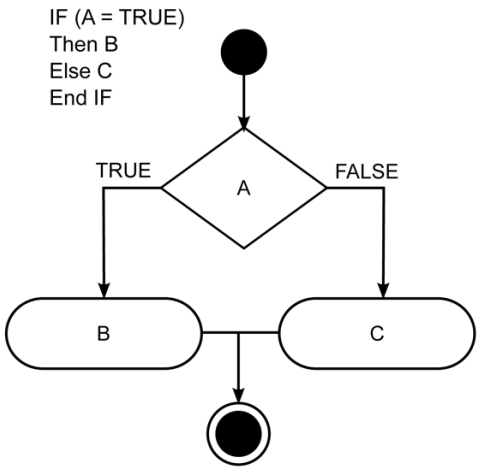
Conditionals

Conditionals are the decision-makers of your JavaScript code. They allow you to control what happens based on specific conditions.



Conditional Operators

Before we get into conditional statements, let's look at basic conditional operators that you might see in code with conditions.

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equal value
===	equal value and type
!==	not equal value or not equal type
&&	AND operator (both conditions must be true)
||	OR operator (either condition can be true)
!	NOT operator (inverts the condition)

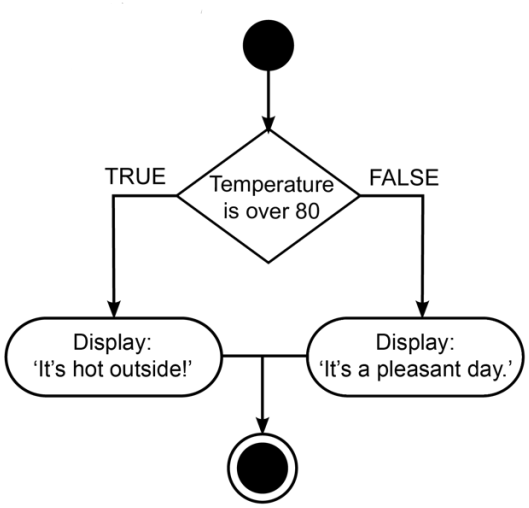
The if statement

One way to write conditionals is with the **if statement**. This statement checks if the temperature is over 80 degrees.

- If that condition is true, **"It's hot outside!"** is displayed in the console.
- However, if the condition is false, **"It's a pleasant day."** is displayed.

```
let temperature = 25;

if (temperature > 80) {
  console.log("It's hot outside!");
} else {
  console.log("It's a pleasant day.");
}
```



So, when a condition is true, the code inside the curly braces `{}` following that condition will run. If the condition is false, the code in the `else` block will execute.

Here's another example that you might see in a program that checks for user log in values.

```
const username = "student";
const password = "learning123";
```

```
let inputUser = prompt("Enter username:");
let inputPass = prompt("Enter password:");

if (inputUser === username && inputPass === password) {
    console.log("Welcome back!");
} else {
    console.log("Invalid credentials, please try again.");
}
```

The if, else if, else statement

If you want to check **more than one condition**, you can use an if…else if…else statement. In this example:

- It checks whether the temperature is over 80 degrees.
- If that condition is false, it checks for temperatures below 40 degrees.
- If neither condition is true, the else statement runs.

Once a true condition is found, the corresponding code block executes, and the program **exits the entire if statement**, skipping any remaining else if or else clauses.

```
let temperature = 25;
if (temperature > 80) {
  console.log("It's hot outside!");
} else if (temperature < 40) {
  console.log("It's really cold!")
} else {
  console.log("It's a pleasant day.");
}
```

You can add as many else if conditions as you need, but with too many, the if statement can become **tedious and messy**.

For example, when checking the days of the week, you might end up with several else if clauses:

```
let day = "Monday";

if (day = "Monday") {
    console.log("Start of the workweek!");
} else if (day == "Friday") {
    console.log("Weekend is near!");
} else if (day == "Saturday" || day == "Sunday") {
    console.log("Its the weekend!");
} else {
    console.log("Just another day.");
}
```

Switch Statement

Instead of writing numerous else if conditions, you can use a **switch statement**.
This code performs the **same task** as the if...else if...else statement but is **cleaner and simpler**.

- After the switch keyword, you specify the variable to evaluate (day).
- Each case represents a possible value.
- The code following the colon : runs if the value matches the case.

```
let day = "Monday";
switch (day) {
  case "Monday":
    console.log("Start of the workweek!");
    break;
  case "Friday":
    console.log("Weekend is near!");
    break;
```

```
    case "Saturday":
    case "Sunday":
      console.log("It's the weekend!");
      break;
    default:
      console.log("Just another day.");
  }
```

Notice the break statements inside each case.
Unlike if…else if…else statements, switch will **keep checking all cases** even after finding a match—**unless you include break**.

- The break statement **stops the execution** of the switch block once a match is found.
- Without it, the program will continue executing the remaining cases, which is often not desired.

So, there we have two ways to handle decision-making in JavaScript:

1. **if…else if…else** → For handling complex conditions and ranges.
2. **switch** → For cleaner code when comparing a single value to multiple cases.