· github : divyaranjan10

DATE

# CHAP 14 · Callback functions in JS ft. Event Listeners

" When you pass a function inside another function as an argument, it is called a callback function"

" It is called a callback because it is used sometimes else / later in your code."

(14.1)

$\xrightarrow{\quad}$ Callback $f^n$

```
setTimeout (function (){
    console. log ("timer");
}, 2000)
```

· Output

```
x
y
timer
```

```
function x (y) {
    console. log ("x"); y();
}
```

$\xrightarrow{\quad}$ Callback $f^n$

```
x ( function y() {
    console. log ("y");
 })
```

DATE__________

"JS is a single-threaded language and it executes one line at a time."
"JS has only one call stack and you can caller it as main thread."

(14.1) Registering of setTimeout and will store it in a separate space and attach a timer to it. Then in the setTimeout we are pass--ing the function as a callback so that we can use it sometimes later in our code.

• Blocking Main Thread in JS: As JS has only one call stack and you can call it as main thread. So, ~~teh~~ everything happens inside this call stack and if there is a heavy operation it can block our main thread. So, it's better to use some async operation like setTimeout which will execute after a certain time and won't block our code.

DATE________

• Event listener in JS:

```
document. getElementbyId ("ClickMe").
  addEventListener (" click", function xyz() {
      console. log ("Button Clicked");
  });
```

xyz() is a callback function

* So, whenever the button is clicked (event is triggered), the callback function gets pushed into the call stack.

• Closures along with Event listeners:
• Create a function that counts no. of clicks:

```
function countClicks () {
var count = 0;
   document. getElementById ("ClickMe").
     addEventListener (" click", function xyz() {
         console. log ("Button Clicked");
   });
}
countClicks ();
```

DATE ________

=> So, here the function xyz() has formed a closure with countClicks() and thus on every 'click' (event triggering), the count is increasing.

· Garbage collection & remove Event listeners :

=> Why need to remove Event listeners :

· Because event listeners are too heavy, so to free up the memory we remove Event listeners.

· So, when we remove the Event listener, all the variables which are held by this, will be garbage collected.