SCOPE COURSE

APRIL 2022

YONATAN BENEZRA

- DO NOT DISTRIBUTE –

- DO NOT COPY – ALL RIGHTS RESERVED TO THE AUTHOR –

Scope determines the accessibility (visibility) of variables.

There are three types of scopes

- Block scope
- Function scope
- Global scope

Block Scope

Variables declared inside a { } block cannot be accessed from outside the block:

```
{
  let x = 2;
}
console.log(x)
```

what will the console log?

[redacted]

Local Scope

Variables declared within a JavaScript function become LOCAL to the function.

```
// code here can NOT use carName

function myFunction() {
  let carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
```

Local variables have Function Scope:
They can only be accessed from within the function.

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts and is deleted when the function is completed.

Function Scope

JavaScript has function scope: Each function creates a new scope.
Variables defined inside a function are not accessible (visible) from outside the function.
Variables declared with **var, let** and **const** are quite similar when declared inside a function.

They all have Function Scope:

```
function myFunction() {
  var carName = "Volvo";   // Function Scope
}
function myFunction() {
  let carName = "Volvo";   // Function Scope
}
function myFunction() {
  const carName = "Volvo";   // Function Scope
}
```

Global JavaScript Variables

A variable declared outside a function, becomes GLOBAL.

```
let carName = "Volvo";
// code here can use carName

function myFunction() {
// code here can also use carName
}
```

A global variable has Global Scope:
All scripts and functions on a web page can access it.

Global Scope

Variables declared Globally (outside any function) have Global Scope.
Global variables can be accessed from anywhere in a JavaScript program.
Variables declared with **var, let** and **const** are quite similar when declared outside a block.

They all have Global Scope:

```
var x = 2;       // Global scope
```

```
let x = 2;        // Global scope
const x = 2;        // Global scope
```

JavaScript Variables

In JavaScript, objects and functions are also variables.

Scope determines the accessibility of variables, objects, and functions from different parts of the code.

Automatically Global

If you assign a value to a variable that has not been declared, it will automatically become a GLOBAL variable.

This code example will declare a global variable **carName**, even if the value is assigned inside a function.

Example:

```
myFunction();

// code here can use carName

function myFunction() {
  carName = "Volvo";
}
```

JS Lifetime

The lifetime of a JavaScript variable starts when it is declared.
Function (local) variables are deleted when the function is completed.
In a web browser, global variables are deleted when you close the browser window (or tab).

***Note:**
It is best to try to avoid using global scopes.
The more specific we are, the better our code will be.