Night lofer (बाबा बालेश्वर सहाय की जय) [18th feb 2023.]

Dom + Modern JS - class 2

→ When we will load our code every js code will then run, we want some code to run after some events.

That's we have, (Browser Events)

(The announcement by browsers)

- → click
- → resize
- → Scroll
- → Double click
- → load event (Dom & img etc)

- events — Invisible → but to access by using method. (mointer event)
- Record to event
- Data on event
- Stop an event
- Phases/lifecycle of event

* ⊙ (Mointer Events) (write in Console.
  ↳ mointerEvents(document);
  ↓
  (Then click on document to see the Events. of website)

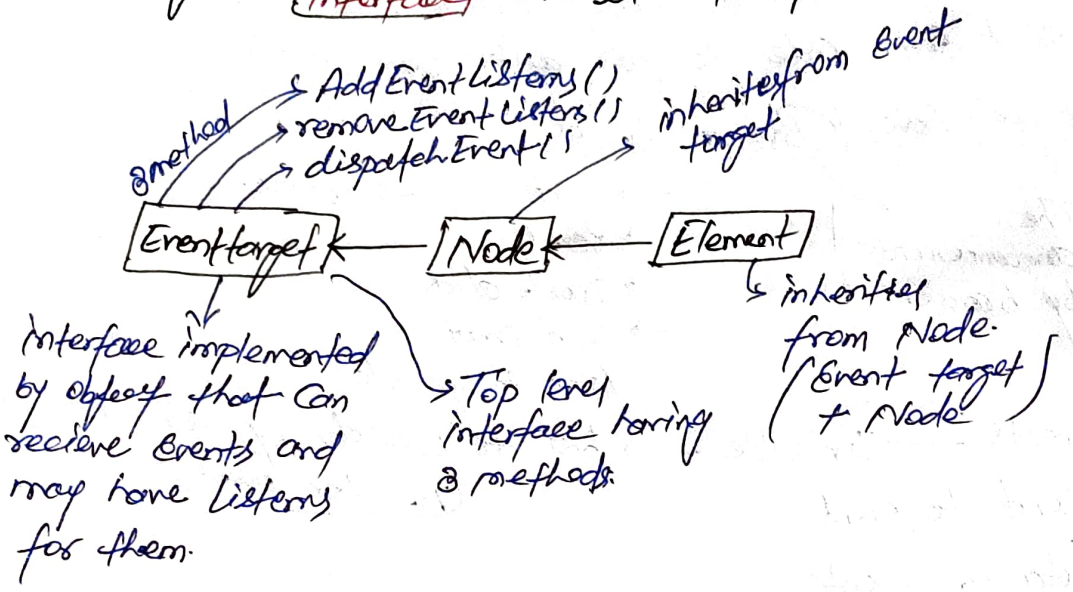
→ This method will let us see different Events, as they are occuring.

mointerEvents()
↳ turn on the Events trigger.

unmointerEvents() → Turn off

{ classes are like Blueprint and objects are Reality }

In JS → interface are like Blueprint.



* Event Listener :- Respond for Events after Received.

Node :- All methods properties of EventTarget is interfaced by Node.

Element :- Element Inherits from Node.
So, also from Event target.

Event Target → interface → Top-level-Entity

Node
Elements

3 methods
- add EventListener ()
- remove Event Listener ()
- dispatch Event ()

I.) addEvent Listener

we Can → listen to event
→ Respond to event
→ hook into Event.

Pseudocode

eventTarget.addEventListener(<event to listen for>, <function-to-run when event happens>)

(two parameters)

We Need

① Event-target
② Event type → click, double click, scroll etc.
③ function → what to do when event happens.

→ on which Component Compound
- document
- p
- div
- video etc

eg:- ① addEventListener()

```
document.addEventList('click', function() {
    Console.log('I clicked on document')
});
```

↓

Now when you will click the HTML document 'I clicked on document' will be printed in Console.

you can also add it in any particular element rather than whole document to see changes in elements.

```
let Content = document.querySelector('h1');
Content.addEventListener('click', function() {
});
```

* RemoveEventListener

== vs ===

↑ loose equality

↑ strict equality

always type coercion

↳ when js will try to convert the items being compared to some type.

The function you have passed for addEventListener you need to pass the exact function to removeEventListeners.

we can only Remove when we will create function with name specifely.

```
function point() {
        console.log('Hi');
}
document.addEventListener('click', point);
document.removeEventListener('click', point);
```

↳ This will here remove event listener because function is an object in javascript. if you will create function in addEventListener and then in remove, they both all are not same.

This is not the correct way for removing purpose.

```
('click', function () {
});
```

To make removeEventListener() work successfully

- same target
- same type
- same function

you can check any websites event listners, inspect and then go to event listner tab beside Console tab.

Phase of an Event :-

- Capturing phase
- At target phase
- Bubbling phase

↓ returning back from at target phase.

Searching of the element before event is triggered.
↳ when reached the element

Syntax of addEventListners. (3 parameters)

addEventListner(type, listner, useCapture)

- type → Event type
- listner → function ↓ should happen after event trigger.
- useCapture → phase in which event is Captured. (By default bubbling phase.)

The Event object :-

when an event occurs, addEventListener function gets (event object)

↓ lots of information about Event.

```
Const Content = document.querySelector('#wrapper');

Content.addEventListener('click', function(Event){
    {
    Console.log(Event);
```

{ ↳ you will get all event information when clicken on element with wrapper id.

The Default Action :-

↳ To prevent default Action we use preventdefault() method.
We can change the default working of any element.

- preventDefault()

Links :-

Anchor tag → Link open

```
let Links = document.querySelectorAll('a');
   Way to fetch 3rd from all
let thirdLink = Link[2];
thirdLink.addEventListener('click'
                function(event){
            event.preventdefault();
        Console.log('maza aya');
});
```

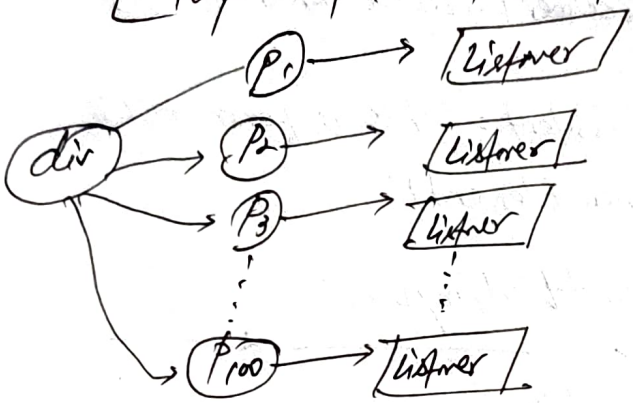
This will change of Anchor tags

How to Avoid too many Events

```
let myDiv = document.CreateElement('div');
    for(let i=1; i<=100; i++){
  let newElement = document.CreateElement('p');
newElement.textContent = 'This is para ' + i;
newElement.addEventListener('click', function(event)
```

```
Console.log('I have clicked on para');
});
myDiv.appendChild(newElement);
}
document.appendChild(myDiv);
```

[Created div, & created 'p' element in a loop and loaded event listner to p.]



This will take memory as same work and lots of listener to optimize.

more optimize:-

But in this we will not be able to individually access paragraph. individually fast. the whole div is access.

Now phase will help Here!

Event target property -

↳ The target property returns the Element where the event assoned.

Now the optimised Code:-

```
let myDiv = document.createElement('div');
  Console.log('para' + event.target.textContent);
    {
myDiv.addEventListner('click', paraStatus);
  for(let i=1; i<=100; i++){
let newElement = document.createElement('p');
  ~~myDiv.append~~
    newElement.textContent = 'This is para' + i;
myDiv.appendChild(newElement);
    {
document.body.appendChild(myDiv);
```

```
<article id="wrapper">
  <p> ABCD <span> xyz </span> </p>
</article>
```

↑ we will add event listner to this span. what will happen?

It will also work when (p) is clicked ✓ for para, span also work! ✗

Now to get rid of this we property → nodeName

```
let element. document.querySelector("#wrapper");
Element.addEventListner('click, function(event){
if(event.target.nodeName === 'SPAN'){
  Console.log 'Span clicked !' + event.target.textContent();
  });
```

Specific tag filtering.

why <script> at the bottom of <body> tag?

↓

if it will in head tag. script will ~~write~~ work before HTML document is loaded.

[How we will know HTML is loaded by Event]
→ DomContentloaded.

if you want to use in head, write domContent (loaded event inside script)

but best protics is bottom of <body> tag

श्री हनुमते नमः

[19th feb 2023]

DOM + ModernJS - class - 3

→ performance
- measure speed of code.
- how to write efficient & ~~pe~~ performaning
- Event loop.

A Standard way to measure how long your code takes to run.

↓

by using

Performance.now() — method.

This is very accurate.

```
Const time1 = performance.now()
[ This is your Code ]
Count time2 = performance.now()
Console.log( timer2 - time1);
```

when we add paragraph in DOM.

2 things happeness
- Reflow (Calculation for element dimention and possitioning etc)
- Repaint (to show element pixel by pixel on your screen)

good practices is — less confined Repaint repetion in your DOC.

{ Reflow takes more time
Repaint take time but less than Reflow }

Best practice → Document fragment

Lightweight document object, no reflow & repaint when we add element to it, then we will add Document fragment to document Then it will do one Reflow & Repaint.

→ The Call Stack :-

Single-threading :- one Command at a time JS is Single Headed Language.

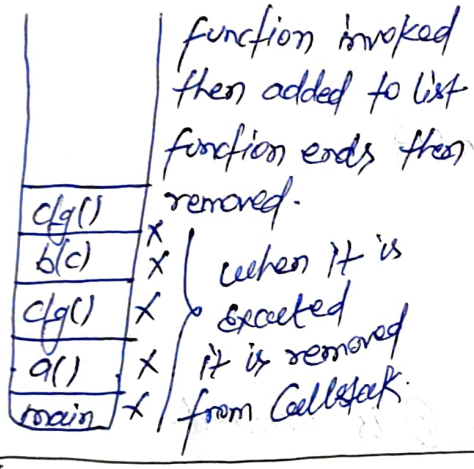
↳ Parsing processing of one Command at a time.

Single-Threaded

Executes line by line.
Ignores function but when function is called goes inside function then again line by line.

→ Run to Compilation nature of Language.
→ JS does not execute multiple line or multiple function at a time.

Call Stack is a list that tracks or store the function :-



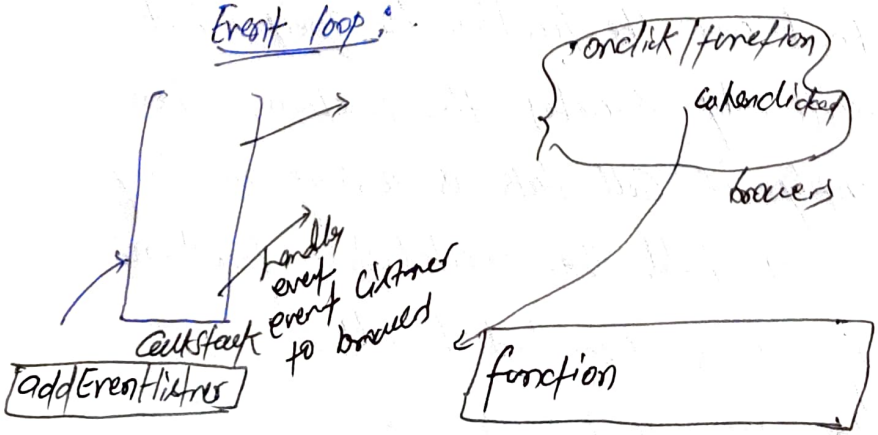

```
function a() {
  Console.log ("Hi")
    b(); }
function b() {
Console.log ('Hello');
  }
  a();
```

Important Event loop :-

Synchronous :- occuring at a same time.

Event-Listener is → Asynchronous because it works when action is performed.

Event loop :



Event query
(This will execute function only when call stack is empty)

Code:-

```
① clg ('Hi');
② element.addEventListner
        ('click' function () {
         clg('123');
③ clg ('Hello');                 }
```

only will run when clicked else ③ will be executed.

addEventListner / loop EXPLAINED

Code:-

```
1) clg ('ABCD')
② element.addEventListener ('click', function () {
        clg('1234');                                 }
③ clg('xyz');
```

Now, in call stack

→ Entry of ① and 'ABCD' is printed & ① is executed.

→ Then entry of ② event listner, but it is when clicked then function, so call stack hands over event listner to brower and move to ③ - clg('xyz')

Now, when clicked

Browers will hand over the function to query, but the query will only execute the function, when Callstack is empty. if call stak is working on any function. query will hold the event listner function when its empty, it is executed finally.

This loop is called Event loop.

A Bit more? -

① Asynch code → depends on JS Event loop.

② Any Asynch Code is handled by browsers.

Callstack → Browser → queue

↓ waits on till callstack is empty.

SetTimeout()

```
SetTimeout(function() {
    alg('Hi');
}, 4000);
```

↑

waits for 1000ms or 4sec before execute but No guarantee 4sec is minimum time can take more, waits for call stack to be empty.

↓

because this is also Asyne Code.

SetTimeout 2 parameters

(function(), Time)

when you want to defer sometimes, your can use SetTimeout.

SetTimeout, 0

↓

does not mean to run immediately.

it will still do not Event loop.