Eventos

Event bubbling

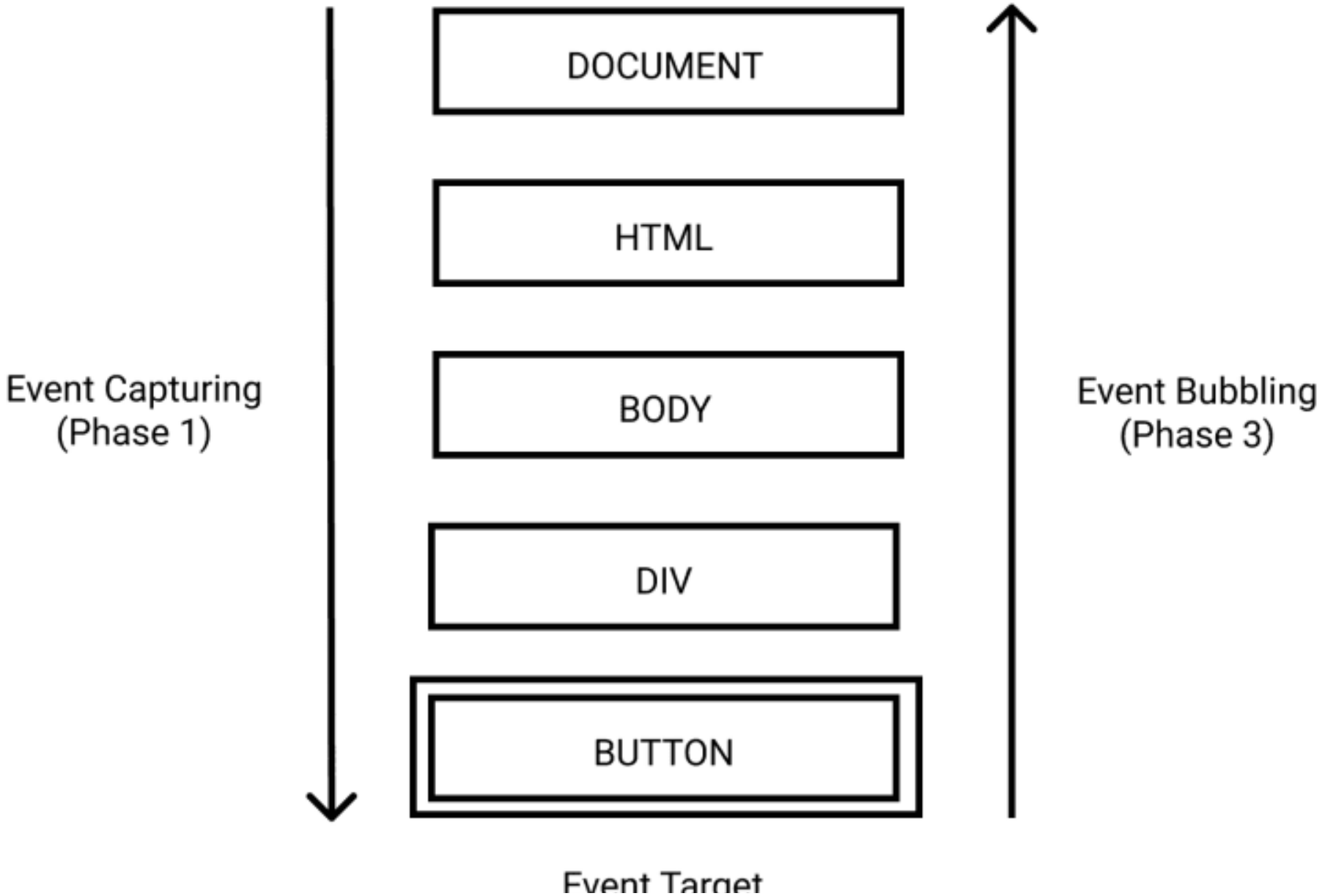
El event bubbling o burbujeo de eventos (por su traducción al español) es un método de propagación de eventos en la API del DOM.

Se da cuando activamos el evento de un elemento, y si su nodo padre tiene registrado otro evento, este último se activara automáticamente y así ira escalando en la jerarquía del DOM.

Fases del event bubbling

El Event bubbling pasa por 3 fases bien definidas:

- **Fase de Captura**: Se busca el elemento mas profundo en el DOM que tenga registrado un evento en su listener.
- **Fase de Target**: Ejecuta el evento del elemento en si.
- **Fase de Burbuja**: Verifica si los elementos padre de dicho elemento tienen eventos registrados en sus listeners, si es así, ejecuta dichos eventos de manera automática.


DOCUMENT
HTML
BODY
DIV
BUTTON
Event Capturing
(Phase 1)
Event Bubbling
(Phase 3)
Event Target

(Phase 2)

Explicarlo en palabras es un poco confuso, mejor verlo con ejemplos:

En HTML:

```
<div id="padre">
  <div id="hijo">
    <button id="button">
      Click me!
    </button>
  </div>
</div>
```

En JS:

```
const button = document.querySelector("#button");
const hijo = document.querySelector("#hijo");

button.addEventListener("click", () => {
  alert("Clicked!")
})

hijo.addEventListener("click", () => {
  alert("Clicked hijo!")
})
```

En el ejemplo:

- Al hacer click en el botón, activamos su evento, y mostramos una alerta.
- Como el contenedor padre del button tiene también registrado un evento, por Event bubbling este se activa, mostrando automáticamente una segunda alerta.

Justamente con este ejemplo se comprende mejor por qué se llama evento de burbuja, porque los eventos se van expandiendo desde el elemento mas profundo al más externo, como si simulara una burbuja.

Detener la propagación burbuja

Una buena manera de desactivar este comportamiento es usando el método `stopPropagation()` del evento en sí, de la siguiente manera:

```
button.addEventListener("click", (e) => {
  e.stopPropagation()
  alert("Clicked!")
})
```

El nombre del método es muy intuitivo, simplemente desactiva la propagación de eventos, dejando el resultado de una manera mas obvia y simple.

Modo captura

Por defecto, JavaScript se comporta con Event bubbling, pero podemos alterar el orden de los eventos cambiando al modo captura.

Esto se logra añadiendo un tercer parámetro a nuestro listener padre, como objeto, pasamos la propiedad `capture:true`:

```
const container = document.querySelector(".container");
const div1 = document.querySelector(".div-1");
const div2 = document.querySelector(".div-2");

container.addEventListener("click",() => {
  alert("container")
}, {capture:true})

div1.addEventListener("click",() => {
  alert("div-1")
})

div2.addEventListener("click",() => {
  alert("div-2")
})
```