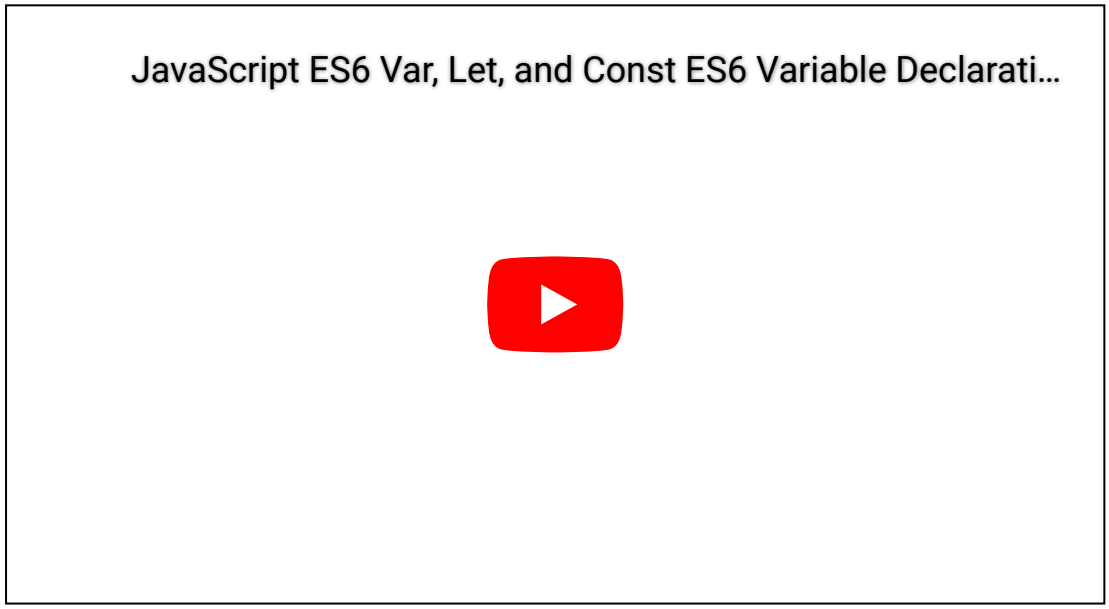
Variables

As we already know, we declare a variable in JavaScript by using the keyword **var**.

Using ES6 offers us another way to declaring our variables.

ES6 Conventions:

- Use `const` by default.
- Use `let` if you have to rebind a variable.
- Use `var` to signal untouched legacy code.



Variable declaration: Var

When using **var** inside blocks like **for** loops, **while** loops, **if** statement, and others, the variable will always be allocated until the end of the execution (even after the end of the block). This allocation can lead to **memory leak** and **unexpected bugs**.

Notice in the example below that the variable i is still defined even after the **for** loop :

```
for(var i = 0; i < 6; i++){
 var myNumber = i;
 console.log(myNumber); // 0, 1, 2, 3, 4, 5
```

```
}

console.log(myNumber); // 5
```

Variable declaration: Let

ES6 provides a new declaration keyword called "**let**".

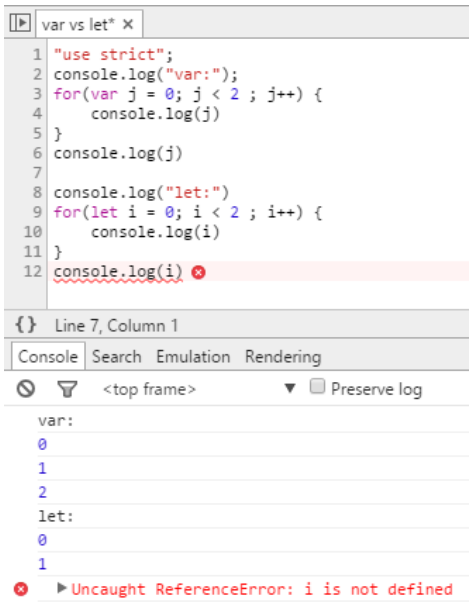
Whenever we declare a variable with **let**, it will be defined only inside the block that contains the declaration.

In other words, it's only defined inside the curly brackets **{}** that contains the **let** declaration.

```
for(let i = 0; i < 6; i++){
 console.log(i); // 0, 1, 2, 3, 4, 5
}
console.log(i); // undefined
```

Variable declaration: Let vs Var

This is a comparison between the scope let and var keywords in declaring a variable :



Variable declaration: const

ES6 also introduces the keyword **const** which allows us to declare constants.

When we are dealing with fixed values, using **const** will reduce bugs and unexpected behaviors.

Attempting to change variables, which are declared with const, will raise an error.

```
const numberOfHoursInADay = 24;

// NOT ALLOWED TO DO THIS

numberOfHoursInADay = 25; // TypeError is raised
```

Previous next