Mohan Sai Potla

22 Oct 2022

JavaScript Assignment 9

1. Carefully observe this example.
a) Is the InnerFunction() a closure?
b) What is output of this program

```
function OuterFunction()
{ var outerVariable = 100;
function InnerFunction() {
alert(outerVariable);
}
return InnerFunction;
}
var innerFunc = OuterFunction();
innerFunc();
```

a) InnerFunction() is a closure. It remembers the value of outerVariable i.e., 100 which is in its lexical scope.

b) The above program alerts the user with value 100.

2. What is the difference between a closure and a scope ?

A function along with its lexical scope forms a Closure. For example, if we create a function inside another function, the inner function is said to be closure.

Scope is nothing but what variables you have access to. It refers to the current context of the code, which determines the accessibility of variables. There are two types of Scopes: 1. Global Scope and 2. Local Scope.

3. What is a lexical scope and how is it related to closure?

Lexical Scope refers to the location where the variable is declared within the source code to determine where the variable is available.

A function along with its lexical scope forms a Closure. It means the closure can remember the variables from the place where it is defined, no matter where it is executed. Closure carries the values in its lexical scope along with it.

4. Output of following closure ?

```
for (var i = 0; i < 3; i++) {
setTimeout(function log() {
console.log(i); // What is logged?
}, 1000);
}
```

The above code will log the value 3 three times after 1 second. This happens because of closure. The closure remembers its lexical environment and remembers the reference of variable ‘i’. By the time of callback function executes, the for loop completes its iteration upto 3. Therefore, it logs the value which is stored in the variable i.

This problem can be avoided by using let keyword instead of var keyword. Because the let keyword is block scoped. So, every time the setTimeout is called the callback function forms a closure with new copy of variable.