Event Delegation Demo

During the event handling tutorial the students wrote code that looked like this.

```
const tasks = document.querySelectorAll("li");

tasks.forEach((task) => {
  task.addEventListener("click", () => {
    if (!task.classList.contains("completed")) {
      task.classList.add("completed");
      task.querySelector("i").classList.add("completed");
    }
  });

  task.addEventListener("dblclick", () => {
    if (task.classList.contains("completed")) {
      task.classList.remove("completed");
      task.querySelector("i").classList.remove("completed");
    }
  });
});
```

In that code you get a reference to all of the list items, loop over them and attach event listeners to each of them. This is a good first step in understanding how event listeners work but it isn't best practice. In this lecture

What is event delegation

The idea is that if we have a lot of elements handled in a similar way, then instead of assigning a handler to each of them – we put a single handler on their common ancestor. To illustrate this you can walk through the simple example included in this folder.

Event Delegation Demo

Start by walking through the `index.html` and showing what is there. You have an unordered list with 3 list items. At the bottom of the file we include `index.js`. To follow what we did in the tutorial we might start off by getting a reference to all of the list items, looping over them and adding a click event listener.

```
const items = document.querySelectorAll("li");
items.forEach(item => {
  item.addEventListener("click", event => {
    console.log(item.innerText);
    event.target.style.backgroundColor = event.target.innerText;
  });
});
```

At this point you can pose the following question: "What if you were to add a new list item at runtime, would it be able to handle a click event"? We can add the following code that will attach an click event for the add orange button and create a new list item.

```
function addOrange() {
  const orange = document.createElement("li");
  orange.innerText = "Orange";
  colors.appendChild(orange);
}

document.addEventListener("DOMContentLoaded", () => {
  const button = document.getElementById("btnAddOrange");
  button.addEventListener("click", addOrange);
});
```

Explain why the orange list item doesn't haven't the event handler. This is where event delegation comes into play so let's rewrite this by getting a reference to the parent which has an id of `colors`. In the handler we get `event.target` which allows us to find out where the event actually happened and how to handle it. When we use event delegation we use some type of clause to find out where the event came from.

```
const colors = document.getElementById("colors");
colors.addEventListener("click", event => {
  if (event.target.nodeName.toLowerCase() === "li") {
    event.target.style.backgroundColor = event.target.innerText;
  }
});
```