JavaScript Variables

Topics Covered

- Variables
- Types of Variables
    - Global variables
    - Local variables
- Different Ways of Declaring Variables
    - Using let
    - Using const
    - Using var
    - Using nothing

Topics in Detail

Variables

- **Variables** are the data storing containers.
- Values in the variable may vary..
- JavaScript variables must have **unique names**.
- The keywords **var**, **let** and **const** are used to declare the variables.
- The same variable should not be redeclared twice.
- JavaScript is an untyped language, i.e a variable can hold any data type values.
- Syntax:

```
var money;
var name;
```

- Variables can be declared in multiple lines or even in single line with only one **var** keyword as below

```
var money, name;
```

- The variables can be initialized at the time of variable creation or at any point when you need that variable.

Types of Variables

JavaScript variables are of two types based on the scope of the variable where it is defined

- Global variables
- Local variables

Global Variable

- The Global variable can be defined anywhere in the JS code.

```
var data=200;//gloabal variable declaration
function a(){
document.writeln(data);
}
```

Local Variable

- The Local variable will be visible only within a particular function where it is defined.
- Parameters of a function are always local to that particular function.

```
function b(){
var data=50;//local variable declaration
document.writeln(data);
}
```

Different ways to declare variables

Declare JavaScript variables using any of the below keywords.

Until 2015, **the var** keyword was used to declare JavaScript variables.

let and **const** keywords were added to JavaScript later.

- Using **let**
- Using **var**
- Using **const**
- Using **nothing**

Using 'let' keyword

- If we try to **Redeclare global** variables with the **let** keyword, we will get a **syntax error**.

```
let x = "John Doe";

let x = 0;

// SyntaxError: 'x' has already been declared
```

- After ES6, JavaScript keywords **'let'** and **'const'** provide **block scope**.

```
{
  let x = 2;
}
// x can NOT be used here
```

- This variable cannot be accessed outside the block

- Variables defined with **let** must be declared before use.

- If we declare with **'let'** keyword, the value of the variables can **change**.
- Redeclaring Variables globally and local simultaneously with let keyword won't impose a problem

```
let x = 10;
// Here x is 10

{
let x = 2;
// Here x is 2
}

// Here x is 10
```

Redeclaring a variable inside a block won't actually redeclare the variable outside the block

Using 'var' keyword

- We can **Redeclare** variables with the **var** keyword.

```
var x = "John Doe";

var x = 0;
```

- Variables declared with **var** keyword don't have **block scope**.

```
{
  var x = 2;
}
// x CAN be used here
```

This variable can be accessed outside the block.

- Redeclaring Variables globally and local simultaneously with **var** keyword will impose a problem

```
var x = 10;
// Here x is 10

{
var x = 2;
// Here x is 2
}

// Here x is 2
```

Redeclaring a variable inside a block will actually redeclare the variable outside the block

Using 'const' keyword

- We **cannot Redeclare** the variables defined with **const** keyword in the same scope. But, we can redeclare the **const** keyword variables in another block or scope.

```
const x = 2;       // Allowed

{
  const x = 3;   // Allowed
}

{
  const x = 4;   // Allowed
}
```

- We cannot **reassign** the variables defined with **const** keyword. It must be assigned at the time of declaration itself.

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

- Always, variables that remain unchanged will be declared with the **const** keyword.
- Variables defined with **const** have Block Scope.

```
const x = 10;
// Here x is 10

{
const x = 2;
// Here x is 2
}

// Here x is 10
```

- **Const** keyword is mostly used to declare
  - New Array
  - New Object
  - New Function
  - New RegExp

- **const** keyword defines a constant reference to a value, but not a constant value.
- We cannot **reassign** a constant value, constant array or constant object, but we can **change the elements** of a constant **array** or change the **properties** of a constant **object**.

Example

```
// You can create a const object:
const car = {type:"Fiat", model:"500", color:"white"};

// You can change a property:
car.color = "red";
```