# Event Delegation

One of the things that took me a while to fully understand is event delegation and the associated concepts of `event bubbling` and `event targeting`.

The idea is that, when we add an event listener to an element, there are multiple events triggered and not just on the element the event is attached to.

If we take the following HTML. How can we attach a single event so that it will work in all `li` elements and those we might add in the future?

```
<ul id="menu">
  <li>item 01</li>
  <li>item 02</li>
  <li>item 03</li>
  <li>item 04</li>
  <li>item 05</li>
  <li>item 06</li>
  <li>item 07</li>
  <li>item 08</li>
  <li>item 09</li>
  <li>item 10</li>
</ul>
```

When an event is fired on an element that has parent elements (in this case, the `li` has the `ul` as a parent), modern browsers run two different phases — the capturing phase and the bubbling phase.

The capturing phase moves from the root of the document (the `html` element) to the element that triggered the event and executes all the matching event handler code that it finds along the way.

The bubbling phase the exact opposite happens, we start at the element we interacted with has a matching event and, if it doesn't it 'bubbles' upward in the chain, until it hits the root element of the page (`html`) executing the event handlers it finds along the way.

In modern browsers, by default, all event handlers are registered for the bubbling phase.

If, at any point, you want to stop the propagation use stopPropagation.

If we want to stop the default behavior of an element, for example stop links from navigating to the destination, we need preventDefault. Note that `preventDefault()` does not stop propagation, only the element's default behavior.

This example uses the default bubbling mechanism to create event delegation.

We set up a single click event handler in the `ul` element rather than the individual children. That way when we click on an `li` element we get the event we want. Furthermore, if we add `li` elements to the list to any depth, they will still work with our event.

```
const menu = document.querySelector('#menu');

menu.addEventListener('click', e => {
    'click'// What was clicked?
    let t = e.target;
    // if the list item was clicked (t.nodeName.toLowerCase() === 'li') {
      // prin'li'// print out the link
      console.log(t.innerHTML);
      // Do whatever else you need
      output.innerHTML = t.innerHTML;
  }
  // Don't follow the links
  e.preventDefault();
});
```

To set the event listener to capture events from the bottom down, add `true` as the second parameter of the event listener.

In cases where both types of event handlers are present, bubbling and capturing, the capturing phase will run first, followed by the bubbling phase.