# 冒泡和阻止冒泡

冒泡是指元素事件处理中，会向上一层冒泡，导致上一层的元素也能捕获到这种处理行为。

```
<body>
    <div id="container1" class="container1">
        <button id="btn1">btn1</button>
    </div>
</body>
```

```
const body = document.body;
body.addEventListener("click", (event) => {
    console.log("body");
    console.log("event-target", event.target);
});
const container1 = document.getElementById("container1");
container1.addEventListener("click", (event) => {
    console.log("container1");
    console.log("event-target", event.target);
});
const btn1 = document.getElementById("btn1");
btn1.addEventListener("click", (event) => {
    console.log("btn1");
    console.log("event-target", event.target);
});
```

当点击 btn1 按钮时，上面 6 个 console.log() 都会执行。因为点击 btn1 时，会向上冒泡，由于 container1 元素和 body 元素都监听了点击事件，所以都会执行。

当给 container1 元素添加 event.stopPropagation() （阻止冒泡传播）后:

```
container1.addEventListener("click", (event) => {
    event.stopPropagation();
    console.log("container1");
    console.log("event-target", event.target);
});
```

再次点击 btn1，只会执行后面 4 个 console.log()。body 绑定的点击事件中的便不会执行了。点击 container 区域（btn1 以外区域），执行 container1 绑定的点击事件里面的代码了。

上面代码如果将 body 绑定事件里面内容改动一下：

```
body.addEventListener("click", (event) => {
    console.log("body");
    console.log("event-target", event.target);
}, true);
```

上面 true ，表示在捕获阶段执行，也就是说如果此时点击 container1，会先执行 body 绑定事件里面的 console.log()。 再执行 container1 绑定事件里面的 console.log()。

因为事件的流转阶段是先捕获，然后找到目标元素，再冒泡。