

FEBRUARY 3, 2021 / #JAVASCRIPT

How the Question Mark (?) Operator Works in JavaScript



Nishant Kumar

The conditional or question mark operator, represented by a `?`, is one of the most powerful features in JavaScript. The `?` operator is used in conditional statements, and when paired with a `:`, can function as a compact alternative to `if...else` statements.

But there is more to it than meets the eye. There are three main uses for the `?` operator, two of which you might not have used or even heard of. Let's learn about them all in detail.

Three Main Uses for the Question Mark (?) in JavaScript:

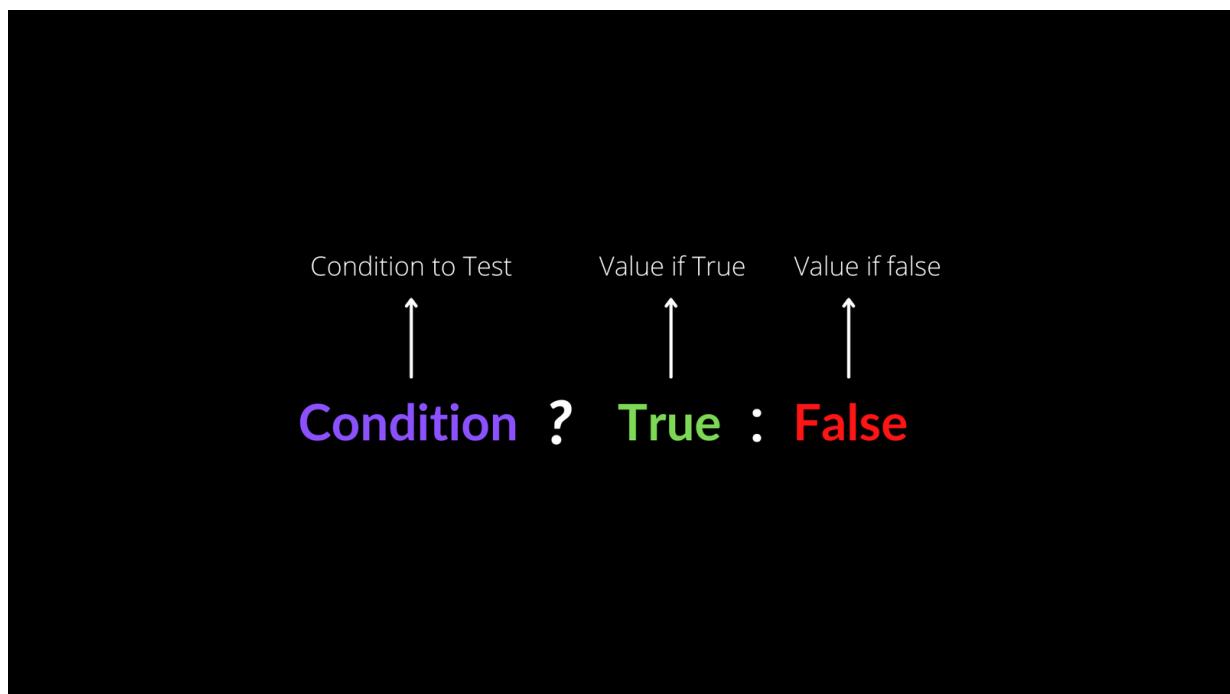
1. Ternary Operator
2. Optional Chaining
3. Nullish Coalescing

We'll look at each of these in detail, starting with the most common way you'll see the `?` operator being used – as a ternary operator.

Learn to code — free 3,000-hour curriculum

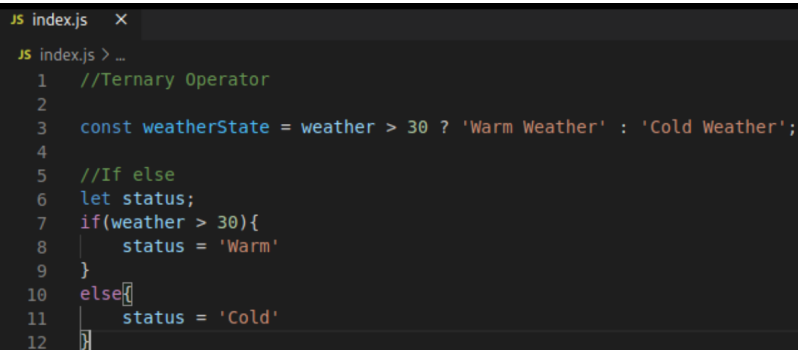
The term **ternary** means composed of three items or parts. The `?` operator is also called the ternary operator because, unlike other operators such as strict equal (`===`) or remainder (`%`), it's the only one that takes three operands.

Starting with `?`, we add a condition on the left side and a value on the right side to return when the condition is true. Then we add a colon (`:`) followed by a value to return if the condition is false.



The ternary operator is basically a shortcut for a traditional `if...else` statement.

Let's compare a ternary operator with a longer `if...else` statement:



```
JS index.js X
JS index.js > ...
1 //Ternary Operator
2
3 const weatherState = weather > 30 ? 'Warm Weather' : 'Cold Weather';
4
5 //If else
6 let status;
7 if(weather > 30){
8   status = 'Warm'
9 }
10 else{
11   status = 'Cold'
12 }
```

Here, the ternary operator occupies only one line of code, whereas the `if...else` takes seven lines.

Using a ternary operator is much more effective, right?

2. Optional Chaining

In 2020, an awesome new feature known as Optional Chaining was introduced.

To understand how it works, imagine this scenario.

Let's say you have code that calls on an object property that doesn't exist, which triggers an error at run time. This may be because of a missing or undefined value in your database or from an API:

Learn to code — free 3,000-hour curriculum

```
JS index.js X
JS index.js > ...
1  const user = {
2    name: 'Nishant',
3    age: 24
4  }
5
6  user.write.salary();|
```

```
user.write.salary();
      ^
```

```
TypeError: Cannot read property 'salary' of undefined
    at Object.<anonymous> (/home/nishant666/Desktop/My Apps/Blogs/Use of Ternary Operator/index.js:6:12)
    at Module.compile (internal/modules/cjs/loader.js:1137:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1157:10)
    at Module.load (internal/modules/cjs/loader.js:985:32)
    at Function.Module._load (internal/modules/cjs/loader.js:878:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12)
```

A common error – TypeError: Cannot read property 'salary' of undefined

Thanks to Optional Chaining, you can just insert a `?` between the property name and the period between the next property.

Learn to code — free 3,000-hour curriculum

```
JS index.js > ...  
1   const user = {  
2     name: 'Nishant',  
3     age: 24  
4   }  
5  
6   user.write?.salary();|
```

With that, it will just return `undefined` instead of throwing an ugly error.

Optional Chaining is truly a life-changing feature for JavaScript developers.

3. Nullish Coalescing

In some cases, you have to set a default value for a missing property name or value.

For example, let's say we are creating a Weather App in which we are fetching the temperature, humidity, wind speed, pressure, time of sunrise and sunset, and the picture of the city. We inputted a place, let's say *Bangalore*, but for some reason, its image is not there in the database.

When the app fetches and displays the data, the picture will be blank, which can look ugly. What we can do, in that case, is set a default picture for those cities which don't have an image, Bangalore in our case.

This way, when the app displays the data, the default picture will be there for the cities without images.

Learn to code — free 3,000-hour curriculum



But if you use `||` to provide a default value, you may encounter unexpected behaviors if you consider some values as usable (for example, `' '` or `0`).

Consider a scenario where a variable has the value of `0` or an empty string. If we use `(||)`, it will be considered as undefined or NULL and return some default value that we have fixed.

Instead of the logical OR `(||)` operator, you can use double question marks `(??)`, or Nullish Coalescing.

Let's learn with an example.

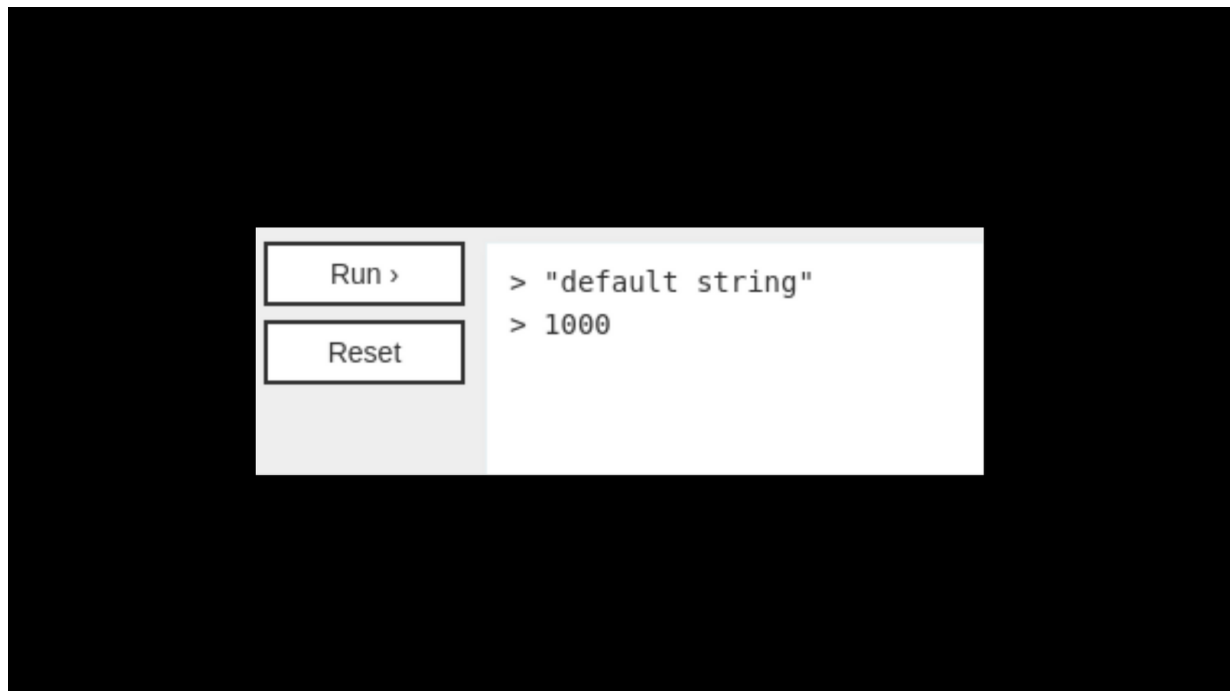
```
const value1 = 0 || 'default string';
console.log(value1);
```

```
const value2 = '' || 1000;
console.log(value2);
```

Learn to code — free 3,000-hour curriculum

Here, we have '0' and 'default string' in variable value1. If we log its value in the console, we will get 'default string', which is weird. Instead of the default string, we should be getting 0, as zero is not undefined or null. So, ' || ' fails to do the job here.

Similarly, it's the same with value2.

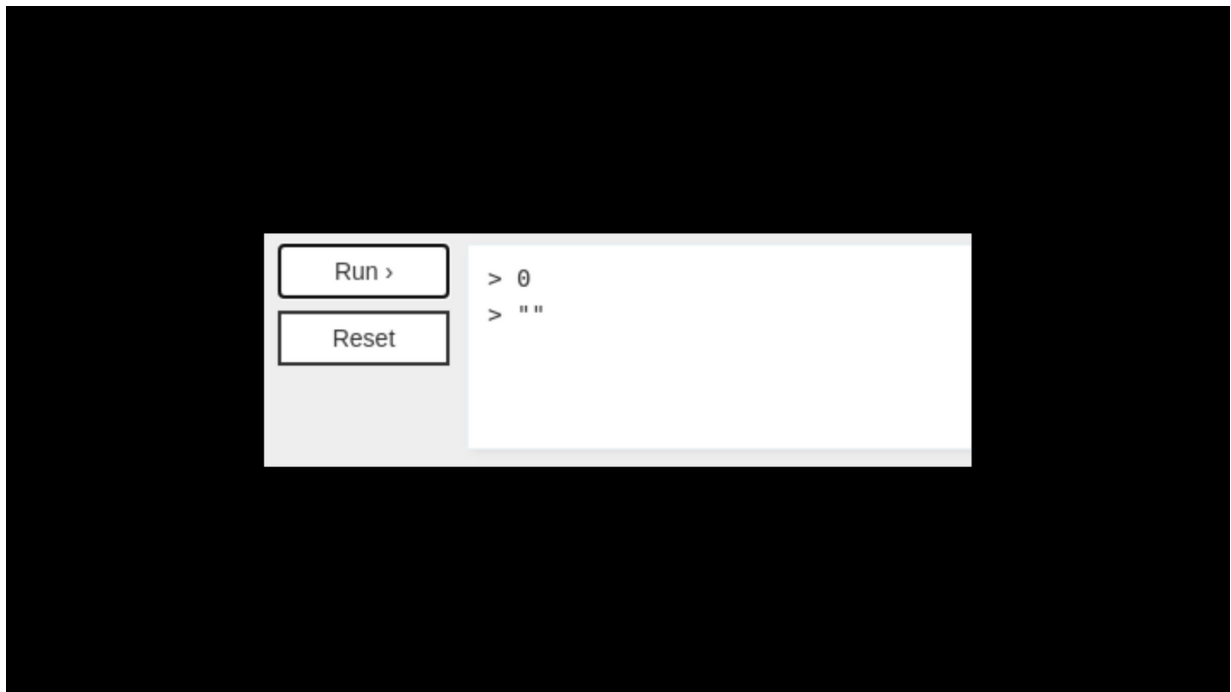


Output for ' || '

```
const value1 = 0 ?? 'default string';  
console.log(value1);
```

```
const value2 = '' ?? 1000;  
console.log(value2);
```

With Nullish Coalescing



Output for ' ?? '

Nullish Coalescing works exactly like the logical OR operator, except you will get the right side value when the left side value is `undefined` or `null`.

In other words, `??` only allows `undefined` and `null` values, not empty strings (`''`) or `0` s.

Conclusion

Now hopefully you understand how the `?` operator works in JavaScript. It looks simple, but it's one of the most powerful characters in the language. It provides syntactic sugar in three awesome but different ways.

Try them them out and let me know how it goes.

Happy learning!

Learn to code — free 3,000-hour curriculum



poetry and stories, playing the piano, and cooking delicious meals.

If you read this far, tweet to the author to show them you care.

[Tweet a thanks](#)

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers. [Get started](#)

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charity organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can [make a tax-deductible donation here](#).

Trending Guides

Hello World in Java

Python Set

What is a 429 Error?

What is REST?

Inline Style in HTML

What is CRUD?

Python Absolute Value

%.2f in Python

Java Switch Statement

Git Reset Origin

Change Font with HTML

What is Localhost?

502 Bad Gateway Error

JS Array.includes()

Static Keyword in Java

Compare Dates in JS

Learn to code — free 3,000-hour curriculum

[Create a Set in Python](#)[What's a Data Analyst Do?](#)[Remove a Dir in Linux](#)[Refresh Page in JavaScript](#)[What is Coding Used For?](#)[Remove Underline from Link](#)[JavaScript String Format](#)[How to Clear an Array in JS](#)[JavaScript String to Date](#)[Capitalize 1st Letter in JS](#)

Our Charity

[About](#)[Alumni Network](#)[Open Source](#)[Shop](#)[Support](#)[Sponsors](#)[Academic Honesty](#)[Code of Conduct](#)[Privacy Policy](#)[Terms of Service](#)[Copyright Policy](#)