Purwadhika
Digital Technology School

Full Stack Web Development

Conditional and loop statements

Outline

Purwadhika
Digital Technology School

- Conditional statements
- Loop statements

Conditional statements

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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 = 21;

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

If statement

Purwadhika
Digital Technology School

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 = 21;

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

Else statement

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

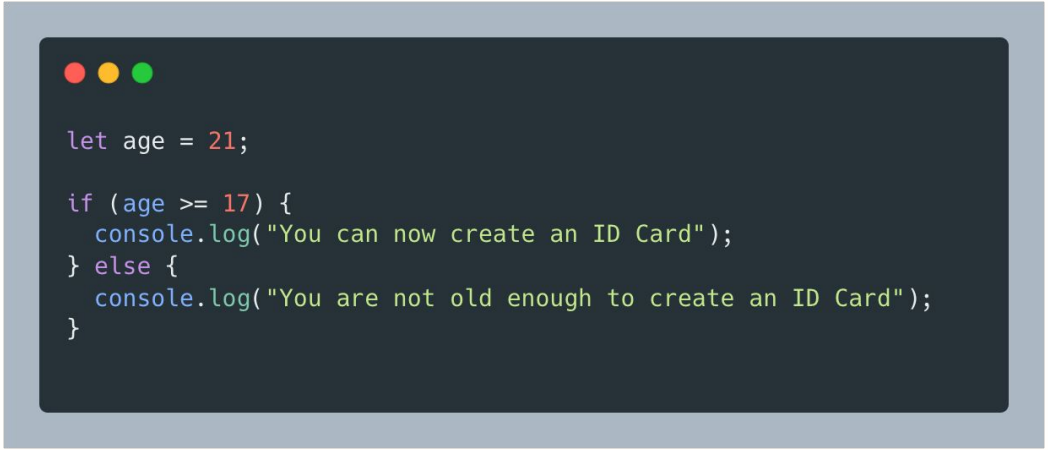
An else statement will act as a **backup plan for if statements**. It does not require a condition, it only needs a block of code to execute.

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

Else statement

Purwadhika
Digital Technology School

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"**



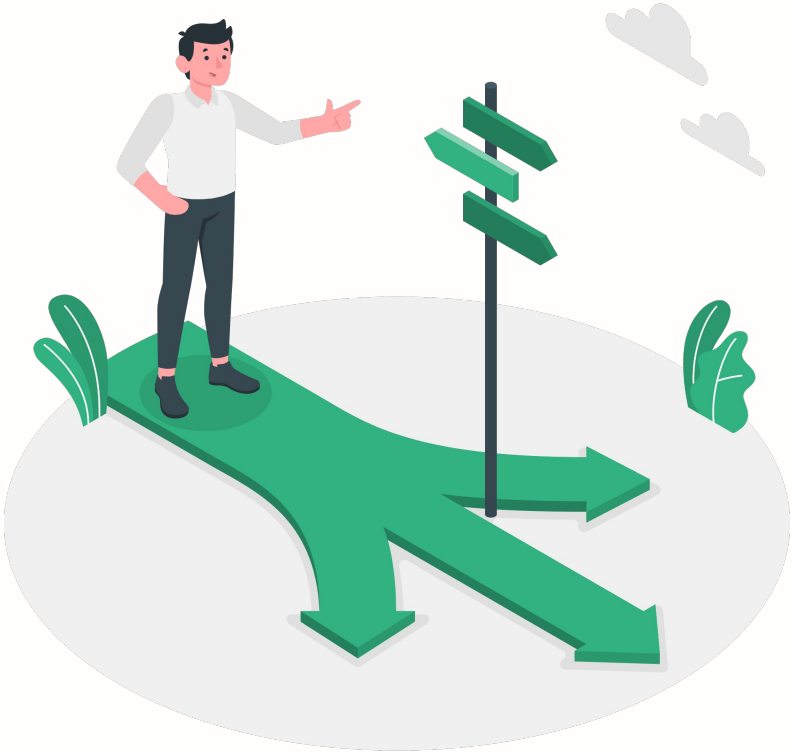
```
let age = 21;

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

Purwadhika
Digital Technology School

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**.

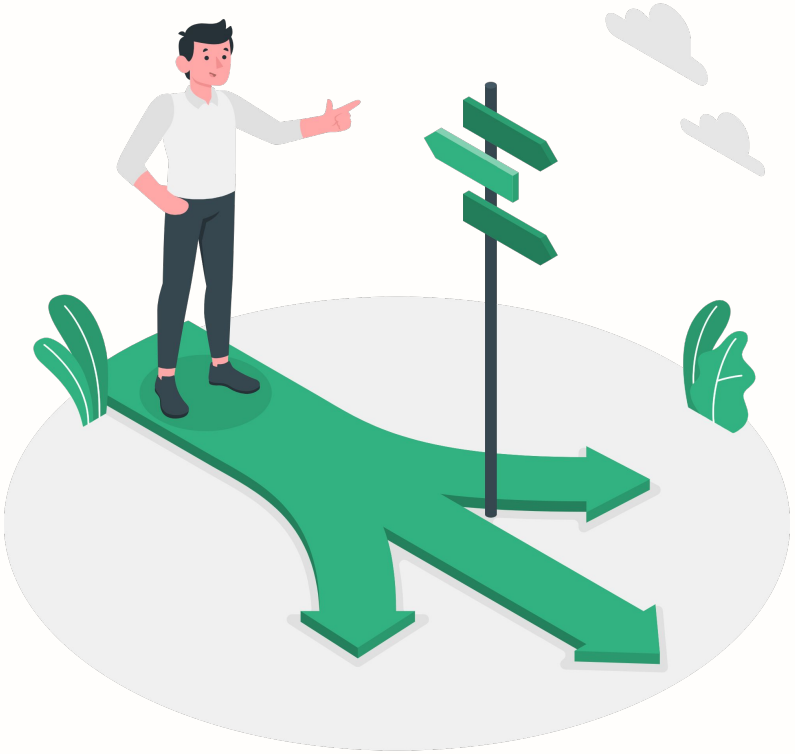


Else if statement

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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 = "B";

if (grade === "A") {
  console.log("Excelent Result!");
} else if (grade === "B") {
  console.log("Great Result!");
}
```

Chaining conditions

Purwadhika
Digital Technology School

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”.

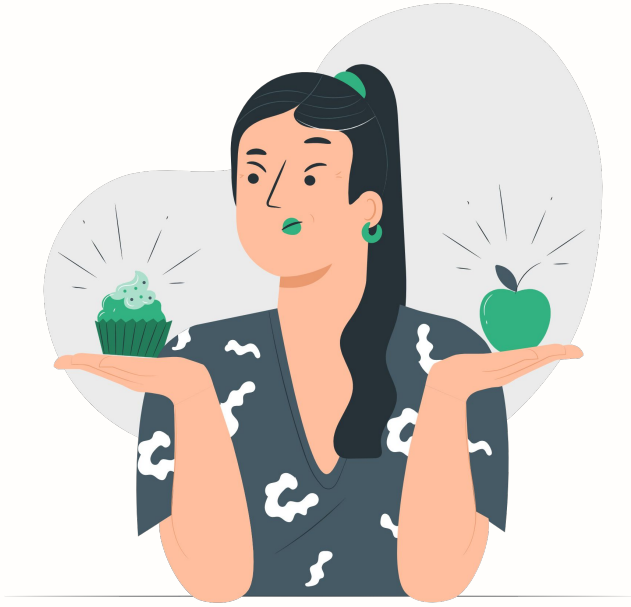
```
let grade = "B";

if (grade === "A") {
  console.log("Excelent Result!");
} else if (grade === "B") {
  console.log("Great Result!");
} else if (grade === "C") {
  console.log("Average Result!");
} else {
  console.log("Invalid Grade!");
}
```

Truthy and falsy values

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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)

Ternary Operator

Purwadhika
Digital Technology School

```
const str = "JavaScript";

if (str === "JavaScript") {
  console.log("JavaScript");
} else {
  console.log("not JavaScript");
}

// Ternary operator
console.log(str === "JavaScript" ? "JavaScript" : "not JavaScript");
```

Loop statements

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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

While loop

Purwadhika
Digital Technology School

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 loop

Purwadhika
Digital Technology School

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 = 0;

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

Do ... While loop

Purwadhika
Digital Technology School

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

Purwadhika
Digital Technology School

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

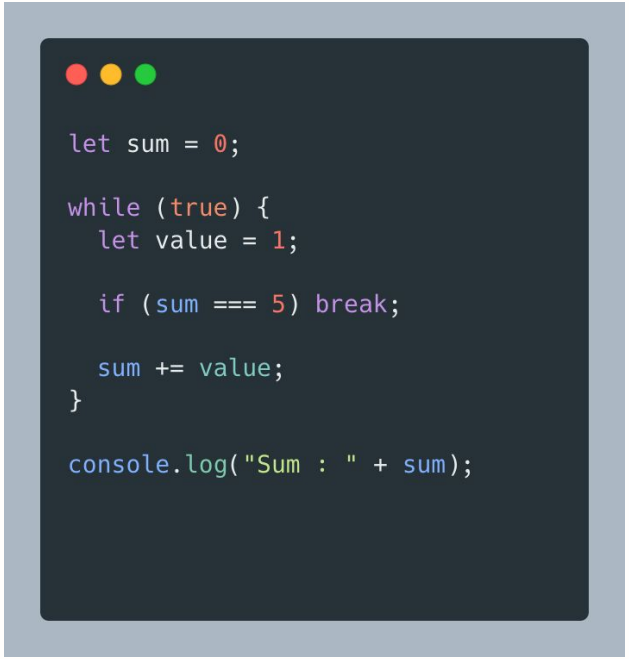
```
let i = 5;

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

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

Break

Purwadhika
Digital Technology School

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.



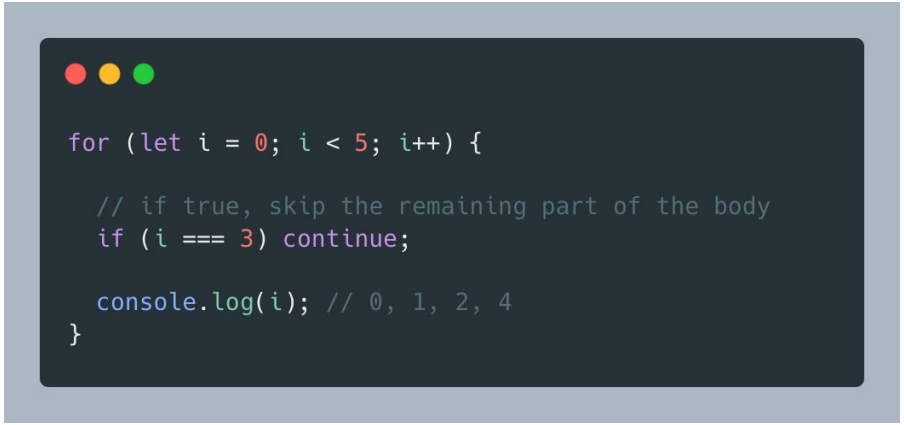
```
let sum = 0;

while (true) {
  let value = 1;

  if (sum === 5) break;

  sum += value;
}

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

Continue

Purwadhika
Digital Technology School

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
}
```

Exercise

Purwadhika
Digital Technology School

- Write a code to convert celsius to fahrenheit.
- Write a code to check whether the number is odd or even
- Write a code to check whether the number is prime number or not
- Write a code to find the sum of the numbers 1 to N.
    - Example : 5 → 1 + 2 + 3 + 4 + 5 = 15
- Write a code to find factorial of a number.
- Write a code to print the first N fibonacci numbers.

Thank You!

Purwadhika
Digital Technology School