I. Kartasov,
2nd students of School of Business of BSU
Scientific supervisor:
senior lecturer
E. Grinevitch

DOM EVENTS AND EVENT LISTENERS. BUBBLING AND CAPTURING

To some, there may seem to be an elusive lack of difference between the instance of an event happening and a certain event listener being triggered.

When an event happens is the starting point of the whole process of Event Handling in JavaScript. But it is the user interaction (in most cases) that leads to an event actually happening and it is a Script that handles the event initiated via a user action.

So when an event happens it means that the user did something like click, focus or even probably simply move the cursor of their mouse. Once that happened, an event (let's for the sake of an example assume it to be a Click Event) was triggered by the user, the JavaScript gets a signal to run its many event listeners. Do note, however, that there may be dozens of event listeners surrounding one type of event (and thus also surrounding so much as a single click or the slightest stir of the cursor). So how do you make sense of all the havoc that goes off?

Well, let's assume that a user clicked a certain HTML element, let's call this element element A (so the user triggered a click event on element A). A single click event happened. Well, in our document there may actually be many event listeners that listen and react to that click. But here is the thing: only the Event Listeners of the Parent Elements of element A as well as element A itself will ever know that a click event happened on element A. The children of element A won't get any signal.

So what do we have as of now? A click was triggered on element A and all the Event Handlers of the ancestors of element A will know and reach to this click. But they aren't going to react to it all at once. In fact there is a very elaborate and sophisticated order in which each and every one of those Event Listeners is going to be triggered.

These circles actually represent event listeners. Some Event Listeners are going to be triggered during the Capture phase (more on that in a moment) and others during the Bubbling Phase (same here). Show the distinction in the picture.

But before we dive into exploring the difference between Capture and Bubbling Phases let's get some terminology straight.

Each of those Event Listeners on Parent Elements will have a so called Event Object that stores lots of data. When an Event is Triggered by the user (it happened not so long ago when they clicked Element A, remember?) there is only one element where the Event really happened. If a click happens on an element all the Event Objects on the many Event Listeners that listen to clicks are all going to have the same reference to that one unique element. In our case all the Event Listeners of the ancestors of element A are going to have the reference to Element A as that one HTML element where the user clicked (we can't really click at two places at the same time right?). To be precise those Event Listeners will store away the reference to Element A each in their own Event Object. That reference to element A is going to be called, quite straightforward really, target. I am going to show current Target later so it makes more sense.

After that the game begins, and dozens of different Event Listeners invoke many functions leveraging the power of just one click.

Well so what goes on really? As I already said there is going to be a certain order in which Event Listeners are going to be executed, and yep those are just functions.

Do think of this whole process as though we had a rubber ball that we throw into water, which can turn into a bubble as soon as it reaches a certain depth.

Why that? Let me show a demo for that.

Well, you should know that we can have many Event Listeners on the same HTML element (in our case on the same ancestor of good old element A for example). Also there are two Types of Event Listeners. The first type is the Sort of Event Listener that is triggered during the Capture Phase, the other is triggered during the Bubbling Phase.

So what that means? Draw back to our Rubber Ball analogy. The Capture Phase is very similar to when we throw the Ball into the water and it sinks, penetrating deeper and deeper. This way when the user did click the element A the very first Event Listener to be triggered is going to be the Capture Phase Type of Event Listener that is attached to the Uppermost ancestor of element A (this is often such huge elements as HTML, body, document or even window). So why the uppermost ancestor during the Capture phase?

Simple. Because it is the highest element in terms of hierarchy and the Capture phase goes before the Bubbling phase (that is just a rule).

Just like the ball sinks, we too are going to climb down that ladder of ancestors to the ancestor just below the uppermost and see if the next (the one below) ancestor has the Capture Type of Event Listener. If it does we are going to invoke it (Event Listener is just a function) and climb lower still or if it doesn't we will simply have to go climb down to the ancestors below without invoking an Event Listener. This is kind of why we call it Capture actually.

Also do note that the Event Object of every ancestor, when we execute the Event Listener of that ancestor is also going to store the reference to that ancestor in the currentTarget property. This way currentTarget is going to be the same element (ancestor) as the one whose Event Listener is currently being executed. All right let’s go down. Take a look at how the currentTarget and target are going to change as we go down.

Eventually we are going to find our element A or target that the user clicked on. Then the browser is going to see if our element A itself had any Event Listeners attached to it. If so, the browser is going to invoke them. Do note a peculiarity about invoking Event Listeners on the same element A that the event occurred on: the target of all Event Objects and the currentTarget reference of the Event Object of element A (if our element A happens to have an Event Listener) are going to be the same. This is also called the Target Phase by the spec. So you could identify this phase as the one in which currentTarget === target.

Then we say that an event bubbles. What that means? Well, the target reference in the Event Objects is still the same and so the target remains our element A. But we are in the Bubbling Phase already. Turning back to our rubber ball analogy, you could think of the moment when we reached element A as the point at which our rubber ball turned into a bubble and now, after sinking deep enough, it is going to travel all the way back to the surface where we started triggering the first Event Listeners of the Huge uppermost ancestors. At this point all the Bubbling Phase Type of Event Listeners are going to be invoked much in the same fashion as we previously observed the Capture Phase Event Listeners do their work, except with one difference: this time we are going to go from the direct ancestor of our element A to the uppermost ancestor of element A, invoking all Bubble Phase Kind of Event Listeners on the way to the top. The currentTarget inside Event Objects of our ancestors is going to vary from the deepest (direct) ancestor to the uppermost ancestor of Element A until we arrive at the very top, at the window perhaps running the very last Event Listener (if there is one attached to window, of course).

Here is a somewhat interesting detail though: think back to the time when our rubber ball turned into a bubble, yeap I am talking about the time when we reached element A at the end of the Capture Phase. Do think about it: element A can too have both Capture Phase type of Event Listener on it as well as the Bubble Phase type of Event Listener.

So logic is pretty straightforward here: as we are leaving the capturing phase the Capture Phase Event Listener is invoked first, then the Bubble Phase event listener is invoked on Element A.

It is confusing because the whole time while currentTarget is the same element as target we are actually in the Target phase. But that is probably more of a somewhat unnecessary technical detail at this point.

Well, here is another thing: whenever anybody tells you that an event can't bubble what they are trying to say is the following: if click events couldn't bubble our element A wouldn't be able to tell all the Bubble Type Event Listeners of its ancestors that they should execute and as a result they would just sleep lambently through the whole process (missing all the fun). The Focus event is a good example! So we can only catch it during the Capture Phase. What this means is that if we assign a Capture Phase kind of event Listener named John as well as a Bubble Phase kind of Event Listener named Bob to any of the ancestors of element A, element A, upon getting focus, will be able to tell John that it should start executing and John will do his job. However, because we say that Focus event doesn't bubble (I mean it is true it actually doesn't) so because of this event's inability to bubble, element A will not be able to tell Bob to execute. So Bob will be a useless Event Listener for the Focus Event because Bob is a Bubble Phase kind of event listener.

Well that is about it for the difference between capturing and bubbling.

Seems like one click took us on a journey of a million years, didn't it?