Below is a **clean, simplified, exam-ready, and internally accurate explanation of Event Bubbling**, written in the **same depth + format** you've been following, with **clear separation of notes and code**, and **conceptual clarity suitable for NoteLM / revision / viva**.

🔄 Event Bubbling in JavaScript

(Concept · Internal Working · Execution Order · Examples)

1️⃣ What is Event Bubbling?

Event bubbling is a mechanism in JavaScript where an event **starts from the element on which it occurs (target element)** and then **propagates upward** through its parent elements in the DOM hierarchy, until it reaches the root (`document`).

📘 Simple Definition

> Event bubbling means that when an event happens on an element, it first runs on that element and then runs on its parent, then its parent's parent, and so on.

2️⃣ Why Event Bubbling Exists (Internal Reason 🧠)

Browsers implement event bubbling to:

- Allow **centralized event handling**
- Enable **event delegation**
- Reduce the number of event listeners
- Improve performance

Without bubbling, each element would need its **own isolated event handling**, which would be inefficient.

3️⃣ Phases of Event Propagation (VERY IMPORTANT ⚠️)

Every DOM event passes through **three phases**:

🔹 1. Capturing Phase (Top → Down)

- Event travels from `document` → `html` → `body` → target
- Rarely used
- Enabled using `{ capture: true }`

🔹 2. Target Phase

- Event reaches the **actual element** where it occurred
- Both capture & bubble listeners can run here

🔹 3. Bubbling Phase (Bottom → Up) ✅ *(Default)*

- Event bubbles from **target → parent → ancestors → document**
- Default behavior in JavaScript

📌 Flow Diagram

```css
Document
  ↓ (capturing)
HTML
  ↓
Body
  ↓
Parent DIV
  ↓
Child DIV (TARGET)
  ↑ (bubbling)
Parent DIV
  ↑
Body
  ↑
HTML
  ↑
Document
```

4️⃣ Event Bubbling – Basic Example (2 Levels)

🧠 Scenario

- Click on **child div**
- Both **child and parent handlers execute**

✅ Code Example

```html
<div id="parent" style="padding:40px; background:#ddd;">
  Parent
  <div id="child" style="padding:20px; background:#66c2ff;">
    Child (Click me)
  </div>
</div>

<p id="output"></p>

<script>
const output = document.getElementById("output");

document.getElementById("parent").addEventListener("click", function () {
  output.innerHTML += "Parent clicked<br>";
});

document.getElementById("child").addEventListener("click", function () {
  output.innerHTML += "Child clicked<br>";
});
</script>
```

🔍 Output (Click on Child)

```nginx
Child clicked
Parent clicked
```

✔️ This confirms **event bubbling**

5️⃣ Event Bubbling – 3 Nested Levels (Execution Order)

🧠 Scenario

Clicking the **innermost element** triggers all ancestors.

✅ Code Example

```html
<div id="level1" style="padding:30px; background:#ff9999;">
  Level 1
  <div id="level2" style="padding:20px; background:#99ff99;">
    Level 2
    <div id="level3" style="padding:10px; background:#9999ff;">
      Level 3 (Click me)
    </div>
  </div>
</div>

<p id="output"></p>

<script>
const output = document.getElementById("output");

document.getElementById("level1").addEventListener("click", () => {
  output.innerHTML += "Level 1 clicked<br>";
});
```

```
document.getElementById("level2").addEventListener("click", () => {
  output.innerHTML += "Level 2 clicked<br>";
});

document.getElementById("level3").addEventListener("click", () => {
  output.innerHTML += "Level 3 clicked<br>";
});
</script>
```

🔍 Output

```mathematica
Level 3 clicked
Level 2 clicked
Level 1 clicked
```

✔️ Event bubbles **from inner → outer**

6️⃣ event.target vs event.currentTarget (KEY CONCEPT 🔑)

Property	**Meaning**
`event.target`	Element where event occurred
`event.currentTarget`	Element handling the event

Example Insight

```js
parent.addEventListener("click", function (event) {
  console.log(event.target);        // Child
  console.log(event.currentTarget); // Parent
});
```

7️⃣ Stopping Event Bubbling 🚫

Sometimes bubbling is **not desired**.

❌ Problem

Child click triggers unwanted parent logic.

✅ Solution: `stopPropagation()`

```js
child.addEventListener("click", function (event) {
  event.stopPropagation();
  console.log("Child only");
});
```

✔️ Prevents event from reaching parent

8️⃣ Bubbling vs Capturing (Quick Comparison)

Feature	Bubbling	Capturing
Direction	Bottom → Top	Top → Bottom
Default	✅ Yes	❌ No
addEventListener	`{ capture:false }`	`{ capture:true }`
Common Use	Event delegation	Rare cases

9️⃣ Real-World Use of Event Bubbling

✔️ Event Delegation
✔️ Dropdown menus
✔️ Tables & lists
✔️ Form handling
✔️ Dynamic UI components

> **Event delegation is impossible without bubbling**

🔟 Common Mistakes ❌

- Forgetting bubbling happens by default
- Confusing `target` and `currentTarget`
- Not stopping bubbling when required
- Using inline handlers + addEventListener together

1️⃣1️⃣ Exam / Interview Ready Answers 🎯

Q: What is event bubbling?

👉 Event bubbling is when an event propagates from the target element to its ancestors.

Q: Which phase is default in JavaScript?

👉 Bubbling phase.

Q: How to stop bubbling?

👉 `event.stopPropagation()`

1️⃣2️⃣ Key Takeaways ✅

✔️ Events propagate upward by default

✔️ Bubbling enables delegation

✔️ Execution order: inner → outer

✔️ `stopPropagation()` controls flow

✔️ Core concept for advanced JS

⏭️ What's the Natural Next Topic?

👉 **Event Capturing (Trickier but important)**

👉 **stopPropagation vs preventDefault**

👉 **Custom Events**

Type `next` and we'll continue step-by-step 🚀