Tuwash Notes

18th Feb 2023

DOM + Modern JS - class 2

→ when we will load our code every js code will run, But we want some code to run after some Events.

That is we have, **Browser Events**

(The announcement by Browser.)

→ click → resize
→ Scroll (etc)
→ Double click
→ Load event (DOM loading etc)

- events — (Invisible) → but to watch by using method (moniterEvents)
- resond to event
- Data Stored in event
- Stop an event
- phases / lifecycle of event.

* MoniterEvents (write in console)

```
moniterEvents(document);
```

(Then click on document to see the Events of website) ✓

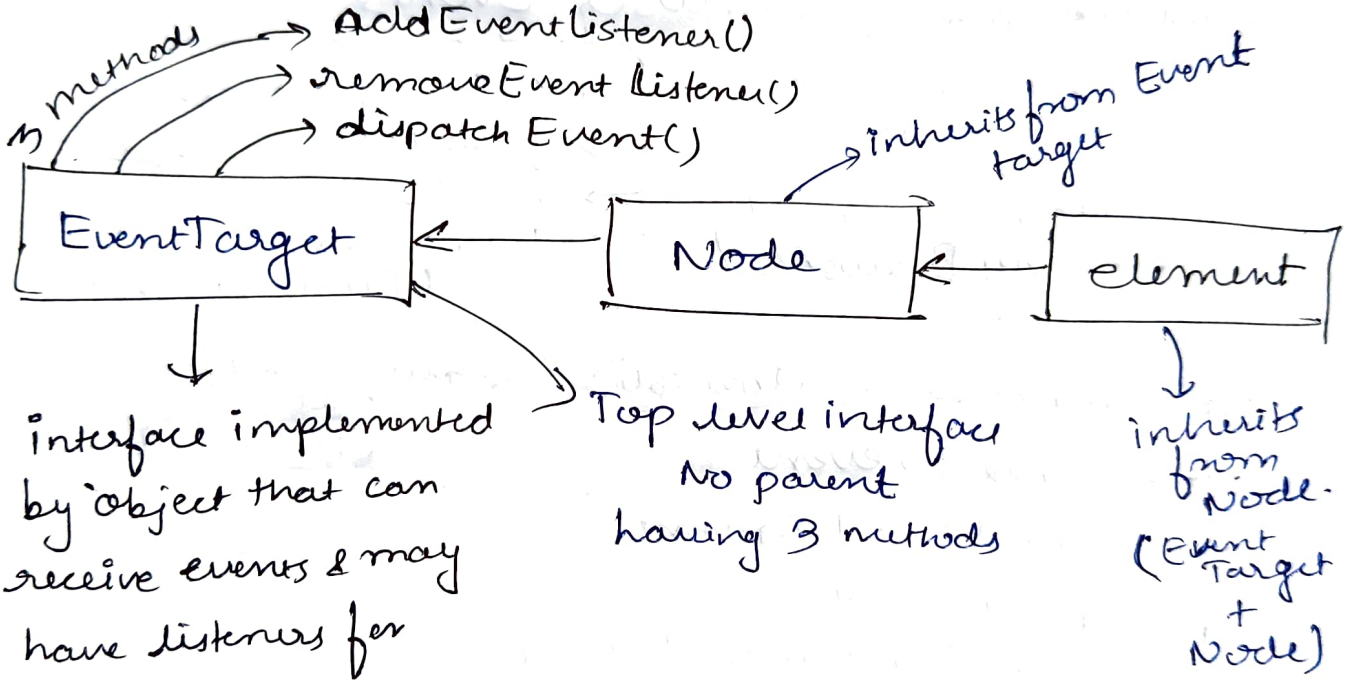
→ This method will let us see diffurent Events, as they are occuring.

monitorEvents()
↳ turn on the events trigger

unmonitorEvents() → Turn off.

{ classes are like Blueprint
& objects are Reality }

in js → interface are like Blueprint



* Event listener :- Respond for Events after Received

Node :- All methods & / properties of EventTarget is inherited by Node

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

Event Target → interface → Top-level-Entity.

Node
elements.

3 methods
- addEventListener()
- removeEventListener()
- dispatchEvent()

1) addEventListener()

we can → listen to event
↳ Respond to event
↳ hook into event

Pseudocode

<eventtarget>.addEventListener(<event to listen for>, <function-to-run when event happened>)

(Two parameters)

We Need
1. Event-target
2. event type → click, double click, Scroll. etc.
3. function → what to do when event happened

on which component
- document
- p
- div
- video
- etc.

eg:- (1) addEventListener

```
document.addEventListener('click', function() {
    console.log("I clicked on Document")
});
```

↓

Now when you will click the HTML Document 'I clicken on Document' will be printed in console

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

```
let content = document.querySelector('h1');
content.addEventListener('click', function(){
    content.style.background = 'red';
});
```

RemoveEventListener

= = v/s = = =

↑ loose Equality

↑ Strict equality

allows Type coercion

↳ when JS will try to convert the items ~~big~~ being compared to same type

✓ The Function you have passed for addEventListener you need to pass the Exact Function to removeEventListener.

We can only Remove when we will create function with name Separately.

```
function print() {
    console.log('Hi');
}

document.addEventListener('click', print);
document.removeEventListener('click', print);
```

This will have remove event listener will work because function is an object in javascript. if you will create function in addEventListener & then in remove, they both 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 Chuck any website's event listeners, inspect & then go to ~~Ent~~ event Listeners ~~tab~~ beside Console tab

Phases of an Event :-

- Capturing phase
- At target phase
- Bubbling phase

Searching of the element where event is triggered

When reached the element

returning back from at target phase

- Syntax of addEventListener. (3 parameters)

addEventListener(type, listener, useCapture)

- type → event type
- listener → function → what should happen after event triggers
- 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);

})
```

- function(event) → you can write any other name also
- you will get all event information when clicken on element with wrapper id.

The Default Action :-

↳ To prevent default Action we use ~~prevent~~ preventdefault() method.

We can change the default working of any element

- preventDefault()

like :-

anchor tag ⟶ link open

```
let links = document.querySelectorAll('a');
```

Now to fetch 3rd from all

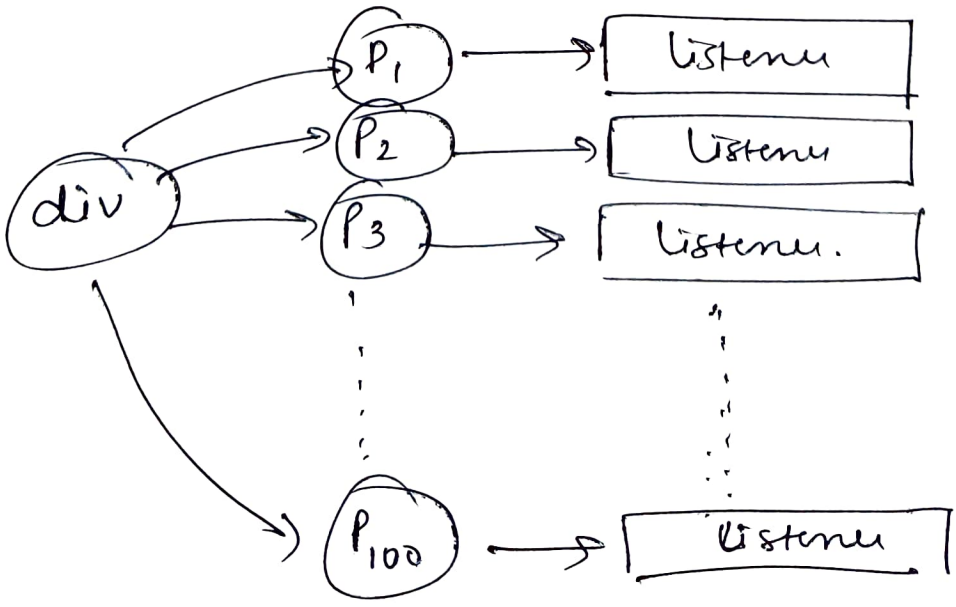
```
let thirdlink = link[2];
thirdlink.addEventlistener('click',
    function(event){
    event.preventDefault();
    console.log('maza aya');
});
```

this will change the default Action of Anchor tag.

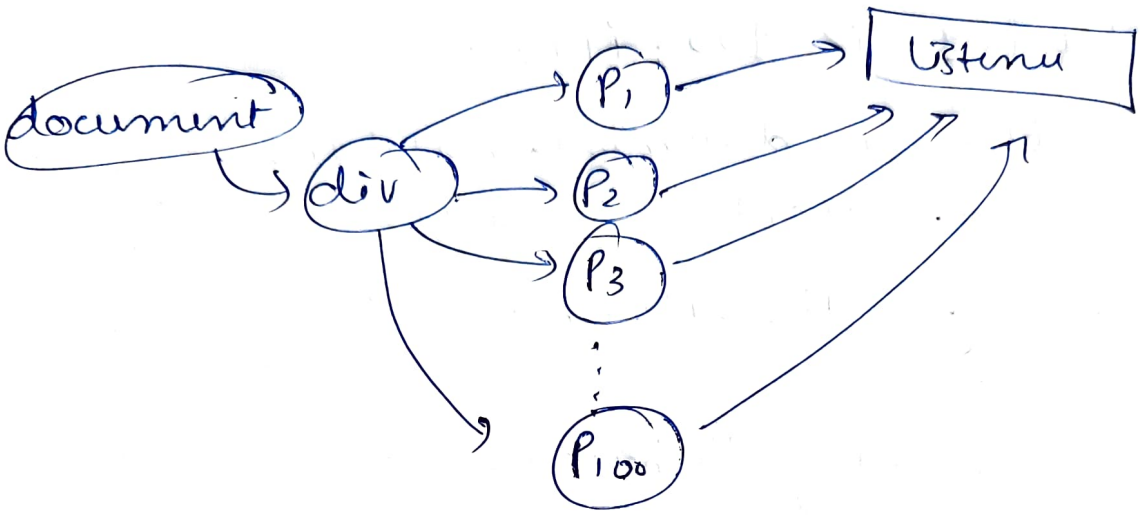
How to Avoid Too many Events ?

```
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);
```

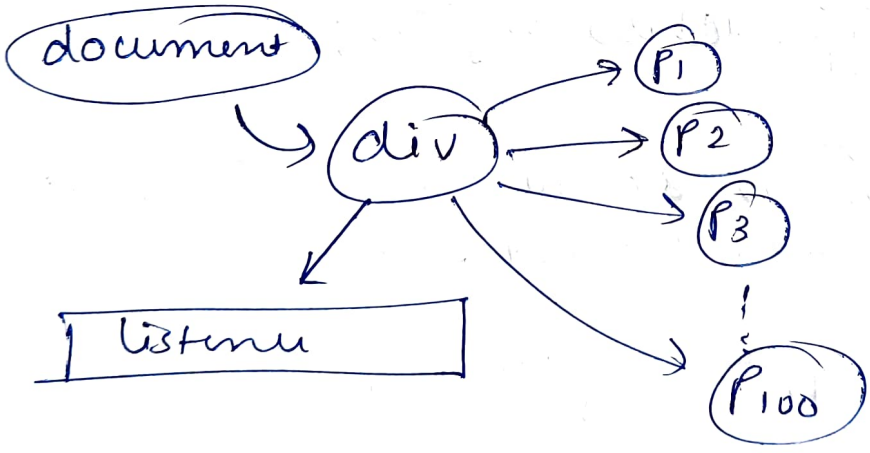
(Created div, & created 'p' element in a loop & added event listener to p.)



This will take memory as
Same work & lots of listeners
to optimize.



More optimize :-



But in this we will not be able to individually access paragraph individuality test. The whole div is access.

NOW PHASES WILL HELP HERE!

Event target property

↳ The target property returns the element where the event occurred.

Now the Optimised code :-

```
let myDiv = document.createElement('div');
Function paraStatus(event) {
  console.log('para' + event.target.textContent);
}
myDiv.addEventListener('click', paraStatus);
For (let i=1; i<=100; i++) {
let newElement = document.CreateElement('p');
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 listener to this span. what will happen?

it will also work when (p) is clicked ✓ For para, span also works (X)

Now to get rid of this use property → `nodeName`

```
let element = document.querySelector('#wrapper');
element.addEventListener('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 be in head tag.
Script will work before HTML document is loaded

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

If you want to use in head, write domContentLoaded event inside Script.

but Best practice is ~~below~~ bottom of `<body>` tag.
