**JS Scope:**

Scope determines the accessibility (visibility) of these variables. One advantage is that scope provide some level of security to your code. In JavaScript there are two types of scope:

1)Local scope 2) Global scope

**Local Scope**: Variables defined inside a function are in the local scope. And they have a different scope for every call of that function. This means that variables having the same name can be used in different functions. This is because those variables are bound to their respective functions, each having different scopes, and are not accessible in other functions.

**Global scope:**  A variable is in the Global scope if it's defined outside of a function. All scripts and functions on a web page can access it.

**e.g.**

var foo = "bar"; //global variable

function one() {

var foo1 = "bar1"; //local variable

function two() {

console.log(foo);

console.log(foo1); //no error as function two is inside one and is in the scope od one

}

}

function three() {

console.log(foo); // no error as it can access global variable.

console.log(foo1); //error: because it cannot access the local variable defined inside function one.

}

**Global Object:** Global scope in javascript behaves like an object. This object usually has a name, which varies depends on the context in which javascript code is running. In browser it is usually called window.

Defining a variable on the global scope causes it to be available as a property of the global object:

var foo = "bar";

function baz() {

console.log(foo); // “bar”

console.log(this.foo); // “bar”

}

baz();

By contrast, defining a variable inside a function, does not mean that that variable will be available on that function's object:

function baz() {

var foo = "bar";

console.log(foo); // “bar”

console.log(this.foo); // undefined

}

baz();