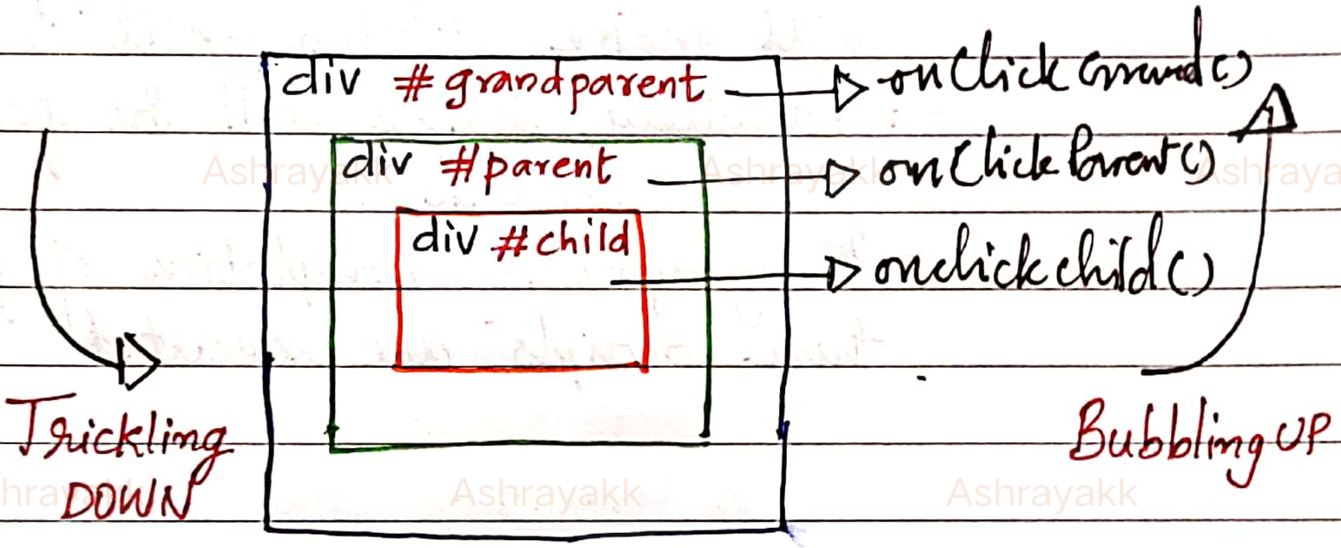
TOPIC - 06

EVENT BUBBLING & CAPTURING

Event bubbling & capturing are the two ways of event propogation in the ~~dom~~ -DOM tree.

Eg :- Suppose we have nested HTML elements.



Suppose $\boxed{\text{child div}}$ has a onClick Child() method So, when you click $\boxed{\text{child}}$, this method is called. Similarly for parent div and Grandparent div.

See the propogation of events.

In case of event bubbling, if we click on child div, onClickChild() method will called first. Then it moves up to the heirarchy. And it goes directly till the end of the ~~d~~ DOM.

[NB: Bubble always comes OUT. So, bubbling UP]

Event Capturing

→ just opposite of bubbling
→ also called as TRICKLING
→ It is capturing down the DOM tree.
→ Means, all event handlers will be triggered, but the order is :-

eg:- 1) onClickGrand()
2) onClickParent()
3) onClickChild()

Both bubbling & capturing are accepted. It depends upon the developer & usecase

addEventHandler('click', () ⇒ { }, useCapture)

↓ event
↓ callback
↓ true/false

- If useCapture is ~~true~~ false, then events will bubble up the hierarchy (if no attribute is passed)
- If useCapture is true, then events are captured / trickled down the hierarchy

- In W3C model, this cycle continues.

- It is based on the useCapture attribute, it is figured out wheather to capture or bubble out the events.

By default, 'Event Bubbling' is used.

Example

.html file

```
<div id ="grandparent">
    <div id = "parent">
        <div id ="child">
        </div>
    </div>
</div>.
```

Suppose we have 3 divs. Now, let's add event handlers to each of the divs and see how event propogates down the hierarchy.

```
★ document.querySelector ("grandparent")
    .addEventListener ('click', () => {
      console.log ("GrandParent Clicked!");
  }, false
  );
```

→ If nothing is passed here, it takes by default as 'false'. this means useCapture is false.

```
★ document.querySelector (" Parent  ")
    .addEventListener ('click', () => {
      console.log ("Parent Clicked!");
  }, false
  );
```

```
★ document.querySelector ("Child  ")
    .addEventListener ('click', () => {
      console.log ("Child clicked!");
  }, false
  );
```

In the above case, all the useCapture attribute is false.
This means capturing won't happen.
Event propogation will be like bubbling out.
So, output will be :-

Output

Child clicked!
Parent clicked!
GrandParent clicked!

If there is nothing given for useCapture, then also the output will be like above.

If it is 'true' for all case, then event capturing will occur. The events will be trickling down. Then the output will be :-

Output

GrandParent Clicked!
Parent Clicked!
Child Clicked!

e. StopPropagation(); is used to stop the propogation of events.