# Let and Const keywords in JavaScript

In JavaScript, the **let** and **const** keywords are used for declaring variables. They were introduced in the ECMAScript 6 (ES6) specification to provide more flexible and controlled variable scoping and assignment behavior compared to the older **var** keyword. Here's an overview of how **let** and **const** work:

1. **let**: The **let** keyword is used to declare a variable that can be reassigned. Variables declared with **let** have block-level scope, which means they are limited to the block, statement, or expression where they are defined. They are not accessible outside that block. This makes the code more predictable and helps avoid unintended variable hoisting issues.

   ```
   let age = 25;

   if (true) {
     let age = 30; // This variable is separate from the outer 'age'
     console.log(age); // Outputs 30
   }

   console.log(age); // Outputs 25
   ```

2. **const**: The **const** keyword is used to declare a constant variable that cannot be reassigned after its initial value is assigned. Like **let**, **const** variables also have block-level scope. However, the key difference is that once a value is assigned to a **const** variable, it cannot be changed.

   ```
   const pi = 3.14159;

   // pi = 3.14; // This will result in an error
   ```

   Note that when a **const** variable holds an object or an array, the variable itself is constant, but the properties or elements within the object or array can still be modified.

   ```
   const person = {
     name: 'John',
     age: 30
   };

   person.age = 31; // This is allowed

   // person = {}; // This will result in an error
   ```

In general, it's recommended to use **const** by default and switch to **let** only when you need to reassign a variable's value. This can help prevent accidental variable reassignment and make your code more readable. Use **let** when you explicitly intend for a variable to be reassigned. You will learn about let and const more in detail in upcoming weeks./