Difference between == and === operator in JavaScript

==	===
The '==' operator tests for abstract equality	the '===' operator tests for strict equality i.e it checks data types are equal or not
Does Type conversion	Does not do type conversion
== make type correction based upon values of variables.	=== takes type of variable in consideration.

1. Slide 52

```
var add = (function() {
   var counter = 0;
   return function() {
      counter += 1;
      console.log(counter);
      return counter;
   }
})();

add();
add();
add();
add();
```

If we print add we get

```
ƒ () {
      counter += 1;
      console.log(counter);
      return counter;
   }
```

This is modular pattern in JS:

→functions which invoke themselves recursive functions. That is the reason Ben Alman gave self-invoking functions a new name: Immediately Invoked Function Expression (IIFE).

→In JavaScript, variables outside the scope of a top-level function are global (meaning, everyone can access them). Because of this, it's common to have "namespace pollution", where completely unrelated code shares global variables.

→It's all about variable scoping. Variables declared in the self executing function are, by default, only available to code within the self executing function. This allows code to be written without concern of how variables are named in other blocks of JavaScript code.

→So basically we use this modular approach because we want only our function to know about the variable and not giving global scope to variable counter so only add which is a function only knows about counter and increments the values to it every time it is invoked.

So first add sets counter = 0

add() →called first times increments counter =1

add() →called second time increments counter=2

add()→called third time increments counter=3

add()➔called fourth time increments counter=4