## Carefully observe this example.

### a) Is the InnerFunction() a closure?

The command **return InnerFunction;** returns InnerFunction from OuterFunction when you call OuterFunction(). The variable **innerFunc** references the I**nnerFunction()** only, not the OuterFunction(). So now, when you call innerFunc(), it can still access outerVariable which is declared in OuterFunction(). This is called Closure.

### b) What is output of this program ?

The output of this program is **100**.

function OuterFunction()

{ var outerVariable = 100;

function InnerFunction() {

alert(outerVariable);

}

return InnerFunction;

}

var innerFunc = OuterFunction();

innerFunc();

## What is the difference between a closure and a scope ?

When you declare a variable in a function, you can only access it in the function. These variables are said to be **scoped** to the function.

If you define any inner function within another function, this inner function is called a closure. It retains access to the variables created in the outer function.

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

The lexical scope allows a function scope to access statically the variables from the outer scopes. Finally, a closure is a function that captures variables from its lexical scope. In simple words, the closure remembers the variables from the place where it is defined, no matter where it is executed.

## Output of following closure ?

for (var i = 0; i < 3; i++) {

setTimeout(function log() {

console.log(i); // What is logged?

}, 1000);

}

## 