episode 10

Closures in javascript

```
- function x(){
    var a = 7;
    function y(){
        console.log(a);
    }
    y();
  }
  x();
```

$\Longrightarrow$ 7

[in y(), closure is formed with the variable which were part of x() lexical scope.]

- closure basically means that, a function bind together with its lexical environment.

- A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (lexical environment).
- In other words, a closure gives you access to another function's scope from an inner function.
- In javascript, closures are created every time a function is created, at function creation time.

Page No.
Date

[once y is returned, x is now gone, vanished (x is no longer in call stack now)]

```
- function x(){
      var a=7;
      function y(){
          console.log(a);
      }
      return y;               ⟹ f y(){
  }                                  console.log(a);
  var z=x();                       }
  console.log(z);
```

```
- function x(){
      var a=7;
      function y(){
          console.log(a);
      }                       ⟹ f y(){
      return y;                      console.log(a);
  }                                }
  var z=x();                       7
  console.log(z);
  z();
```

[this prints 7 in console, and here closure comes into picture. Functions are so beautiful that when they are returned from another function, they still maitains their lexical scope, they remember that where they were actually present. Though x() function is completely vanished but still y() function remembers its lexical scope where it came from i.e. it remembers that there was something a]

so here function with its lexical scope i.e. closure was returned.

Page No.
Date

-
```
function x(){
    var a=7;
    function y(){
        console.log(a);
    }
    a=100;
    return y;
}
var z= x();
console.log(z);
z();
```

$\Rightarrow$
```
f y(){
    console.log(a);
}
100
```

- Uses of closures:-
  1. Module Design pattern
  2. Currying
  3. functions like once
  4. memoize
  5. maintaining state in async world
  6. setTimeouts
  7. Iterators

[in javascript, you can pass function as argument to another function.
you can also return a function out of another function.
when you return a function, closure is returned i.e. function with its lexical scope is returned]