CLASSTEAM Page No.
Date / /

DOM + Modern JS - 2

What will we Study?

Browser Events

- → Events
- → Respond to Events
- → Data Stored in Event
- → Stop an Event
- → Life Cycle of Event.

* Browser Events are the basically an annoucement which is done by the browser.

examples :- Onclick, mouse movement etc.
DOMContentloaded ~~etc~~, mousetouch, pointer, Eventer.

* **monitorEvents ()** :-

This is the method which shows the invisible events happen on the screen.

Syntax :- monitorEvents (document) ;

Output :-

PointerEvent	Scroll
Touch Event	click

* **unmonitorEvents();**

This methods stops the events happen on the screen.

What is an event?

→ Events are things that happen in the system you are programming — the system produces a signal of some kind when an event occurs.

For example :-

* The user clicks, selects or hover the cursor over a certain element.
* The user chooses a key on the keyboard.
* The user resizes or closes the browser window.
* A form is submitted.
* A video is played, paused or ends.
* An error occurs.

Event Listener :-

An event Listener is a procedure in JavaScript that waits for an event to occur.

* **Event Target :-**

Event target is an interface and top-level entity implemented by Object that can receive events and may have listeners for them.

Event Target ← Node ← Element

Node: inherited prop. from Event target.

Element: Inherited properties from both Node and Event Target.

Event Target has 3 methods :-

a.) addEvent Listener()
b.) Remove Event Listener()
c.) dispatchEvent()

Example of Event Target :-

(i) document object
(ii) Elements / Tags
(iii) window object

} that supports events

(a) **addEventListener() :-**

This is the method of event Target interface. Sets up a function that will be called whenever the specified event is delivered to the target.

Work function is to → Listen to Event.
→ Respond to Event.
→ Hook into event.

* **Pseudocode :-**

```
eventTarget.addEventListener(eventType To listen for, Function to run when event happen);
```

We need here in pseudocode :-

(i) **eventTarget** → At which components → document, p, div, article, video

(ii) **eventType** → click, double click, Scroll etc.

(iii) **function** → define what do do when event happen.

example :-

(1)
```
document.addEventListener('click', function(){
    console.log("I clicked on document");
});
```

Output: (After clicking on document)

I ~~have~~ clicked on document

(2)
```
let content = document.querySelector('h1');
content.addEventListener('click', function(){
    content.style.background = 'red';
});
```

CLASSTEAM | Page No. | Date

(b.) removeEventListener ():-

Firstly, We should have the knowledge of

Loose Equality	Strict Equality
→ "=="	→ "==="
→ Value Same	→ Value Same, Type Same
→ Allow Type Coercion	→ Prevent from Type Coercion
ex:- 1 == '1' → True	ex:- 1 === '1' → false.

* **Type Coercion** :- Where JS will try to convert the items, being compared to same type.

→ The removeEventListener() method of EventTarget interface removes an event listener previously registered with 'EventTarget.addEventListener()' from the target.

→ The event listener to be removed is identified using a combination of event type and function itself matching.

CLASSTEAM | Page No. | Date

Important Note :-

different References

```
document.addEventListener('click', function(){
    console.log('Hello');
});
```

DIFFERENT FUNCTION

Different References

```
document.removeEventListener('click', function(){
    console.log('Hello');
});
```

different References

They are Different Function as the function are object works on different references in the memory.

Solution of this Problem

example →

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

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

Point to same function

* removeEventListener() works only if
  - Same Target
  - Same event Type
  - Same function

example:-

```
let content = document.querySelector('H1');

function print(){
        console.log("Hello");
    }

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

Same Target

Same event Type

Same function

(c) dispatchEvent() :-

* This method allows us to automatically event happening without manually clicking on Button.
* This method allows also to create our own custom event by [new Event()] constructor.
* Then, we can dispatch the event by using [dispatchEvent()].

example:-

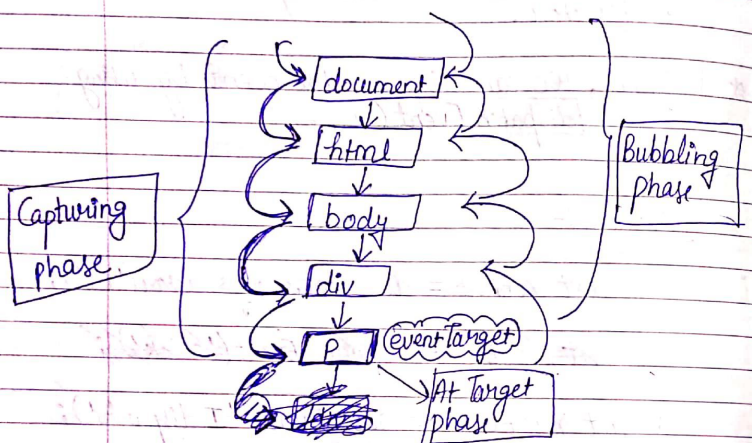
```
let content = document.querySelector('h1');
~~let newEvent = new Event('click');~~

let newEvent = new Event('I dispatch');
content.addEventListener('I dispatch', function(){
        console.log("This is the Dispatched");
    });
content.dispatchEvent(newEvent);
```

**** Note:- We need not click on H1, It automatically triggered with custom events.

Phases of an Event

Three phases :-

(i) Capturing phase
(ii) At Target phase
(iii) Bubbling phase.



NOTE :-

* By default, addEventListener is execute at → Bubbling phase

* If we want at Capturing phase, then 3rd argument of addEventListener will be true.

Syntax :- addEventListener (type, listener, useCapture)

```
eventTarget.addEventListener('onclick', function(), true);
```

true → Capturing phase

The Event object :-

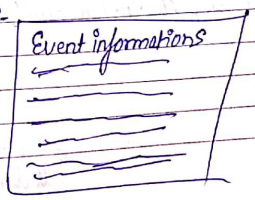
When an event occurs, an event object is passed to addEventListener() method.

Event object has lot of information about event.

ex :-
```
let content = document.querySelector('#wrapper');
content.addEventListener('click', function(babbar){
    console.log(babbar);
});
```

babbar → variable (any name)

Output :-



preventDefault() :-

The preventDefault() method Cancels the event if it is cancelable.

Meaning that the default action that belongs to the event will not occur.

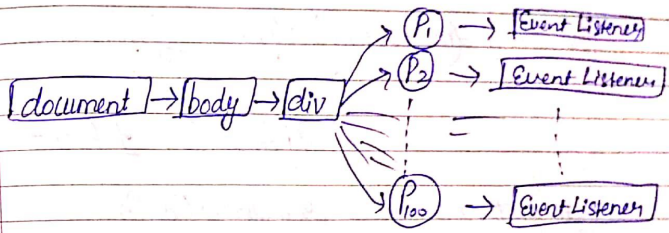
For example :- The default behaviour of ~~a link~~ anchor tag is to open a new link. But this can be prevent from opening link.

Code example :-

```
let links = document.querySelectorAll('a');
let thirdLink = links[2];
//prevent third link default behaviour here //

thirdLink.addEventListener('click', function(event){
        event.preventDefault();
        console.log('Stop Link Behaviour');
});
```

3rd Link will not Open

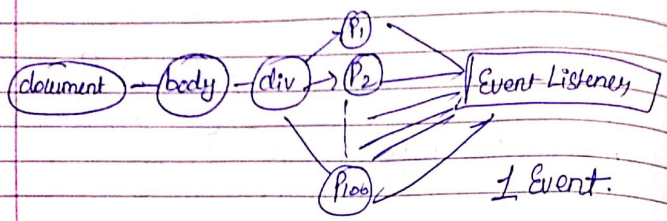
Avoid Too many Events :-




```
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.body.appendChild(myDiv);
```

This has 100 times event listener, we can avoid this.

100 Event → 1 Event




```
let myDiv = document.createElement('div');

function paraStatus(event){
        console.log('I have clicked on para');
}

for(let i=1; i<=100; i++){
        let newElement = document.createElement('p');
        newElement.textContent = 'This is para' + i;
        newElement.addEventListener('click', paraStatus);
        myDiv.appendChild(newElement);
}

document.body.appendChild(myDiv);
```

we can write here for more optimized

This is little Optimized but here 100 times of Event Called. So, More optimized.

We can optimized by using `event.target` property

```
let myDiv = document.createElement('div');

function paraStatus(event){
        console.log('para' + event.target.textContent);
}

myDiv.addEventListener('onclick', 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);
```

Output:-

This is para 1
This is para 2 ← I have clicked on para 2
⋮
This is para 100

Para Span Problem

Para Span
Para Span

```
<article>
  <p> Para <span> Span </span>
  </p>
  <p> Para <span> Span </span>
  </p>
</article>
```

If we click on Span, then only show Span clicked But not para behaviour.

```
element.addEventListener('click', function(event){
    if(event.target.nodeName === 'SPAN'){
        console.log('Span clicked' + event.target.
                    textContent);
    }
});
```

Qw:- How do we know that DOM content Loaded?

Sol⇒ Inside the head tag

```
<head>
  <script>
    document.addEventListener('DOMContentLoaded',
      function(){

      });
  </script>
```

This is Safe option but it is bad practice. ✓