

Event

Bubbling



TARGET

Trickling

Capturing

Propagation



Quick Intro to Events

Event An event is a signal which will let you know that something has happened.

Event Handlers

A **function** that runs to react on these events.

Event Listeners

So there should be something which **listens** to this event and **attaches** the event handler..right??

so here comes **addEventListener**.

Syntax

```
target.addEventListener(event, function, useCapture)
```


target: The HTML element to which you wish to listen to the event and attach your event handler to.

event : event name to which you need to listen

function: The function to run when it listens to this event

useCapture : optional Boolean value that specifies whether the event should be executed in the capturing or bubbling phase.

- false - Bubbling phase true - Capturing Phase

If we do not mention any third parameter in `addEventListener()`, then by default event bubbling will happen.

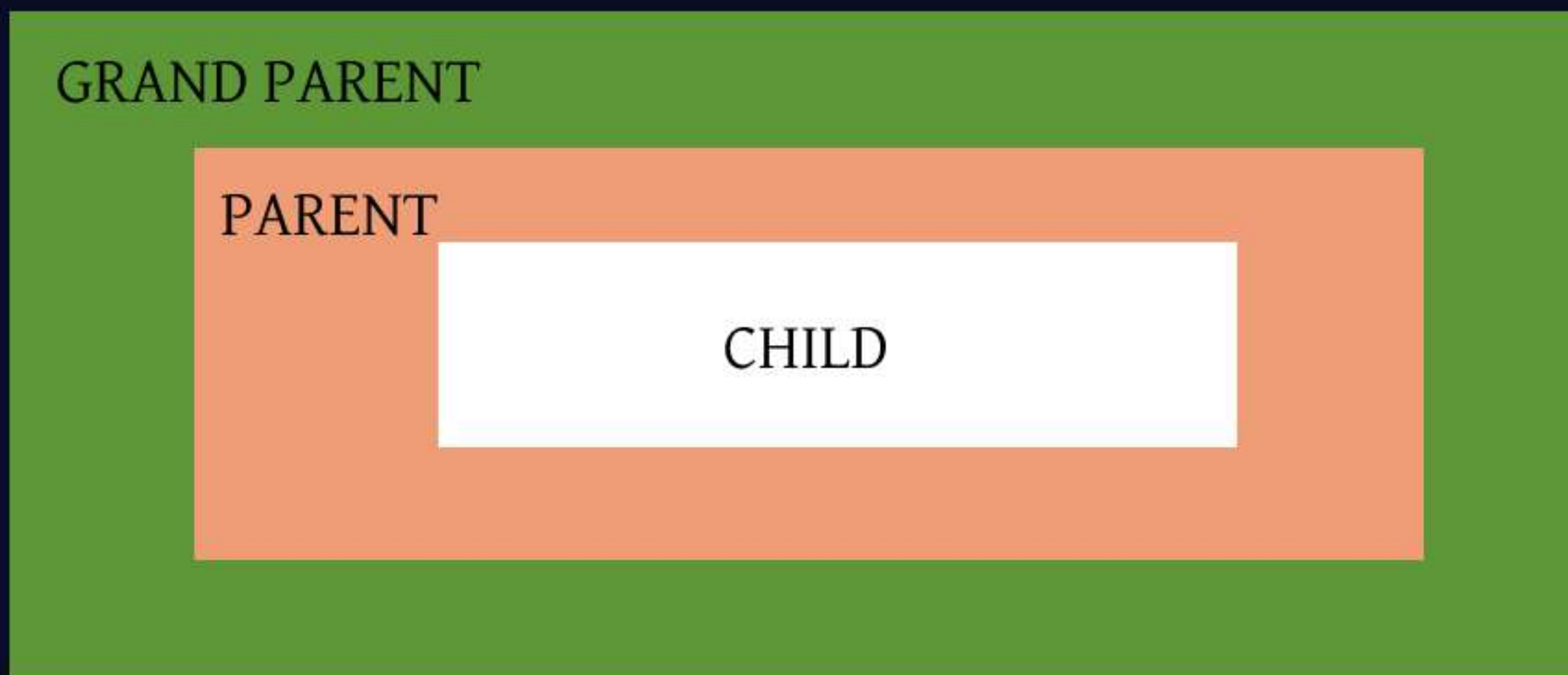
DOM Tree: When a web page is **loaded**, the **browser** creates HTML elements/Objects as **tree structure**. Which is called DOM tree or Document Object Model tree.

Now lets deep dive into main topic about event propagation

Event Propagation Event propagation is a way which defines how events propagate or travel through the DOM tree to arrive at its target.

Target : Element on which event has been triggered. Let's visualize it with simple example.

Event Bubbling



Now here we have three divs nested.

Lets assume each of them have their own event handlers. Check the example in next slide

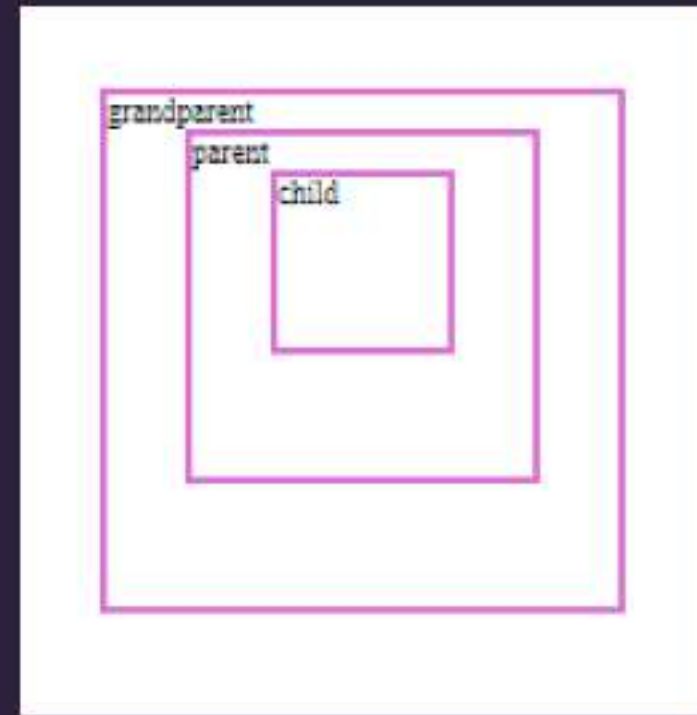

```
<body>
  <div class="grandparent"> grandparent
    <div class="parent">parent
      <div class="child">child</div>
    </div>
  </div>

  <script>

    let child = document.querySelector('.child');
    let parent = document.querySelector('.parent');
    let grandparent = document.querySelector('.grandparent');

    grandparent.addEventListener('click',()=> console.log('grandparent'));
    parent.addEventListener('click',()=> console.log('parent'));
    child.addEventListener('click',()=> console.log('child'));

  </script>
</body>
```



Click On Child Dev: Target element - Child div
ancestor elements - Parent and Grand parent div

OutPut :

```
child
parent
grandparent
```

The process is called “**bubbling**”, *Here the event gets Bubbled up from the target to up through parents of the DOM tree. So it starts from target element and moves up the DOM tree.*

Click On Parent Div:

Target element - Parent div

ancestor elements - Grand parent div

OutPut :



```
graph TD; parent --> grandparent;
```

Notice this now as the target is parent, It starts with **parent** and **bubbles up** to the **ancestors**. Here we have only one ancestor for parent which is grandparent div.

Event Capturing / Trickling

Here the third parameter in `addEventListener` usecase needs to be true to enable **event capturing effect** also called as **trickling**.

Event **capturing** is entire **opposite** of event **bubbling**...It starts from Ancestor elements and moves down till the target element.

Lets take the same example of nested divs and click on child element.
Check the example in next slide.

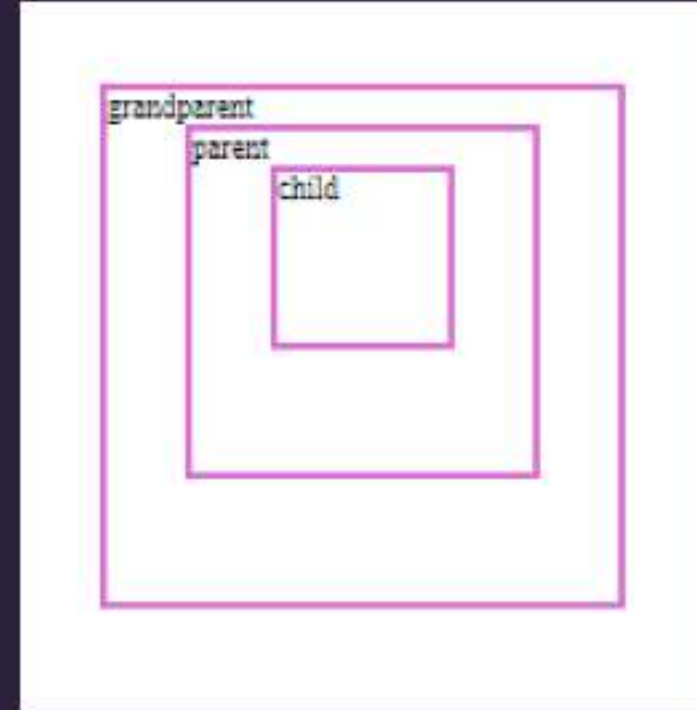

```
<body>
  <div class="grandparent"> grandparent
    <div class="parent">parent
      <div class="child">child</div>
    </div>
  </div>

  <script>

    let child = document.querySelector('.child');
    let parent = document.querySelector('.parent');
    let grandparent = document.querySelector('.grandparent');

    child.addEventListener('click',()=> console.log('child'),true);
    parent.addEventListener('click',()=> console.log('parent'),true);
    grandparent.addEventListener('click',()=> console.log('grandparent'),true);

  </script>
</body>
```



Click On Child Dev: Target element - Child div
ancestor elements - Parent and Grand parent div

OutPut :

```
grandparent
parent
child
```

Since capturing will start from ancestors and move downwards till it finds the target element. so ancestor Grandparent -> Parent and then it finds target element -> child...stops it.

Click On Parent Div:

Target element - Parent div

ancestor elements - Grand parent div

OutPut :



So it will **start** with ancestor and trigger it and then **moves down** the **DOM tree** and triggers the parent div which is our target element...so it just stops here and doesn't proceed further

Event Bubbling and Capturing together

So in this case first capturing phase happens till it reaches the target element and then bubbling phase happens.

Now lets take it next step...what if I give few elements with usecapture true and some with false?? meaning some with **capturing** and **bubbling**

Check the example in next slide


```
<body>
  <div class="grandparent"> grandparent
    <div class="parent">parent
      <div class="child">child</div>
    </div>
  </div>

  <script>
    let child = document.querySelector('.child');
    let parent = document.querySelector('.parent');
    let grandparent = document.querySelector('.grandparent');

    // Event Bubbling and capturing

    grandparent.addEventListener('click',()=> console.log('grandparent'),false); //Bubbling
    parent.addEventListener('click',()=> console.log('parent'),true); //capturing
    child.addEventListener('click',()=> console.log('child'),true); //capturing

  </script>
</body>
```




Lets click on child event

1)Phase1 -**capturing** .. So here in capturing phase it will start from ancestors till it reaches target elements and checks only with elements set to capturing and skip the bubbling.so it starts with grandparent>parent>child.but grandparent is using bubbling.it skips it so the output for phase1 will parent>child...we have reached target element ...phase 2 starts....

2) phase2 - **bubbling**...So in this phase it go up the dom tree from target to ancestors and check starting from child
>parent>grandparent...so you have set child and parent as capturing so it will skip those two elements and move up and gives grand parent.

OutPut :



```
parent
child
grandparent
```

Summary

- Event Bubbling starts from target element and travels up through the DOM tree.
- Event Capturing travels down through the DOM tree till it reaches target element.
- Event capturing is also called as Event trickling.
- If we use both bubbling and capturing both then it starts with capturing phase and moves to Bubbling phase.

Did you find it helpful??



Like this post!



Share with your friends



Save it for later



Follow for more!



@startwithmani



@M.serisha Kothapalli