

Event listeners

Capture : starts from outer layer to the inner layer

Bubble : starts from the innermost layer to outer layer

1. Capturing phase → from window down to the target
2. Target phase → event reaches the actual element
3. Bubbling phase → from target back up to window

The browser does **not** decide the path based on your capture: true/false. The path is already fixed. The event will always travel from outermost → target → back out.

capture: true or false only decides **when your listener runs**, not how the event travels.

So let's build your scenario.

Imagine:

```
<div id="layer1">
  <div id="layer2">
    <div id="layer3">
      <div id="layer4">
        <button id="target">Click</button>
      </div>
    </div>
  </div>
</div>
```

Now suppose:

- layer1 → capture: true
- layer2 → capture: false
- layer3 → capture: true
- layer4 → capture: true
- target → doesn't matter for now

You click the button.

What happens?

Phase 1: Capturing phase (outer → inner)

The browser walks the tree:

window → document → html → body → layer1 → layer2 → layer3 → layer4 → target

Now it checks which listeners should fire in capturing phase.

- layer1 (capture: true) → fires
- layer2 (capture: false) → does NOT fire
- layer3 (capture: true) → fires
- layer4 (capture: true) → fires

Important:

layer2 is NOT skipped in the traversal. It just doesn't execute a capturing listener because you didn't register one.

Phase 2: Target phase

Target's listeners fire (capture listeners first, then bubble listeners).

Phase 3: Bubbling phase (inner → outer)

Now the event climbs back up:

layer4 → layer3 → layer2 → layer1 → body → html → document → window

Now only listeners registered with capture: false run.

So:

- layer4 (if it has capture: false listener) → fires
- layer3 (if capture: false) → fires
- layer2 (capture: false) → fires
- layer1 (capture: false) → fires

So your specific case:

If layer2 has capture: false and the others have capture: true, then:

During capture phase:

- layer1 fires
- layer3 fires
- layer4 fires
- layer2 does NOT fire

During bubbling phase:

- layer2 fires
- others won't (unless they also have bubble listeners)

There is no “break” in the propagation. The event flow is independent of your listener options.

That's the mental bug many people have.

Now here's the deeper mental model:

Think of the event as a train moving along tracks that are already laid.

capture: true means: “Let me get on the train while it's going down.”

capture: false means: “Let me get on the train while it's coming back up.”

You're not controlling the track. You're just choosing which direction to intercept the train.

The screenshot shows a browser window with developer tools open. On the left is a code editor with a file named 'script.js' containing the following code:

```
JS script.js
const divs = document.querySelectorAll("div")

document.addEventListener("click", e => {
  if (e.target.matches("div")) {
    console.log("hi")
  }
})

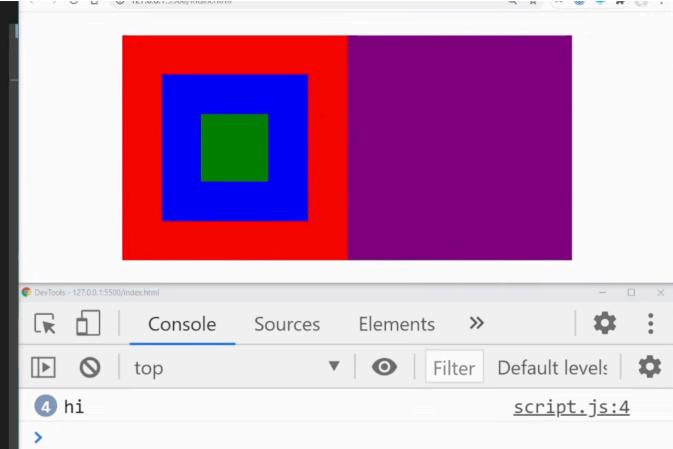
const newDiv = document.createElement("div")
newDiv.style.width = "200px"
newDiv.style.height = "200px"
newDiv.style.backgroundColor = "purple"
document.body.append(newDiv)
```

On the right, there is a visual representation of three nested div elements. The innermost div is green, then blue, then red. Below this is a screenshot of the browser's DevTools Console tab. The console shows the output of the script: a single line of text 'hi' with a timestamp 'script.js:5' and a number '9' indicating the line number.

```
JS script.js
const divs = document.querySelectorAll("div")
addGlobalEventListener("click", "div", e => {
  console.log("hi")
})

function addGlobalEventListener(type, selector, callback) {
  document.addEventListener(type, e => {
    if (e.target.matches(selector)) callback(e)
  })
}

const newDiv = document.createElement("div")
newDiv.style.width = "200px"
newDiv.style.height = "200px"
newDiv.style.backgroundColor = "purple"
document.body.append(newDiv)
```



```
const child = document.querySelector(".child")

grandparent.addEventListener("click", e => {
  console.log("Grandparent Bubble")
})

parent.addEventListener("click", printHi)

setTimeout(() => {
  parent.removeEventListener("click", printHi)
}, 2000)

child.addEventListener("click", e => {
  console.log("Child Bubble")
})

function printHi() {
  console.log("Hi")
}
```

