JavaScript Scope
Temporal Dead Zone

Learning objective: By the end of this lesson, students will understand the concept of the Temporal Dead Zone in JavaScript, especially in the context of variables declared with let and const.

As explained previously in this module, `let` and `const` work differently than `var` in regards to hoisting. If we try to access a `var` variable before declaring or initializing it, the variable will simply come back `undefined`, as the declaration itself has been hoisted (as shown in previous examples).

However, if we try to access a `let` or `const` variable prior to declaring or initializing it, we'll run into a very specific `Reference Error: "Cannot access _ before initialization".` This is called a **Temporal Dead Zone** error, and it is caused by the fact that `let` and `const` declarations are attached to the block scope but are not initialized until they actually appear in the block.

Let's look at an example:

```
// This will log `undefined` to the console.
console.log(varVariable);
var varVariable = 10;

// This will throw a `ReferenceError`: `Cannot access letVariable before initialization`.
console.log(letVariable);
let letVariable = 20;
```

Copy

The **temporal dead zone** ensures that `let` and `const` variables cannot be accidentally accessed before they are initialized. This can help to prevent errors and make our code more predictable.

Check out the MDN docs for more details.

© 2025 General Assembly
Attributions