accordingly results changes.

* this keyword

```
const a = {
    eng : 12,
  math : 45,
  getavg(){
      let a = this.eng      // to access value of
                              eng inside fun
                              we use this
```

- **try and catch :-**

→ Try allows to define a block of code to be tested for errors while executed.

→ Catch allows you to define a block of code to be executed ~~if any error occur in try block.~~

```
try {
    console.log(a);
}
catch {
    console.log("xyz"

}

// or
catch (e) {
        console.log("~~not exist~~ error")
        console.log(e) // for knowing what is error)
```

- **Arrow function :-** // Nameless

```
const func = (arg1, arg2) => {
        console.log(arg1 + arg2);
}
func(2, 4)
func  //  (arg1, ar2) => {
                 console.log(a+b);
          }
```

→ If one argument only then without () also funⁿ can work.

⇒ When no argument there () must

- **Implicit** Return (Arrow funⁿ)
  ↓ automatic

only change :-

```
const func = ( arg1 ,arg2) => { value(;) };
```

use parenthesis in place of curley.
not use this

- **Set Time out** :- callback(कोई भी aas a funⁿ जो दूसरे funⁿ it as a argument jata)

```
→ setTimeout (function , timeout)   (millisecond)
→ console.log ("hi there");
  setTimeout ( () => {
          console.log ("Apna clg");
  } , 4000);   4sec
console.log( "welcome to");
// Agr iske bad ca[illegible] bta hote then pehle ye
   print 2 unke   print hone ke 4sec bad   Apna clg
```

output :-

```
hi there
welcome to
// After 4sec
Apna clg.
```

* **Set Interval** :- (Value ki interval k bad multiple times chalega)

(ms)

```
→ set Interval (funⁿ, timeout)
→ set  ''  ( () => {
                console.log("Apna")
    }, 2000);
~~console~~. log(id
→   let id = se
    console.log(id)    // तो uski id देगा
    clearInterval(id)  // Print hona bnd.
```

An if 3 बार print करवाना है तो इसे use and after loses we get timeout & inside it clearInterval(id)

Arrow	function
~~Khud ka scope~~	Khud ka scope
inherit Parent ka scope.	

```
const  student = {
           name  : "aman",
           marks : 95 ,
           prop  : this        //Global scope.
    //when we print student it will be like(prop : window)
           getName : function(){
                  console.log(this);   // यहा this
                  return this.name;
           }.
           getmarks : () => {
                  console.log(this);   //window object.
                  return this.marks ;  //undefined as window
                                         object के लिए marks
                                         undefined
           }
}
student. get Name();
student. get Marks();
```

Object के लिए इसका this na global scope तो window object.

funⁿ के लिए this calling vali object

eg: //arrow funⁿ के लिए this parent scope

arrow funⁿ → obj. ke funⁿ → window | Hence lexical scope

इसका भी this

Where arrow is useful :-

Upper vala as it is :-

In the case of arrow if we use () in place of {} then no return will be there i.e. () & to be used only for one command only.
• {} → return a+b; same work
() → a+b

for more ref. 2:55:8
2:55

```
get Info1: function(){
    setTimeout(() => {
        console.log(this);   //student.
    }, 2000);
```

As arrow object तो
जो parent का vhi iska
& function() का this
student hai as from
above

यहाँ parent है
function() है

यहाँ भी call window object ne lagayi
but due to arrow funⁿ ye aphi call
this deteken. as setTimeout को window
object call krta.

```
get Info2 : function(){
        setTimeout( function(){
            console.log(this);   //window
        }, 2000);
}
};
```

यहाँ arrow nhi
to ye देखेंगे
किसने call
lagayi h to
hoga
window object

⇒ ऊपर भी setTimeout, set interval or other built in
funⁿ arrow funⁿ useful.