JavaScript DOM + Modern JS

Browser Events

When we want to load the JS content with respect to some kind of event like click, double click, hover etc.

Example: click, scroll, resize etc. are events.

Interface: are like blueprint.
Example:

1. **eventTarget**:
    a. top level interface implements by object that can receive **events** & may have **listener (which define the action accordingly)** for them.
    b. Every element come into this like document, para, article, video etc.
    c. **addEventListener**():
        i. adding the event listener
        ii. we can listen to event or respond to event or hook into event
        iii. first we need to get <event.target>.addEventListener(<event-to-listen-for>, <function-to-run-when-event-happens>)
        iv. It also takes one more argument <useCapture> which tells in which phase we want to execute event Listener by default it works on bubbling phase.
        v. event-target: is basically a component on which that event is attached.
        vi. here event-to-listen-for: like click, scroll, hover etc.
        vii. function-to-run-when-event-happens: any function which will be called when event-to-listen-for happens
        viii.
```
// Adding Event Listener

let btn = document.querySelector('.button-new');
console.log(btn);
let cntClick = 0;
function buttonClicked() {
    console.log('You just click the button');
    if(cntClick % 2 == 0) {
        btn.style.background = 'yellow';
    }
    else {
        btn.style.background = 'red';
    }
    cntClick++;
}

btn.addEventListener('click', buttonClicked);
```
    d. **removeEventListener**()
        i. removing the event listener
        ii. it require same function which is passed in the addEventListener then only we can remove event listener.
        iii. Removing that event listener must be register with add event listener.
        iv.
```
btn.addEventListener('click', buttonClicked);


// Removing EventListener
btn.removeEventListener('click', buttonClicked);
```
        v. Same event target, same event type, same function: these all are required to work removeEvent Listener.
    e. **dispatchEvent**()

2. **Node**:
    a. Node will inherit eventTarget properties and methods.
3. **Element**:
    a. Element inherit Node properties and methods.

Events are kind of announcement.
= = allow us type coercion: where JS will try to convert the items being compared to same type.
= = = it prevents type coercion

- **Events**:
    - Events are kind of announcement and these are invisible.
    - We can see these events using **montiorEvents**() when any event is triggered.

```
> monitorEvents
< f monitorEvents() { [native code] }
> monitorEvents(document)
< undefined
```

    - **monitorEvents()** method turn on all the events
    - to turn of the events we can use **unmonitorEvents**()
- **Phases of an Event:**
    - Capturing phase:
        - It searches the element from top to down
    - At target phase:
        - When we reach to the required location it is known as target phase.
    - Bubbling phase:
        - Now when we go again to top it known as bubbling phase.
- **Event Object:**
    - When an event occurs, addEventListener have a function which has a event object which has multiple properites.

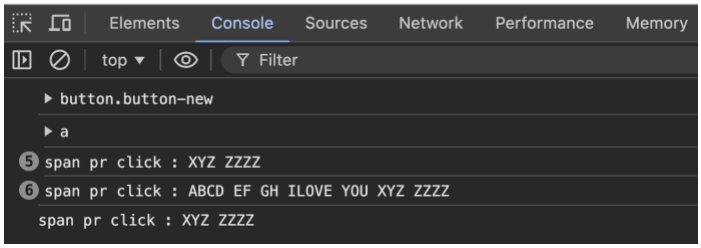
```
// Event Object
const content = document.querySelector('#wrapper');

content.addEventListener('click', function(event) {
    console.log(event);
});
// here event is an object
```

    - When we require event specific data then we can use this event object.
- **The default actions:**
    - Like an anchor tag when someone click on this a link open in new tab or in current tab. We can prevent this using preventDefault() method.

```
// Preventing Default Action
let link = document.querySelector('a');
console.log(link);

link.addEventListener('click', function(event) {
    event.preventDefault();
    console.log('Heheh');
});
```

- How to avoid too many events:
  - ```
    // Avoid too many Div
    let myDiv = document.createElement('div');
    function paraStatus(event) {
        console.log('I have clicked on para');
    }
    for(let i = 0; i<=100; i++) {
        let newEle = document.createElement('p');
        newEle.textContent = 'This is para : ' + i;
        // newEle.addEventListener('click', function(event) {
        //     console.log('I have clicked on para');
        // });
        newEle.addEventListener('click', paraStatus);
        myDiv.appendChild(newEle);
    }
    // these event Listeners are creating new object for every event
    // create the function outside
    let bdy = document.querySelector('body');
    bdy.appendChild(myDiv);
    ```
  - ```
    // Avoid too many Div
    let myDiv = document.createElement('div');
    function paraStatus(event) {
        // here we are accessing individual paragraph
        console.log('Para ' + event.target.textContent);
        // console.log('I have clicked on para');
    }
    myDiv.addEventListener('click', paraStatus);
    ```
  - Example: to access different paragraph in same event listener there we can use event.target attribute.
  - ```
    // event.target property may occur problem
    let element = document.querySelector('#wrapper-1');
    element.addEventListener('click', function(event) {
        console.log('span pr click : ' + event.target.textContent);
    });
    ```
  - ```
    <article id="wrapper-1">
        <p>ABCD EF GH ILOVE YOU <span>XYZ ZZZZ</span> </p>
        <p>ABCD EF GH ILOVE YOU <span>XYZ ZZZZ</span> </p>
        <p>ABCD EF GH ILOVE YOU <span>XYZ ZZZZ</span> </p>
        <p>ABCD EF GH ILOVE YOU <span>XYZ ZZZZ</span> </p>
    </article>
    ```
  - ABCD EF GH ILOVE YOU XYZ ZZZZ
    ABCD EF GH ILOVE YOU XYZ ZZZZ
    ABCD EF GH ILOVE YOU XYZ ZZZZ
    ABCD EF GH ILOVE YOU XYZ ZZZZ
  - 

  - Here as you can see if we are clicking on the paragraph also then it is printing the content of the paragraph also.
  - So to stop this like we do not want any kind of event listener on the p tag. We want to use event listener on the span tag there we can use **Node Name.**
  - ```
    // event.target property may occur problem
    let element = document.querySelector('#wrapper-1');
    element.addEventListener('click', function(event) {
        if(event.target.nodeName === 'SPAN') {
            console.log('span pr click : ' + event.target.textContent);
        }
    });
    ```
    ```

Why we add the script file just before the body tag why not just after the head tag.

- Because it may happen that we are accessing those thing which are not loaded yet hence we use at the last.
- We can check these using DOMContentLoaded property
- We can use script in head tag also while using the condition DOMContentLoaded but it is not best practice.