Here lets discuss a good eg of closure to tell in interview .

Q. wt is a closure in js.

A function along with reference to outer environment together forms a closure, in other words, we can say that , closure is a combination of a function and its lexical scope bundled together.

Q. can u explain it more about it.

Its like each and every function in js has access to its outer lexical env , I.e., access to its outer lexical environment .like access to the variables and functions which is present in the environment of its parent.so each and every function has access to them.

So even when this functions is ececuted in other scope , not in its present scope, if its executed in some other scoep, it still remembers its outer lexical env where it was originally present in code.

Q. can u give an eg to demonstrate wt u said.

// suppose if we have nested fucntions - let say funcstion outer and function inner inside it.

// function outer(){

//     function inner(){

//     }

// }

// so actually in JSON, this inner function will have access to its outer env. so lets create a variable in its outer environment. var a=10;

//so inner function will have access to this variable a.

// so we can do console.log(a); this inner function, along with its outer lexical environment is known as closure.

// function outer() {

//   var a = 10;

//   function inner() {

//     console.log(a);

//   }

// }

// so the major part of closure is, suppose if i return  this inner function to outside,so by using this function trying to access 'a' from somewhere else in outer scope also ,it still remembers the value of 'a' was.

//

function outer() {

    var a = 10;

    function inner() {

      console.log(a);

    }

    return inner;

  }

  outer()();  // it stil prints 10.- executed some wehre else, not in lexical scope. so it remembers  a.

Q. wt u done in the last line.? Outer()()? Wt does it mean

This parenthesis is like calling the inner function.

If I do outer() – it returns inner function.

Outer()()- this call will call the inner function in same line. Shortcut

We can even do it like this

Var close=outer();

Close();

Both are same, it gives same answer

**Q. suppose wt if I move line no var a=10; to the line before return stmt, then also it will be a closure right?**

function outer() {

  function inner() {

    console.log(a);

  }

  var a = 10;

  return inner;

}

outer()();

//it will still form a closure and it will work still the same way.actually the inner fucntion forms  a closure with its outside env not in a particular sequence that wer it present. it still forms a closure.

**Q. wt if I change var a to let a. wt difference it make, as let is a block scope.**

function outer() {

  function inner() {

    console.log(a);

  }

  let a = 10;

  return inner;

}

outer()();

// as let has a block scope here, we cant access 'a' outside , but still it behaves same way. the inner funciton will still forms a closure and  it works sae way.

**Q. suppose in line no 1 – in outer function if we have a extra parmeter then wt happens.**

**Lets assume that this outer func takes a parameter ‘b’ . then how it works in.**

function outer(b) {

    function inner() {

      console.log(a,b);

    }

    let a = 10;

    return inner;

  }

var close=outer("hellowrold") ;

close();

//in this case,it behaves the same way,suppose if i access b also  as console.log(a,b)- it prints a and b.

// it works proper bcoz, inner function will forms a closure with outer env and b is also a part of outer env for inner function. so b is alsos trated in same way and inner forms a closure with b also.  so it should now print 10 and helloworld

**Q. wt if ur outer function is nested inside another function.? Wt happen will inner will have access to env of that func also .**

grandouter() {

    let x = 30;

    function outer(b) {

      function inner() {

        console.log(a, b, x);

      }

      let a = 10;

      return inner;

    }

    return outer;

}

    var close = grandouter()("hellowrold");

    close();

  //    var close = grandouter()("hellowrold"); - thiss line is  calling  grandouter() first- it returns the outer function and  then this returned outer function will get called with parameter 'helloworld'

**Q. wt if we have a global variable with conflicting name.**

**Let say the line let a=10; is again repeated with same variable ‘a’ globally that is at like before var close=…. Line.**

So inner func has access to inner a. has a ref of inner a only.

And the outer a will be completely new var in global scope.so, these two variables will be completely different variables,it will still from a closure with inner a only and still print 10 only.

And it doesn’t care outer a-100.

And when this copy of inner func goes to close and it is run here , it remembers the valueof which points to inner a variable but not outer a variable.

So it still print same 10.

grandouter();

{

  let x = 30;

  function outer(b) {

    function inner() {

      console.log(a, b, x);

    }

    let a = 10;

    return inner;

  }

  return outer;

}

let a = 100; //conflicting global variable.

var close = grandouter()("hellowrold");

close();