EDUFY

Booleans: true, false and the decisions your code makes

Maximilian Bahrami · 10 November 2026

Store a yes or no in a variable, get one from a comparison, combine them with &&, || and !, learn which values are truthy and falsy, and avoid the "false"-in-quotes trap.

Has the user won? Is it raining? Is the form filled in? Every decision your code makes comes down to a yes or a no, and in JavaScript that yes or no is a **boolean**: true or false. By the end of this lesson you'll be able to store a yes/no in a variable, get one from a comparison, combine several with &&, || and !, and know which values JavaScript treats as "true enough" in an if.

Two values: true and false

A boolean has only two possible values, true and false. You write them **without quotes**, and always in **lower case**:

```
let hasUserWon = false;
let isRaining = true;

console.log(hasUserWon); // false
console.log(typeof hasUserWon); // "boolean"
```

True, TRUE or "true" are not booleans. True is an error (JavaScript looks for a variable with that name), and "true" is a piece of text, a string.

Booleans in an if

An if runs its code when the value in the brackets is true, and the else code when it's false. Here is the lesson's starting code. Press **Run**:

```
<!DOCTYPE html>
<html lang="en">
<body>
<script>
let hasUserWon = false;
if (!hasUserWon) {
console.log("You have lost the game");
} else {
console.log("You have won the game");
}
```

```
let weatherIsRaining = true;
if (!weatherIsRaining) {
console.log("The weather is sunny");
} else {
console.log("The weather is raining");
}
</script>
</body>
</html>
```

The ! means **not**. It flips a boolean: !true is false, and !false is true. So if (!hasUserWon) reads as "if the user has **not** won".

Ask the positive question first

> if (!x) { ... } else { ... } works, but it makes the reader flip it in their head. Most teams write the positive case first:

```
if (hasUserWon) {
console.log("You have won the game");
} else {
console.log("You have lost the game");
}
```

> Keep ! for when there's only one case: if (!isLoggedIn) { showSignIn(); }.

Name them like a question

A boolean answers a yes/no question, so give it a name that reads like one. Developers start them with is, has, can or should:

```
let isRaining = true;
let hasPaid = false;
let canEdit = true;
let shouldSave = false;
```

Now if (hasPaid) reads almost like English. Names like paid, status or flag leave the reader guessing.

Comparisons give you a boolean

You rarely type true or false yourself. Most booleans come from a **comparison**:

```
const age = 17;

console.log(age >= 18); // false
console.log(age < 18); // true
console.log(age === 17); // true
```

```
console.log(age !== 17); // false
console.log("cat" === "Cat"); // false: case matters
```

Operator	Means	Example	Result
===	equal	5 === 5	true
!==	not equal	5 !== 5	false
> / <	greater / less than	3 > 2	true
>= / <=	or equal	2 <= 2	true

Because a comparison is a boolean, you can store it in a variable with a good name:

```
const age = 20;
const isAdult = age >= 18;

if (isAdult) {
console.log("You can sign up");
}
```

Combine them: &&, || and !

Write	Name	True when
a && b	and	**both** are true
a || b	or	**at least one** is true
!a	not	a is false

```
const isRaining = true;
const hasUmbrella = false;

console.log(isRaining && hasUmbrella); // false
console.log(isRaining || hasUmbrella); // true
console.log(!hasUmbrella); // true

if (isRaining && !hasUmbrella) {
console.log("You will get wet");
}
```

isLoggedIn is true and isAdmin is false. What is isLoggedIn && isAdmin?

- true

- ✓ false
- "false"
- an error

&& needs **both** sides to be true. isAdmin is false, so the whole thing is false. With || it would be true.

Truthy and falsy

An if doesn't only take booleans. Give it any value and JavaScript decides whether it counts as true (**truthy**) or false (**falsy**). Only these are falsy:

Falsy value	What it is
false	the boolean
0 (and -0)	the number zero
""	an empty string
null	"nothing here, on purpose"
undefined	"nothing set yet"
NaN	"not a number"

Everything else is truthy, including "0", "false", [] (an empty list) and {} (an empty object). Boolean(value) shows you which one a value is:

```
console.log(Boolean("")); // false
console.log(Boolean("hello")); // true
console.log(Boolean(0)); // false
console.log(Boolean(42)); // true
console.log(Boolean(null)); // false
console.log(Boolean("false")); // true: it's text!
console.log(Boolean([])); // true
```

This is why you'll see short checks like this all over real code:

```
const name = "";

if (name) {
console.log("Hello, " + name);
} else {
console.log("Please type your name");
}
```

Are true and false 1 and 0?

Almost. When JavaScript turns a boolean into a number, true becomes 1 and false becomes 0:

```
console.log(Number(true)); // 1
console.log(Number(false)); // 0
console.log(true + true); // 2

// Count how many answers are right:
const answers = [true, false, true, true];
const right = answers.filter(Boolean).length;
console.log(right + " of " + answers.length);
```

But a boolean is **not** a number. true === 1 is false, because === also checks the type. Treat booleans as yes/no, not as maths.

A switch you can click

A boolean is perfect for anything that is on or off. Press **Run**, then click the button:

```
<!DOCTYPE html>
<html lang="en">
<body style="padding: 16px; font-family: sans-serif">
<button id="toggle">Turn the light on</button>
<p id="status">The light is off</p>
<script>
let isLightOn = false;
const button = document.getElementById("toggle");
const status = document.getElementById("status");

button.addEventListener("click", () => {
isLightOn = !isLightOn; // flip it

if (isLightOn) {
document.body.style.background = "#fff6c2";
status.textContent = "The light is on";
button.textContent = "Turn the light off";
} else {
document.body.style.background = "";
status.textContent = "The light is off";
button.textContent = "Turn the light on";
}
console.log("isLightOn:", isLightOn);
});
</script>
</body>
</html>
```

isLightOn = !isLightOn is the one-line way to switch a boolean to its other value.

Common mistakes

Writing true in quotes

Wrong

```
let isDone = "false";
if (isDone) {
console.log("Finished!"); // runs!
}
```

Right

```
let isDone = false;
if (isDone) {
console.log("Finished!"); // doesn't run
}
```

> "false" is a string with five letters in it, and every non-empty string is truthy. Values from forms, the address bar and localStorage are **always** strings, so turn them into a boolean on purpose: const isDone = value === "true";

Comparing with true

Wrong

```
if (isRaining === true) { ... }
if (hasPaid == false) { ... }
```

Right

```
if (isRaining) { ... }
if (!hasPaid) { ... }
```

> A boolean already is the answer. === true only makes the line longer.

One = instead of three

Wrong

```
if (isRaining = false) {
console.log("Sunny!");
}
```

Right

```
if (isRaining === false) {
console.log("Sunny!");
}
```

> One = **sets** the value: it changes isRaining to false, and the if never runs. === **compares**. (Better still: if (!isRaining).)

Exercise

Can you open the door?

> A door opens only if the person **has a key and the door isn't locked from inside**, or if they **are the owner**.

> 1. Make three booleans: hasKey, isLockedInside and isOwner.
> 2. Make a fourth, canOpen, out of the first three with &&, || and !.
> 3. Print "The door opens" or "The door stays shut".

> Change the three values and run it again. Can you make it open in three different ways?

Answer

```
const hasKey = true;
const isLockedInside = true;
const isOwner = false;

const canOpen = (hasKey && !isLockedInside) || isOwner;

if (canOpen) {
console.log("The door opens");
} else {
console.log("The door stays shut");
}
```

> The brackets make the order clear: first "has a key **and** not locked inside", then **or** "is the owner". With these values the key isn't enough, because the door is locked from inside, so it stays shut. Set isOwner to true and it opens.

Key points

> - A boolean is true or false: no quotes, lower case.
> - Comparisons (===, !==, >, <, >=, <=) give you a boolean. Store it in a name that reads like a question: isAdult, hasPaid.
> - && is true when both are, || when at least one is, and ! flips the value.
> - Falsy: false, 0, "", null, undefined, NaN. Everything else, even "false" and [], is truthy.
> - Number(true) is 1 and Number(false) is 0, but true === 1 is false: a boolean is not a number.