Lesson:

let and const

Topics Covered

1. Introduction to variable
2. Need of let, const & var
3. Reassigning Values
4. Basic difference between let, var and const

Introduction to variable

Variables are containers for pieces of data,they are used to store the information. Variables store data of any data type that can be used throughout a program.

Need of let, const & var

In JavaScript, we need to first declare a variable with one of three keywords:

- var
- let
- const

JavaScript

```
var name = "Prabir"
let name = 'Prabir'
const name = "Prabir"
```

Unset

```

```

Reassigning Values

If we want to change a value, we can do that here because we're using the keyword **"var"**. When it comes to changing a number, directly, we can't use "const". **"Const"** means the value is constant and cannot be changed. So, we have to use **"var" or "let"** if we want to change the value of a variable.

JavaScript

```
var FirstName = 'Prabir';
```

JavaScript

```
let x = 100;
```

Let’s reassign x to 300

```
JavaScript
x = 300
```

Now, in some cases, you may want to simply declare a variable and not assign a value to it.

```
JavaScript
let score;
```

In this case the score is equal to “undefined”. If we want to assign a value to it, we can

```
JavaScript
let score = 10
```

Constants:

Let's talk about const, which works in a slightly different way compared to let or var. So, if I create a name like this.

```
JavaScript
const x = 400;
```

And then , we reassign that value

```
JavaScript
x = 300 ; // results in error
```

We get an error, because we can’t directly re-assign a value to a constant . We also can’t initialize a constant as undefined. Let's take an example to understand better.

```
JavaScript
const score1 // results in error
```

Basic difference between let, var and const

- When we declare variables using the **var** keyword, it can be accessed within the function or globally, hence called **function scoped** (scope defines boundaries in the program where any variable can be accessed).
- When we declare a variable using the **let** keyword, it will be accessed only within the block in which it is declared, hence called **block scoped** (scope defines boundaries in the program where any variable can be accessed).
- Using const declaration, we define constants (which does not change later), and they are like block scoped.

var	let	const
Function Scoped: Variables accessible within the function block in which they are defined.	**Block Scoped:** Variables accessible within the block in which they are defined.	**Block Scoped:** Variables accessible within the block in which they are defined.
Initialized with value **undefined**	Unlnitialized	Unlnitialized
We can reassign values to variables.	We can reassign values to variables.	We cannot reassign values to variables.

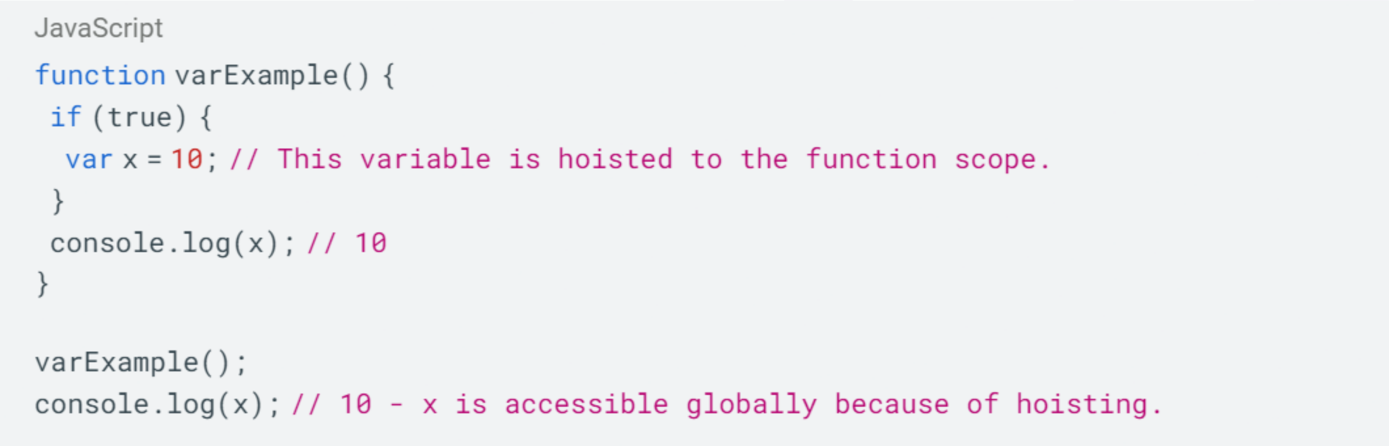
Let's take some examples to understand better:

var:

Variables declared with var have function scope or global scope, but not block scope.

They are hoisted, meaning they are moved to the top of the function or global context during compilation.

Example:



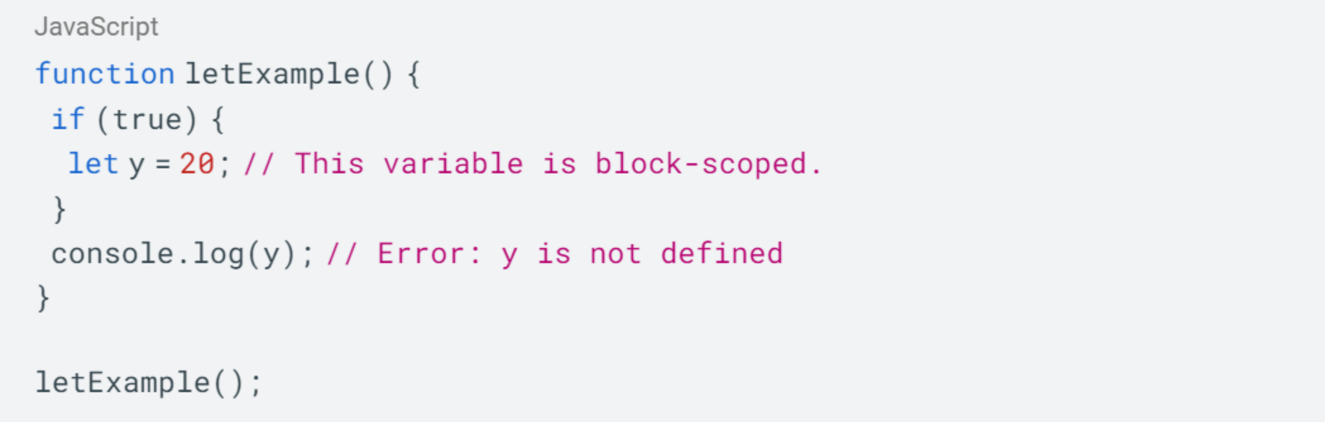
JavaScript

```
function varExample() {
 if (true) {
  var x = 10; // This variable is hoisted to the function scope.
 }
 console.log(x); // 10
}

varExample();
console.log(x); // 10 - x is accessible globally because of hoisting.
```

let:

Variables declared with let have block scope, which means they are only accessible within the block they are declared in.



JavaScript

```
function letExample() {
 if (true) {
  let y = 20; // This variable is block-scoped.
 }
 console.log(y); // Error: y is not defined
}

letExample();
```

const:

Variables declared with const also have block scope.
They must be initialized with a value, and once assigned, their value cannot be changed.

JavaScript

```javascript
function constExample() {
 if (true) {
  const z = 30; // This variable is block-scoped and cannot be
reassigned.
 }
 console.log(z); // Error: z is not defined
}

constExample();
```