JavaScript globalThis

Summary: in this tutorial, you'll learn how to about the JavaScript `globalThis` object.

Introduction to the JavaScript globalThis object

ES2020 introduced the `globalThis` object that provides a standard way to access the global object across environments.

Historically, JavaScript had a global object with different names in different environments.

In web browsers, the global object is `window` or `frames` .

However, the Web Workers API doesn't have the `window` object because it has no browsing context. Hence, the Web Workers API uses `self` as a global object.

Node.js, on the other hand, uses the `global` keyword to reference the global object.

Environment	Global
Web Browsers	this
Web Workers	self
Node.js	global

If you write JavaScript code that works across environments and needs to access the global object, you have to use different syntaxes like `window` , `frames` , `self` , or `global` .

To standardize this, ES2020 introduced the `globalThis` that is available across environments.

For example, the following code checks if the current environment supports the Fetch API:

```
const canFetch = typeof globalThis.fetch === 'function';

console.log(canFetch);
```

The code checks if the `fetch()` function is a property of the global object. In the web browsers, the `globalThis` is the `window` object. Therefore, if you run this code on a modern web browser, the `canFetch` will be `true` .

The following code returns true on the web browser:

```
globalThis === window
```

Output:

```
true
```

Summary

- Use the `globalThis` object to reference the global object to make the code work across environments.