

Start the time
and move
ahead

Closures

```
1 function todo(task) {  
2   setTimeout(function fun() {  
3     console.log("Completed", task);  
4   }, 2000);  
5 }  
6  
7 console.log("Starting");  
8 todo("assignments"); → calling todo  
9 console.log("Ending");
```

→ At the time instance
when function fun will
be called, the todo
function is already over

→ Starting
Ending

if fun is called after the completion
of todo, & todo has ended, how
fun is able to access task variable

This happens due to closures.

Closure is when a function "remembers" its lexical scope even when the function is executed outside that lexical scope.

if todo is completed it will be removed from
call stack. how actually the variables still
persist ?? where are they ??

```

function fun(task1 , taks2){
  task1 = "paras"
  setTimeout(function gun(){
    console.log("completed" , task1);
  }, 2000)
  task1 = task2 ;
  task2 = "assign";
}

fun("12", "34");

```

Handwritten annotations on the code:

- A red arrow points from the word `fun` in the function signature to the word `fun` written above it.
- A red arrow points from the word `gun` in the `setTimeout` callback to the word `fun` written above it.
- A red bracket is drawn around the `task1` parameter in the function signature.
- A red bracket is drawn around the `task1` variable in the `console.log` statement.
- A red bracket is drawn around the `task1` and `task2` assignments in the function body.

task1 = "34"

task2 = "assign"

```

1 function test() {
2   for(var i = 0; i < 3; i++) {
3     → setTimeout(function exec() {
4       console.log(`i : ${i}`); // 'i: ' + i
5     }, i*1000);
6   }
7 }
8
9 test();

```

funcⁿ scope → test

i = 0 x 3

Timer → 0ms

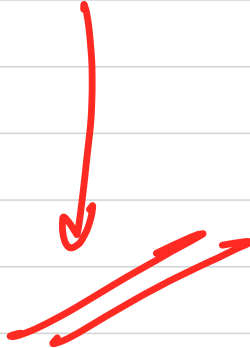
Timer → 1s

Timer → 2s

i : 3

i : 3

i : 3



block scope

```
1 function test() {  
2   for(let i = 0; i < 3; i++) {  
3     setTimeout(function exec() {  
4       console.log(`i : ${i}`); // 'i: ' + i  
5     }, i*1000);  
6   }  
7 }  
8  
9 test();
```

i = 0

Time → 0ms

i = 1

Time → 1s

i = 2

Time → 2s

```
1 function todo(task) {  
2   ↳ setTimeout(function fun() {  
3     console.log("Completed", task);  
4   }, 2000);  
5 }  
6  
7 console.log("Starting");  
8 todo("assignments");  
9 console.log("Ending");
```

Completed —

task

