10

* DOM EVENTS

⇒ Signals that something has occurred.
(user input / actions.)

• first inline case? (agr bahar double to andar single '' use)

<button> click me </button>

<button onclick = " console.log('button was clicked'); console.log('apna clg'); " > ~~[illegible]~~ click me </button>

⇒ But inline is not that much good. as if inline ke andar JS then code Bulky.

(i) On click

```
let btn =
btn.onclick = Hello;   // or we can make fun` here also.
function Hello() {
        console.log("[illegible]")
}
```

(ii) On mouse enter. (kuch nhi kena just

```
btn. onmouseenter = function(){
        console.log ("you are [illegible] on");
}
console.dir( btn) // for showing btn
```

* Event Listeners

• add EventListener

// onclick & onmouse जैसी prop. के द्वारा की only/values // set.

>> element. addEventListener (event, callback) (event → click, drag, keyboard key)

```
btn.   "   ( "click", function() {
        console.log("button clicked");
});
```

> can make multiple also.

// btn.addEventListener("click", sayHello);
// " " " ("click", sayName);

function sayHello(){
  alert("Hello");
}
fun^n sayName(){
  alert("Name")
}

* Event listener for elements

let p = document.querySelector("p")
p.addEventListener("click", function(){
  console.log("para was clicked");
});

* this in event. listener.

let btn = ..
btn.addEventListener("click", function(){
  console.log(this); // <button> click me! </button>
console.log(this) // button vali object.
console.log(this.innerText); // click me ka innerText
this.style.backgroundColor = "blue"; // button ka bgc blue when clicked.

// Why help & help us to remove redundancy (repetition of same line) for more understanding see its code as a single fun^n is useful for 4 statements

* Keyboard Events

→ Event * let btn = document
btn.addEventListener("click", function(event){
// Pointer event console.log(event); // triggered event ki information
  console.log("button clicked") //
});
btn.add___("dblclick", function(event){
  console.log(event) // mouse event
  // // ("button clicked")

• Keyboard
let input =

// keydown: kuch bhi key press whether it is space, del etc.

input.addEvent... ("keydown", function(event){
console.log(event)
})

→ Keyup: a key ko bahut der tak hold kiya hi release: key was released → console.log("...")

Key Code.html

* event: KeyboardEvent {
// Parameter to be considered.
Code = "KeyA" // Bcoz A press
key = "a" // Bcoz a pressed.
KeyCode = 65

// If space:
key : " ", Code = "space".

> console.log(event.key)
" " (event.code)

If capital A is pressed then here is A but in code still there is KeyA. Same for KeyCode

// Application of above: Application. html

* Form Events
• In HTML form action
input
button

In JS let form = doc... .que... ;
form.addEventListener("submit", function(event){
event.preventDefault();
// ye use karte se jo submit pe click karne pe data url pe jata vo nhi jata page pe.
alert("form submitted")

// A alert can be seen more properly as when no preventdefault then console window pe kuch print hua dikh nhi pata bcoz rapidly aage chala jata hai.

• // If we use prevent command the whole val easily and properly seen as it don't disappear

* Extracting form Data :-

```
let form = document.querySelector("form");
form.addEventListener("submit", function(event){
    event.preventDefault();  // button में दबाने के बाद
                             // या link uspe jane se prevent

    let inp = docume.q - - ("input");
    console.dir(inp)
    console.dir(inp.innerText)  // Blank as jo hm console
                                   window में submit vo यहां नहीं आता
// vo value में आएगा when form submit कर देते
time jo input की value.
    console.dir(input.value)
});
```

• for accessing username & Password:

```
let form =
form.add — — ("submit", function(event){
    event.preventDefault();
    let user = document.qu — — ("#user");
    // pass = // // // ("#pass");
    alert(`hi ${user.value}, your pass is ${pass.value}`);
})
```

in place of these

```
> console.dir(form)
// & if form Object printed → elements
> form.elements
// collection
> form.elements[0] // first element.
> let user = this.element[0]; or form.elemnt[0]
  // pass = // // [1]; // // ...
```

→ ye same y/a pr

• **More Events** (more event. html)

→ change Event.

let user = document.querySelector("user")
user.add______ ("change", function(event){
event.prevent
console.log("changed")
" " (this.value);

//when register is clicked then above 2 wala chalega
//after giving values or jese hi humne user vale
ko chodh dia as screen ke kahi bhi click as whenever
initial state change to output trigger.

→ input event:
The input event fires when the value of an
<input>, <select> or <textarea> element has changed.
(छोटी सी मोटी कोई input हो vo show hogi)
(As when कुछ कुछ value enter kr rhe then vo change
ho rhi but change vala तभी trigger when हम us
box se bahar click)
(Non-character & arrow keys can't trigger it)
→ user.add______ ("input", function(event){
//same as above

* Typescript activity.

+ Assign: re (mouseleave)
p1.innerText = user.value.

* To do delete:
ul add__ ("click", function(event){
> console.log(event.target.nodeName)
↓
किसे पे click kia
> console.log(event.target)
// li (if clicked on elements in list) & button.delete (if on button)
// button.delete → यह है class of button
> if(event.target.nodeName == "BUTTON"){
let x = event.target.parentElement;
x.remove
console.log("deleted")
}

event delegation

Part 11

Event bubbling

→ nested element के event listener ko trigger to vo uske upper vala parent ko bhi trigger karta.

→ like पहले div we

→ in 3 के लिए alag alag addEvent listener add.

→ To stop parent ko trigger के
event.stopPropagation.

Event delegation

// Parent ke upper event listener add.

// [illegible] Button. [illegible] Parent उठाते है । (ये Parent का concept है)