Event bubbling in JavaScript is a mechanism where an event triggered on a child element propagates upward through its ancestors in the DOM. It allows parent elements to respond to events triggered by their child elements.

Propagation Direction: In event bubbling, the event starts at the target element and propagates upward through its parent elements to the root of the DOM.

Default Behavior: Event bubbling is enabled by default in JavaScript.
Event Listeners: If multiple event listeners are attached in the bubbling phase, they are executed in sequence, starting from the innermost target element

Event Triggering: The click event is triggered on the child element (button), initiating the event propagation.

Event Capturing: In the capturing phase, the event propagates from the root of the DOM down to the target (child). However, no listeners are explicitly set to handle events in this phase in the given code.

Event Bubbling: After reaching the target element (child), the event enters the bubbling phase, propagating back up through the DOM tree to the parent (parent).

Listener Behavior: Event listeners are attached to both parent and child elements using addEventListener. By default, these listeners respond during the bubbling phase unless the capture option is set to true.

Execution Order: When the button is clicked, the child listener executes first (console.log("Child")), followed by the parent listener (console.log("Parent")) as the event bubbles up.

Event Bubbling: Clicking the child triggers the event to propagate upward, activating listeners on the parent and grandparent.
Console Output: The order is "Child Clicked", "Parent Clicked", "Grandparent Clicked", showing the bubbling flow.

Event Object: Each listener receives the event object e, which includes details like e.target and methods like e.stopPropagation().

CSS Structure: Visual borders and sizes make the DOM hierarchy and event propagation clear.

How to Stop Event Bubbling?

To stop event bubbling, you can use the event e.stopPropagation() method in the event handler. This prevents the event from propagating to parent elements, so only the target element's event listener is triggered.

Event Bubbling Stopped: e.stopPropagation() prevents the event from bubbling to parent elements.

Independent Logs: Only the clicked element logs its message (e.g., "Child Clicked"), as bubbling is stopped.

Nested Visualization: CSS borders clearly show the DOM hierarchy and event flow.

Unique IDs: Each element has a unique ID for easy event listener management.

Practical Use: Useful for isolating behaviors, like preventing dropdowns or modals from closing when interacting with inner elements.
Use Cases of Event Bubbling

Delegated Event Handling: Instead of adding event listeners to multiple child elements, attach one to their parent and handle the events as they bubble up.

Simplified Code: Reduces redundancy and improves performance when handling events for dynamic or large DOM structures.