Variables And Data Types

Data Types

There are 7 data types available in JavaScript. You need to remember that JavaScript is loosely typed(or dynamic language), so any value can be assigned to variables in JavaScript.

According to the latest ECMAScript standard, there are 6 primitive data types and 1 non-primitive object.

Primitive	Non-Primitive
Number - represents integer and floating values	Object - represents key-value pair
String - represents textual data	
Boolean - logical entity with values as true or false.	
Undefined - represents a variable whose value is not yet assigned.	
Null - represents the intentional absence of value.	
Symbol - represents a unique value.	

All of them except Object have immutable values, i.e. the values which cannot be changed.

We'll study them in-depth in the next module

Variables

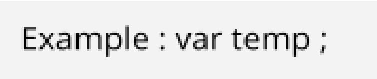
Variables are named containers for storing data (values). You can place data into these containers and then refer to the data simply by calling the container.

There are 3 ways to declare a JavaScript variable:

- Using **var**
- Using **let**
- Using **const**

Variable Declaration

Variable declaration is only declaring a variable with a variable name and not assigning any value to it.

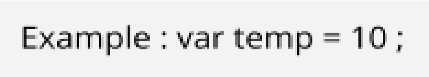


```
Example : var temp ;
```

Here temp is the variable name without any value. By default, it is assigned with **undefined.**

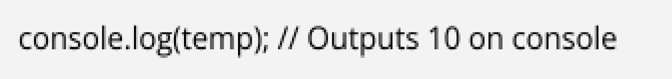
Variable Definition

Variable definition is a declaration of a variable with a variable along with assigning a value to it.



```
Example : var temp = 10 ;
```

Now, the temp variable is assigned with a value of 10



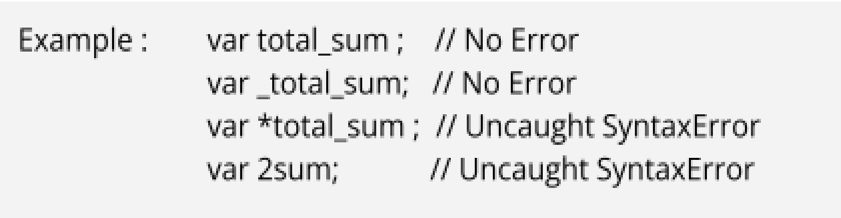
```
console.log(temp); // Outputs 10 on console
```

JavaScript Identifiers

JavaScript variables are identified with some unique names.
These unique names are called identifiers.

Rules for assigning unique identifiers to a variable:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
- Names are case sensitive (temp and Temp are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

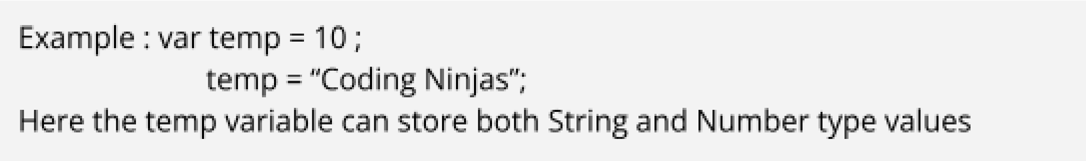


```
Example :      var total_sum ;    // No Error
               var _total_sum;   // No Error
               var *total_sum ;  // Uncaught SyntaxError
               var 2sum;         // Uncaught SyntaxError
```

Assigning values to a variable

JavaScript is an untyped language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what value the variable will hold.

In other languages like Java/C++, you need to tell the data type of a variable before assigning a value to it

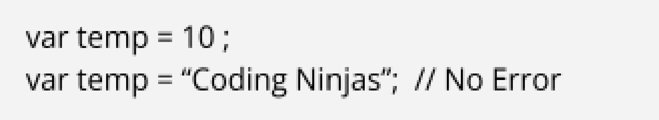


```
Example : var temp = 10 ;
               temp = "Coding Ninjas";
Here the temp variable can store both String and Number type values
```

The value type of a variable can change during the execution of a program, and JavaScript takes care of it automatically.

Note: Use the var keyword only for declaration or initialization., once for the life of any variable name in a document. You should not re-declare the same variable.

What if you’ve written the above example like

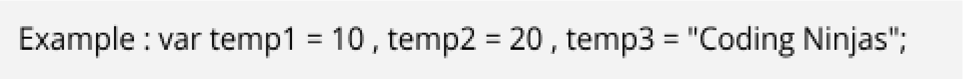


```
var temp = 10 ;
var temp = “Coding Ninjas”;  // No Error
```

This is also accepted in JavaScript, but you should not use var again for the same variable.

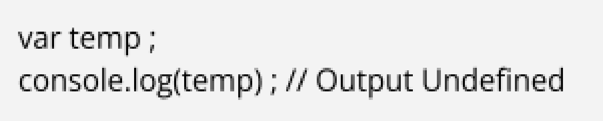
Declaring multiple variables at the same time

Separate the variable names with a comma(,)



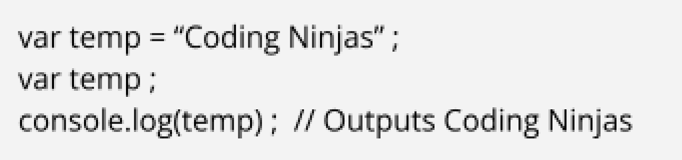
```
Example : var temp1 = 10 , temp2 = 20 , temp3 = "Coding Ninjas";
```

Re-Declaration of a variable



```
var temp ;
console.log(temp) ; // Output Undefined
```

But,

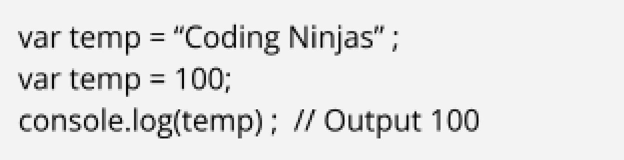


```
var temp = “Coding Ninjas” ;
var temp ;
console.log(temp) ;  // Outputs Coding Ninjas
```

Here re-declaring the variable does not make any difference to the value.

Re-Definition of a variable

Here variable takes the most recent assigned value to it.



```
var temp = “Coding Ninjas” ;
var temp = 100;
console.log(temp) ;  // Output 100
```