**Events in JavaScript**

Event is action / occurrence that happens in the browser, often triggered by user interactions / the browser itself.

Types of event in JavaScript

* **Mouse events**

click, dbclick, mousedown, mouseup, mousemove, mouseover, mouseout.

* **Keyboard events**

Keydown, keyup, keypress

* **Form events**

Submit, input, change, focus, blur

* **Window / document events**

load, resize, scroll, unload, DOMContentLoaded

* **Touch events**

touchstart, touchmove, touchend

* **Custom events**

Created using new Event() / customEvent.

**What is event propagation in JavaScript?**

Event propagation is a mechanism by which event flow through the DOM tree when event is triggered on a specific elements.

When event occurs, it doesn’t just trigger on the target element. Instead, it moves through different phases of the DOM hierarchy, enabling parent and child elements to listen to the event.

**What are the phases of event propagation?**

Event propagation occurs in 3 phases

Capturing Phase

The event travels from the topmost ancestor of the DOM tree down to the target element. In this phase parent elements can listen first to the event before it reaches the target.

Target Phase

The event reaches the target element where it was originally triggered. Any event listeners on the target element will execute.

Bubbling phase

The events starts at the target element and then propagates upward to its ancestors in the DOM hierarchy. Parent elements can listen to the event at last.

The order of phases is:  
**1. Capturing Phase → 2. Target Phase → 3. Bubbling Phase**

**What is event delegation and how does e.target help?**

Event delegation is technique where a single event listeners on a parent element handles events for its child elements using event bubbling. The e.target property refers to the actual element that trigger the event.

**What is e.stopPropagation ()?**

The e.stopPropagation () method is used to stop the propagation of event in the DOM. It prevents event from bubbling and capturing phases.

**What is the difference between e.stopPropagation () and e.stopImmediatePropagation ()?**

e.stopPropagation ()

It stop the event from propagation to parent / ancestors elements, but other event listeners on the same element will still run.

e.stopImmediatePropagation ()  
It stops the event propagation and prevents other event listeners on the same element from running.

**What happens if we call e.stopPropagation () in the capturing phase?**

If e.stopPropagation () propagation is called during the capturing phase, the event will stop propagating and will don’t reach the target element.

**Can we stop both bubbling and capturing phases?**

Yes, calling e.stopPropagation () will stop the event from propagating further in both bubbling and capturing phases.

However, e.stopPropagation () cant prevent the default behaviour of the event. To do that, we need to call e.preventDefault ()

**What is e.preventDefault ()? How is it different from e.stopPropagation ()?**

e.preventDefault ()  
Prevents the default behaviour of event e.g, stopping a link from navigating / a form from submitting.

e.stopPropagation ()  
Stops the event from propagating through the DOM.

**Can e.stopPropagation () and e.preventDefault () be used together?**

Yes, we can be used together. e.preventDefault () stops the default behavior. e.stopPropagation prevent event propagation.

**Why is event.stopPropagation () sometimes avoided?**

Using event.stopPropagation () indiscriminately can interfere with other event listeners on parent / ancestor elements, making the event flow unpredictable. A better alternative is to use event delegation / properly manage event listeners.