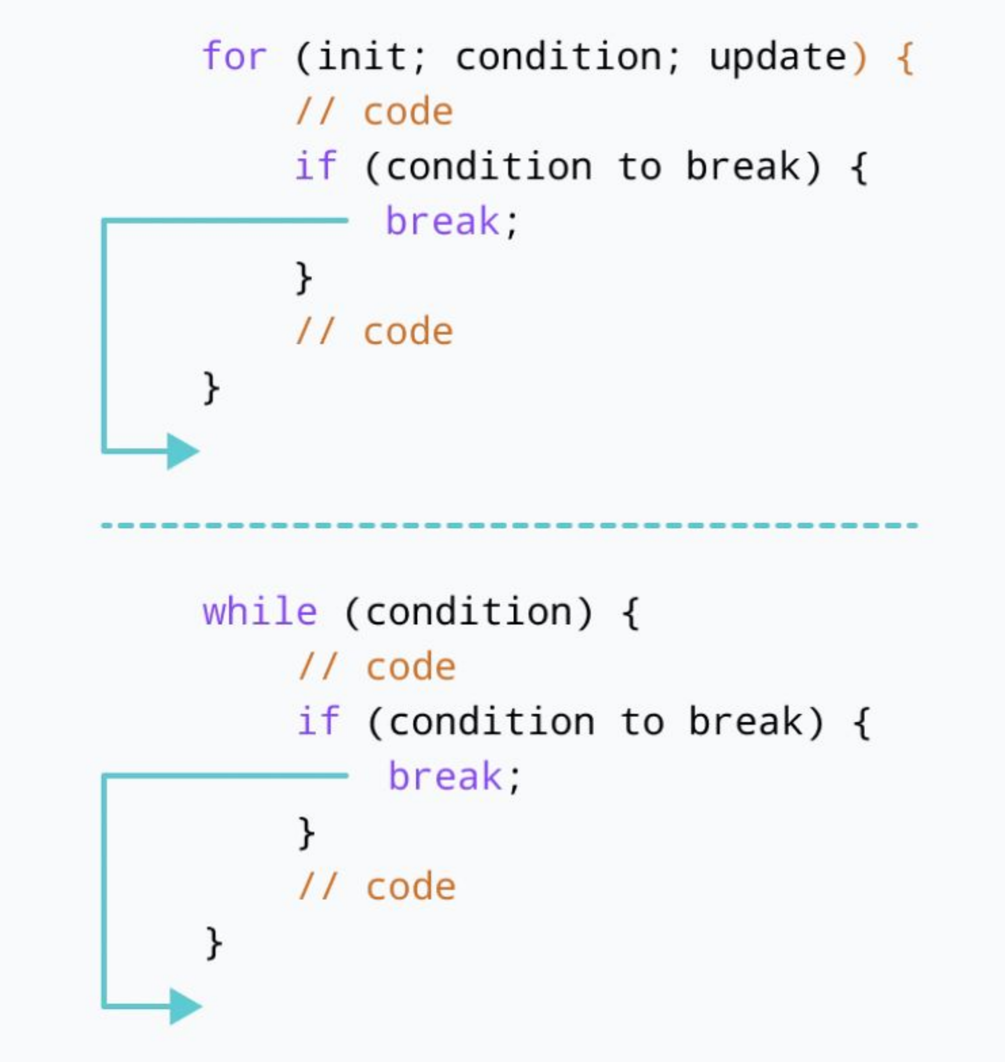
Jump statements

‘break’ statement is used to break the loop immediately



Example 1: break with for Loop

```
// program to print the value of i
for (let i = 1; i <= 5; i++) {
    // break condition
    if (i == 3) {
        break;
    }
    console.log(i);
}
```

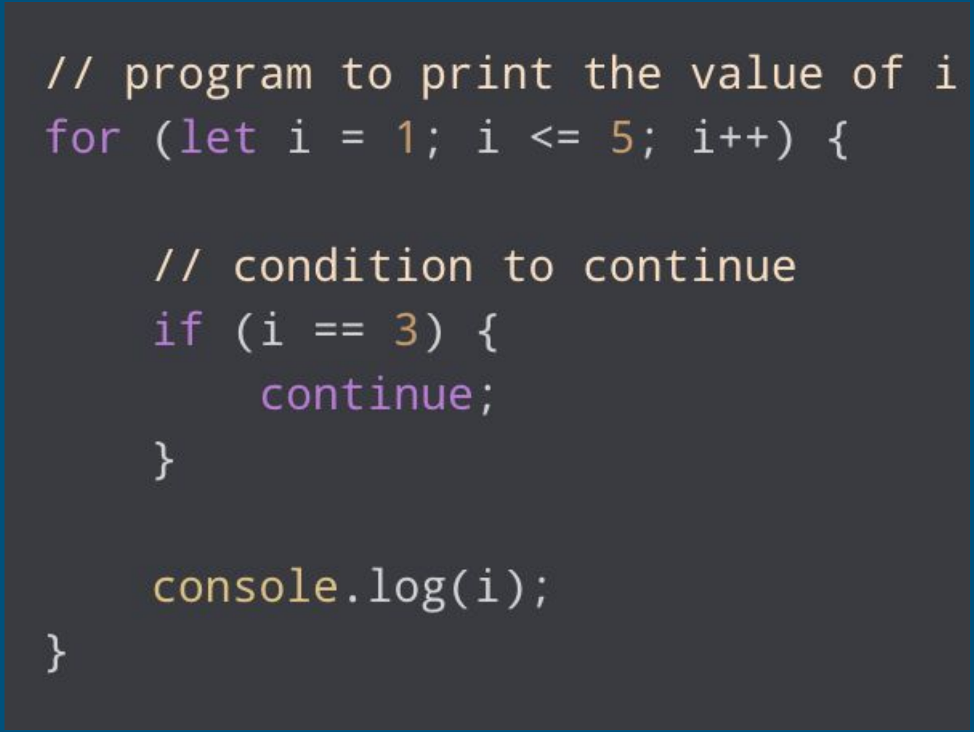


```
for (init; condition; update) {
    // code
    if (condition to break) {
        break;
    }
    // code
}
```

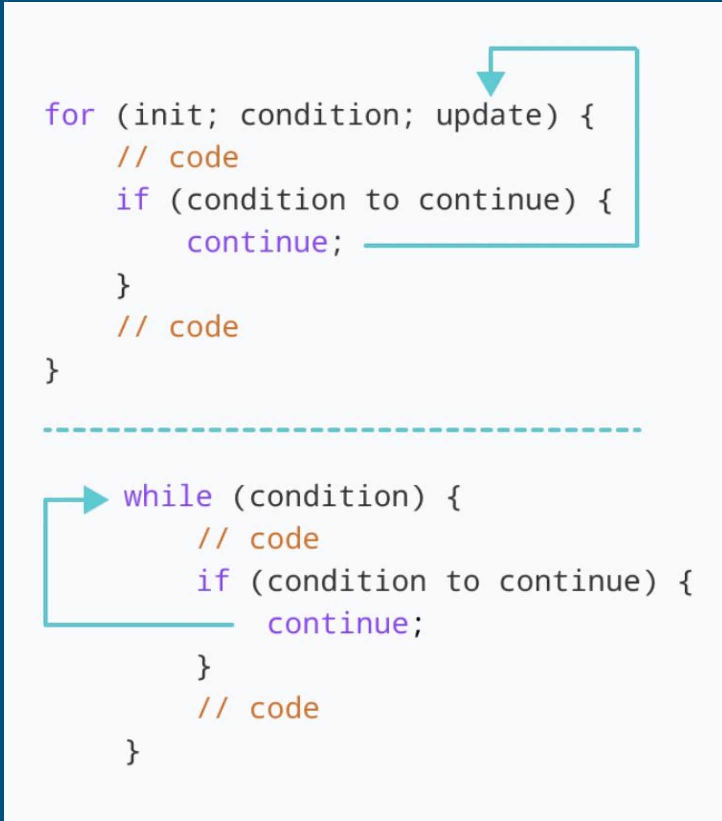
```
while (condition) {
    // code
    if (condition to break) {
        break;
    }
    // code
}
```

‘continue’ statement

The continue statement is used to skip the current iteration of the loop and the control flow of the program goes to the next iteration.



```
// program to print the value of i
for (let i = 1; i <= 5; i++) {

    // condition to continue
    if (i == 3) {
        continue;
    }

    console.log(i);
}
```



```
for (init; condition; update) {
    // code
    if (condition to continue) {
        continue;
    }
    // code
}
```

```
while (condition) {
    // code
    if (condition to continue) {
        continue;
    }
    // code
}
```