

Null & Undefined

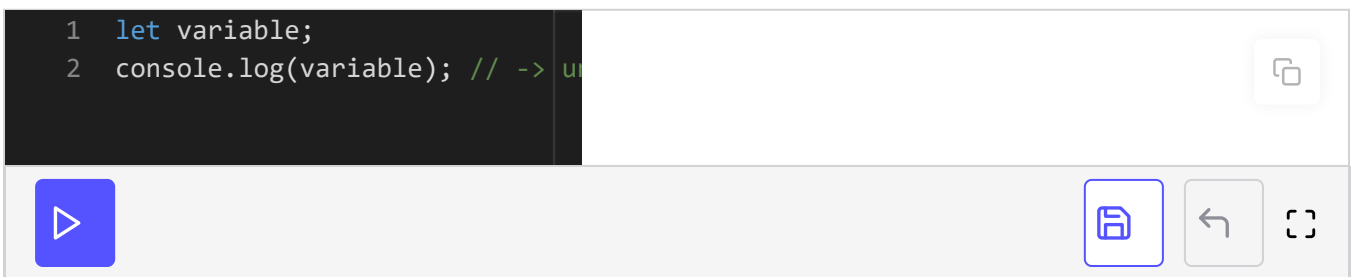
Learn two more variable types in JavaScript: null and undefined. Learn what they represent, how they differ, and how to use them.

There are two more variable types that we should discuss.

undefined

`undefined` is meant to represent the idea that something doesn't exist. When we try to use a variable that has no value, we get `undefined`. Here's an example.

```
1 let variable;  
2 console.log(variable); // -> undefined
```

A code editor interface with a dark theme. The code area shows two lines: '1 let variable;' and '2 console.log(variable); // -> undefined'. To the right of the code is a light gray area for output. Below the code area is a toolbar with a blue play button, a blue save icon, a gray back arrow, and a gray full-screen icon.

When we declare a variable using `let` but don't give it a value, it receives the default value of `undefined`. This is JavaScript telling us that we're trying to use something that isn't there.

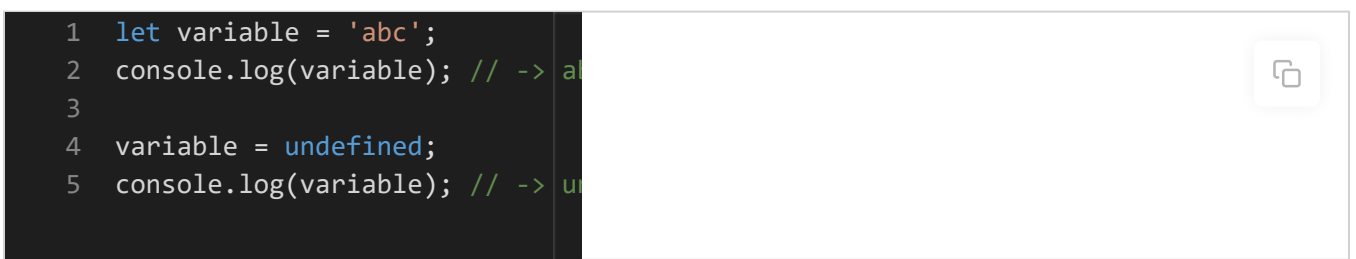
We'll see this pop up again in future lessons, following the same guideline.

`undefined` is a special variable type used to indicate that something is missing. If we see it, it usually means something is wrong with our code.

For example, we'd have no reason to use a variable if we haven't given it a value, so when we do try to use it, JavaScript gives us `undefined`.

We can use it ourselves and assign the value to variables directly.

```
1 let variable = 'abc';  
2 console.log(variable); // -> abc  
3  
4 variable = undefined;  
5 console.log(variable); // -> undefined
```

A code editor interface with a dark theme. The code area shows five lines: '1 let variable = 'abc';', '2 console.log(variable); // -> abc', '3', '4 variable = undefined;', and '5 console.log(variable); // -> undefined'. To the right of the code is a light gray area for output. Below the code area is a toolbar with a blue play button, a blue save icon, a gray back arrow, and a gray full-screen icon.



Although we can do this, we pretty much never want to assign a variable a value of `undefined`. It's meant to show that something has gone wrong and some value is missing.

If we want to specify that something has no value, we should use `null`.

`null`

`null` is another variable type. It represents something that's empty. The difference between `null` and `undefined` lies in they're implemented by JavaScript and used by developers.

`null` is something that is safe to use and to assign to variables.

```
let variable = null;  
console.log(variable); // -> null
```



`undefined` vs `null`

As mentioned, the JavaScript engine will make variables that have no value `undefined`. If we want to specify that a variable should be empty, the “correct” way to do so is to make it equal to `null`.

Code will work whether we use `undefined` or `null`. However, the development community often has a set of standard “best practices” to make code easier to read, interpret, and debug.

Preferring `null` over `undefined` is one of those best practices. Setting a variable to `null` is clear and communicates that we want to essentially delete the variable. We're done with it.

Setting it to `undefined` will make people look twice. They'll see it, be confused, and might look down upon the failure to follow common practice.

At this point, the concept of `null` and maybe `undefined` may seem confusing. When would we ever want to set a variable to `null`? These questions will be answered later.

answered later.

At this point, just know that these are two more variable types in JavaScript. We won't be using them too much.

Quiz

Feel free to test your understanding.

1

What will this code print?

```
let variable;  
let variableCopy = variable;  
variableCopy = 'def';  
console.log(variable);
```

- ☐ A) def
- ☐ B) null
- ☐ C) undefined
- ☐ D) error

2

What is the best synonym for `null`?

- ☐ A) empty

☐ B) missing

☐ C) undefined

3

What is the best synonym for `undefined`?

☐ A) empty

☐ B) missing

☐ C) null

CHECK ANSWERS