NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

PROGRAM: SOFTWARE ENGINEERING

WEB ENGINEERING

Assignment No. 02

SUBMITTED BY:

Name: Ahmed Ali

Roll No: 22P-9318

Section: BS(SE)-6B

INSTRUCTOR NAME: Sir Umer Haroon

A DEPARTMENT OF COMPUTER SCIENCE

A detailed report explaining:

- **The benefits of event delegation over direct event binding. When and why stopPropagation() is necessary.**
- **The differences between bubbling and capturing and their practical implications.**

REPORT

Why JavaScript?

Think of JS as the magic stick of the web.

Without it, a website is just a pretty poster. But with JS, You can turn that poster into a fully interactive app. From handling clicks, to validating forms, showing animations, or updating the cart in real time without reloading the page, it is all JS

- It runs in the browser (no installation needed)
- Fast, responsive, and lightweight
- It powers 95% plus of websites today

Benefits of Event Delegation Over Direct Binding

Think you are managing 100 buttons on your page if you assign a separate click event to each one, your browser is doing 100 jobs but what if you just assign one event to their parent container and that one listener watches for events bubbling up

That is event delegation. Instead of micromanaging, you delegate just like a good manager.

Benefits:

- Efficient memory usage (fewer event listeners)
- Better performance (especially for dynamic content)
- Easy to manage events for elements added dynamically (e.g: new products)

When and Why is stopPropagation() Needed?

Sometimes, you click a button and it unintentionally triggers a function in its parent.
It is like pressing a doorbell and accidentally setting off a whole alarm system.
With stopPropagation(), you stop the event right there. It is saying, “Hey, stop right here. This interaction ends with me.”

Use it when:

- You do not want parent containers reacting to a child click (e.g: remove button inside cart triggering the whole cart logic)
- You need precision in user interaction

Bubbling vs. Capturing

Think of events in the DOM like passing a message

- Capturing Phase: The message starts from the top (window → document → html → body → down to your button). This is like a top down inspection.
- Target Phase: The event reaches the actual element you interacted with.
- Bubbling Phase: The event then bubbles back up the DOM tree.

Example:

You click a “Remove” button:

- In capturing, the parent sees the event first.
- In bubbling, the child sees it first.

Practical Implications:

- Capturing is useful when you need to intercept an event before it hits the element (rare)
- Bubbling is more common and intuitive default in most browsers

In assignment you can toggle between the two depending on your needs.
