@startwithmani

@M.serisha Kothapalli

Stop default actions

stop immediate propogation

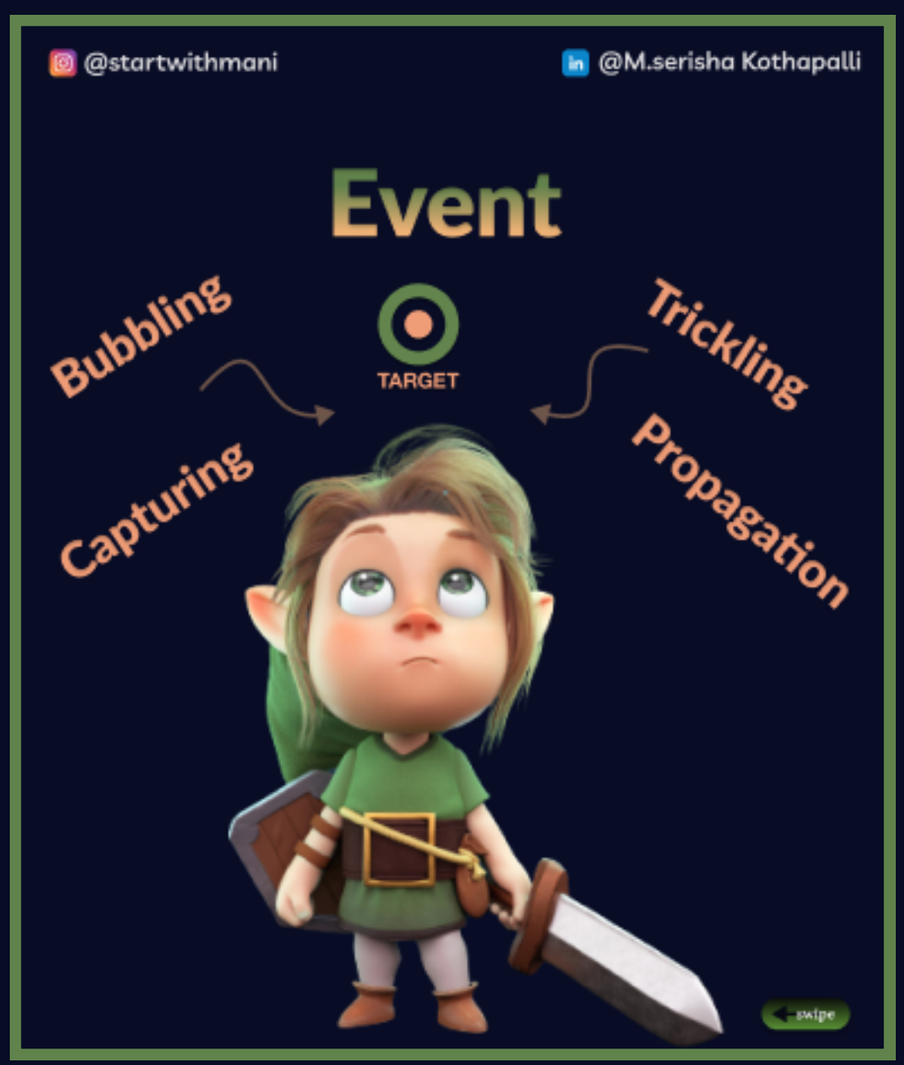
prevent default

stop propagation

swipe

To have a better understanding of this post, I would highly recommend you to check this below posts on **event propagations** and **event delegation**. Check the links in description and proceed further.

If you are already familiar with the above concepts, Feel free to get into the next slides directly.



Browser Default Actions

Browsers have few **default behaviors** for different events. events will automatically bring up certain actions. But what does it mean??

- Toggling a checkbox is the default action of clicking on a checkbox
- when a user clicks a submit button on a form, the form is submitted to a URL that was specified in action attribute by default.
- When you click a link - an anchor tag with a href attr, the default action is to navigate on the browser to the clicked link.

Just think of it, are you explicitly doing something or writing any code to make the above things work??

No right these are by default behaviours which happens on your events.

swipe

@startwithmani

@M.serisha Kothapalli

Prevent Default Browser Actions - preventDefault

Sometimes, while handling events in JavaScript, you don't want the default browser action to occur. Instead of that, you want to perform another action.

How come you can prevent these default actions??

```
<body>
      <a href="https://www.javascript.com/">Javascript</a>
</body>

<script>

        let a = document.querySelector('a');

        a.addEventListener('click',(e)=>

        {
            e.preventDefault();

            console.log('clicked');

        })
</script>
```

Output:

clicked

As you can see in the above example, I have used **preventDefault method** of the respective event to prevent the default action of browser. So on clicking the link, it **doesn't navigate** to respective link.

swipe

@startwithmani

@M.serisha Kothapalli

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.

Default propagation

Whenever the event triggers on the target element -

The event gets Bubbled up/**movesup** from the target to up through parents of the DOM tree. So it starts from target element and moves up the DOM tree. This is called **Bubbling.**

Event **capturing** is entire **opposite** of event **bubbling**...It starts from parent elements and **moves down** till the target element. You can enable capturing using **usecapture**

So when an event has been triggrered, **Propagation** of the event is something which happens by **default.**

swipe

Default propagation Example

So in the below example, Clicking on child gives the output as shown below due to bubbling of the event.

OutPut :

```
child
parent
grandparent
```

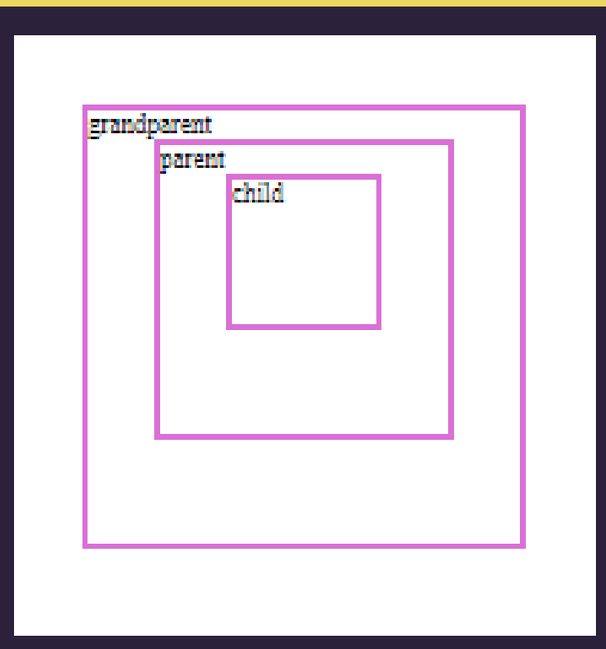
```html
<body>
    <div class="grandparent"> grandparent
        <div class="parent">parent
            <div class="child">child</div>
        </div>
    </div>


    <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>
```



Prevent Default propagation - stopPropagation

As you have seen in previous example, You wanted to stop this default propagation of either bubbling or capturing.How can we do it?? Lets utilize same example from previous 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',(e)=> {
            e.stopPropagation();
            console.log('child');
            });

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

    </script>
</body>
```

OutPut :

```
child
```

Notice this

So when you click on child this time, the event doesn't get propagated to parents like before.Using **stopPropagation()** method, it stop an events from propagating through the DOM tree

swipe

Why two??

Do you know that stopImmediatePropagation also stops the propagation of the event throught DOM tree.Then why two??

If several event listeners are attached to the same HTML element for the same event type like below. Just stressing on the point- Same event on same Html element with different event listeners. lets see with an example

```
<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 listener - 1

        child.addEventListener('click',(e)=> {
            console.log('child1');
            });

    //Event listener - 2

        child.addEventListener('click',(e)=> {
            console.log('child2');
            });

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

    </script>
</body>
```

Notice they are exceuted in order they have been added

```
child1
child2
parent
grandparent
```

swipe

@startwithmani

@M.serisha Kothapalli

stopPropagation for multiple listeners

```
<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 listener - 1

        child.addEventListener('click',(e)=> {

            e.stopPropagation();
            console.log('child1');
            });

    //Event listener - 2

        child.addEventListener('click',(e)=> {
            console.log('child2');
            });

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

    </script>
</body>
```

Notice this

```
child1
child2
```

Here we have added stoppropagation so that is the reason it has stopped the propagation to parents (parent and grandparent). But did you notice the second attached listener is still being executed? what if you don't want it to happen?

swipe

stopImmediatePropagation for multiple listeners

Lets replace same example with stopImmediatePropagation and notice now it has just printed child1 and stopped the propagation immediately there itself. It didn't print the second event listener which was attached

```
<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 listener - 1

        child.addEventListener('click',(e)=> {
            e.stopImmediatePropagation();
            console.log('child1');
            });
// Event listener - 2

        child.addEventListener('click',(e)=> {
            console.log('child2');
            });

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

    </script>
</body>
```

Notice this

```
child1
```

swipe

Summary

- **preventDefault** - It prevents the default action of the event in the browser from happening.

- **stopPropagation** - It prevents propagation of the current event through the DOM tree (Bubbling and Capturing)

- **stopImmediatePropagation** - It prevents propagation of the current event through the DOM tree and also Prevents other listeners of the same event from being called.

swipe

Did you find it helpful??

Like this post!

Share with your friends

Save it for later

Follow for more!

@startwithmani

@M.serisha Kothapalli