EDUFY

if statements: make your code decide

Maximilian Bahrami · 10 November 2026

Run code only when something is true with if, else and else if, put your checks in the right order, combine them with && and ||, and use the short ? : for simple choices.

Show a message only when the form is empty. Give a discount only to students. Say "Good morning" in the morning and "Good evening" at night. All of this is done with **if statements**: code that runs only when something is true. By the end of this lesson you'll be able to write if, else and else if, put the checks in the right order, combine them, and use the short ? : version you'll see in a lot of frontend code.

if: run code only when something is true

An if checks a condition in brackets. When the condition is true, the code between { } runs. When it's false, it's skipped:

```
const temperature = 30;

if (temperature > 25) {
console.log("It's hot today!");
}

console.log("This line always runs");
```

Change 30 to 15 and press **Run** again: the first message disappears, the second one stays.

else: what to do otherwise

Add else to run other code when the condition is false. Exactly **one** of the two blocks runs, never both:

```
const age = 15;

if (age >= 18) {
console.log("You can vote");
} else {
console.log("You can vote in " + (18 - age) + " years");
}
```

else if: more than two choices

When there are more than two possibilities, put else if between them. JavaScript checks them **from the top**, and runs only the **first one** that is true:

```
const score = 72;

if (score >= 90) {
console.log("Grade: A");
} else if (score >= 75) {
console.log("Grade: B");
} else if (score >= 50) {
console.log("Grade: C");
} else {
console.log("Grade: F");
}
```

72 isn't 90 or more, it isn't 75 or more, but it is 50 or more: so you get **C**, and JavaScript stops checking. The last else catches everything that's left.

score is 95. What does the code above print?

- ✓ Grade: A
- Grade: A, Grade: B and Grade: C
- Grade: C

> 95 >= 90 is true, so the first block runs and JavaScript skips all the others. Only **one** block ever runs in an if ... else if ... else chain.

Order matters

Because the first true check wins, put the **most specific** check first. Look at this one, with the checks the wrong way round:

```
const score = 95;

if (score >= 50) {
console.log("Grade: C");
} else if (score >= 90) {
console.log("Grade: A"); // never reached for 95
}
```

95 >= 50 is true, so it prints C and never gets to the A check. When you use >=, start from the **biggest** number.

Two conditions at once

You met && (and) and || (or) in the booleans lesson. They're how you check two things in one if:

```
const age = 16;
const isStudent = true;

if (age < 18 && isStudent) {
console.log("Student discount: 50%");
}

const day = "Saturday";

if (day === "Saturday" || day === "Sunday") {
console.log("It's the weekend");
}
```

This is easier to read than one if inside another. If you find yourself writing an if inside an if, check whether && can do the job.

if on a web page

In frontend code, if is everywhere: checking what a user typed before you do anything with it. Press **Run**, then try the button with an empty box, with abc, and with your real age:

```
<!DOCTYPE html>
<html lang="en">
<body style="padding: 16px; font-family: sans-serif">
<input id="age" placeholder="Your age" />
<button id="check">Check</button>
<p id="message"></p>
<script>
const input = document.getElementById("age");
const message = document.getElementById("message");

function checkAge() {
const age = Number(input.value);

if (input.value === "") {
message.textContent = "Please type your age.";
} else if (Number.isNaN(age)) {
message.textContent = "That's not a number.";
} else if (age < 13) {
message.textContent = "Sorry, you must be 13 or older.";
} else {
message.textContent = "Welcome!";
}
```

```
console.log("age:", input.value);
}

document.getElementById("check")
.addEventListener("click", checkAge);
</script>
</body>
</html>
```

The checks go from "nothing typed" to "not a number" to "too young". Each one only runs when the ones above it were false, so by the time you reach age < 13, you know age is a real number.

The short way: ? :

For a simple "this or that" **value**, developers often use the **ternary operator** condition ? valueIfTrue : valueIfFalse:

```
const items = 1;

// The long way
let word;
if (items === 1) {
word = "item";
} else {
word = "items";
}

// The short way: the same result on one line
const word2 = items === 1 ? "item" : "items";

console.log(items + " " + word); // 1 item
console.log(items + " " + word2); // 1 item
```

Read it as: is items === 1? If yes, "item", otherwise "items".

When to use which

> Use ? : only to choose between **two values**, like a word or a colour. As soon as there's more to do, or more than two choices, use a normal if. It's much easier to read.

Common mistakes

One = instead of ===

> **Wrong**

```
if (day = "Sunday") {
console.log("Rest day");
}
```

Right

```
if (day === "Sunday") {
console.log("Rest day");
}
```

One = **sets** day to "Sunday", so the if is always true, whatever day it really was. Use === to compare.

A semicolon after the if

Wrong

```
if (age >= 18); {
console.log("Adult");
}
```

Right

```
if (age >= 18) {
console.log("Adult");
}
```

The ; ends the if right there, with nothing in it. The block below is no longer part of it, so it **always** runs.

Comparing one value with "or" the wrong way

Wrong

```
if (day === "Saturday" || "Sunday") { ... }
```

Right

```
if (day === "Saturday" || day === "Sunday") { ... }
```

|| "Sunday" doesn't compare anything: "Sunday" on its own is just a non-empty string, which is always truthy. So the if is always true. Write the full comparison on both sides.

Exercise

Good morning, good afternoon

new Date().getHours() gives the current hour, from 0 (midnight) to 23.

Write a function getGreeting(hour) that **returns**:

- "Good morning" from 5 to 11
- "Good afternoon" from 12 to 17

- "Good evening" from 18 to 21
- "Good night" the rest of the time

Test it with a few hours, then with the real time.

Answer

```
function getGreeting(hour) {
if (hour >= 5 && hour <= 11) {
return "Good morning";
} else if (hour >= 12 && hour <= 17) {
return "Good afternoon";
} else if (hour >= 18 && hour <= 21) {
return "Good evening";
} else {
return "Good night";
}
}

console.log(getGreeting(8)); // Good morning
console.log(getGreeting(14)); // Good afternoon
console.log(getGreeting(20)); // Good evening
console.log(getGreeting(2)); // Good night

const now = new Date().getHours();
console.log("Now: " + getGreeting(now));
```

"The rest of the time" is 22 to 23 **and** 0 to 4, which is hard to write as one check. The final else catches both without any maths.

Key points

- if (condition) { ... } runs the code only when the condition is true.
- else runs when it's false; else if adds more choices. Only the **first** true block runs.
- Put the most specific check first: with >=, start from the biggest number.
- Combine checks with && and ||, and write the full comparison on both sides.
- condition ? a : b picks between two values. For anything more, use if.