Bubbling

Interviewquestion-54

JS

Follow on in

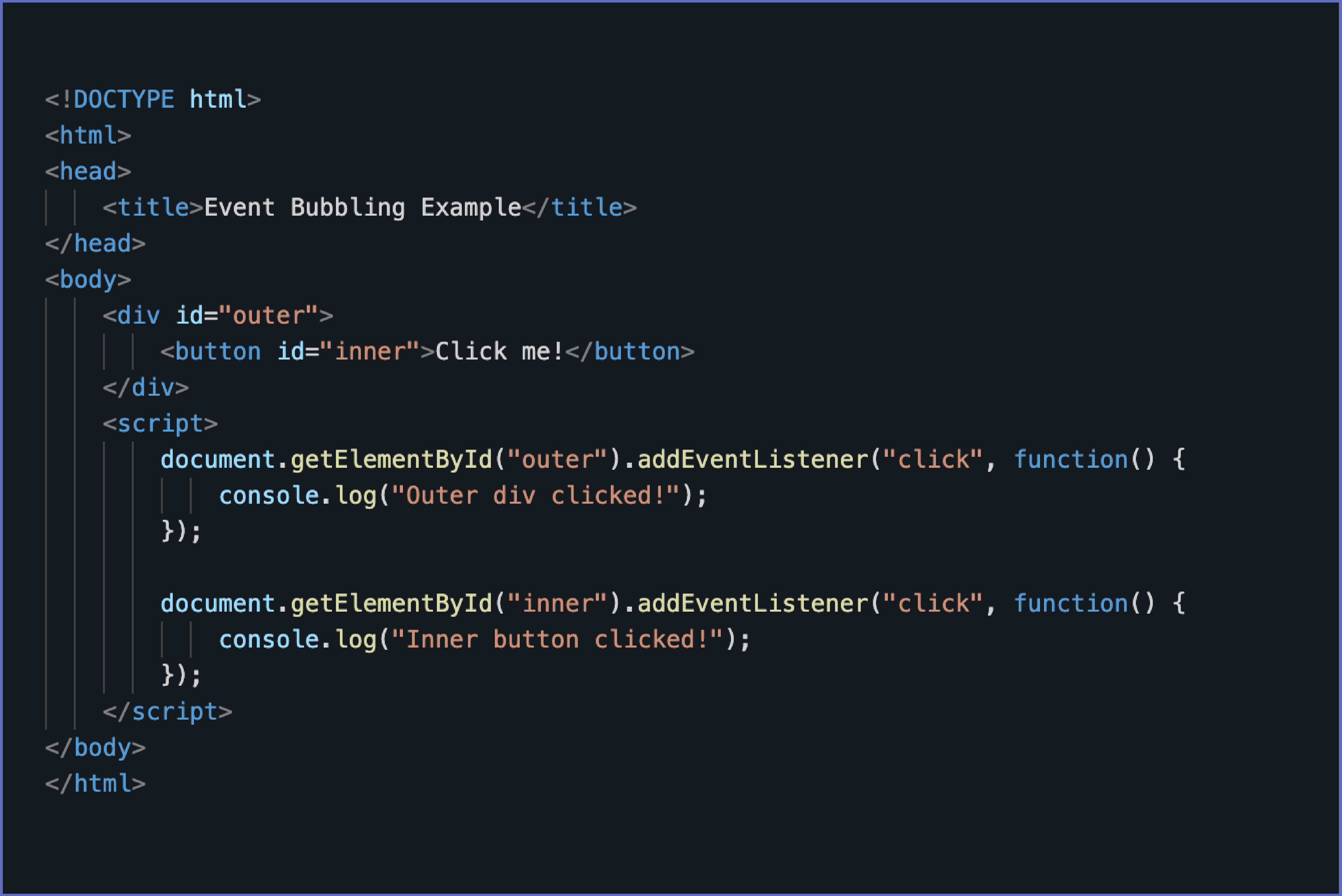
@Duvvuru Kishore

Bubbling

- Event bubbling is a key concept in JavaScript and DOM (Document Object Model) events.
- It refers to the order in which events are propagated through the DOM hierarchy, from the target element up to the root of the document.
- During this propagation, events "bubble up" from the target element to its parent elements, all the way to the document or window level.

Text is always not the best way to understand but example can make you understand the text in better way

Example to understand bubbling



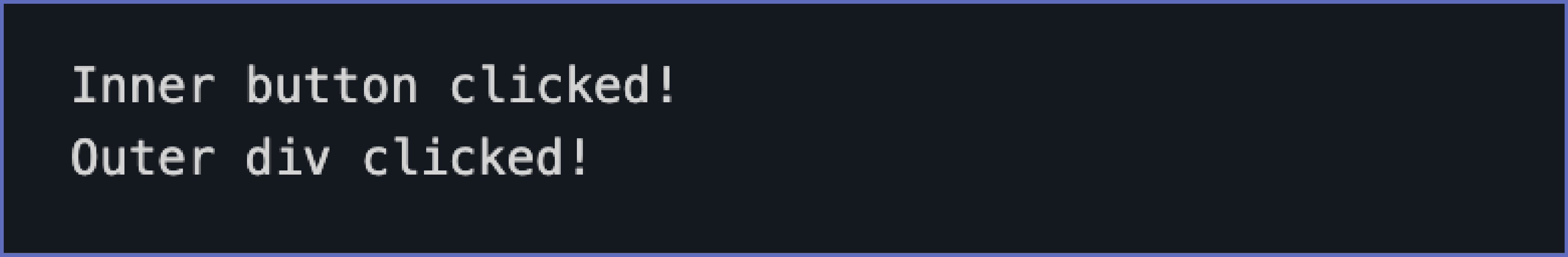
```html
<!DOCTYPE html>
<html>
<head>
    <title>Event Bubbling Example</title>
</head>
<body>
    <div id="outer">
        <button id="inner">Click me!</button>
    </div>
    <script>
        document.getElementById("outer").addEventListener("click", function() {
            console.log("Outer div clicked!");
        });

        document.getElementById("inner").addEventListener("click", function() {
            console.log("Inner button clicked!");
        });
    </script>
</body>
</html>
```

- Now if I click on button, not only button event listener ,even the div event listener will be called

- It’s falling in the approach of calling events from child to parent. which is bubbling.

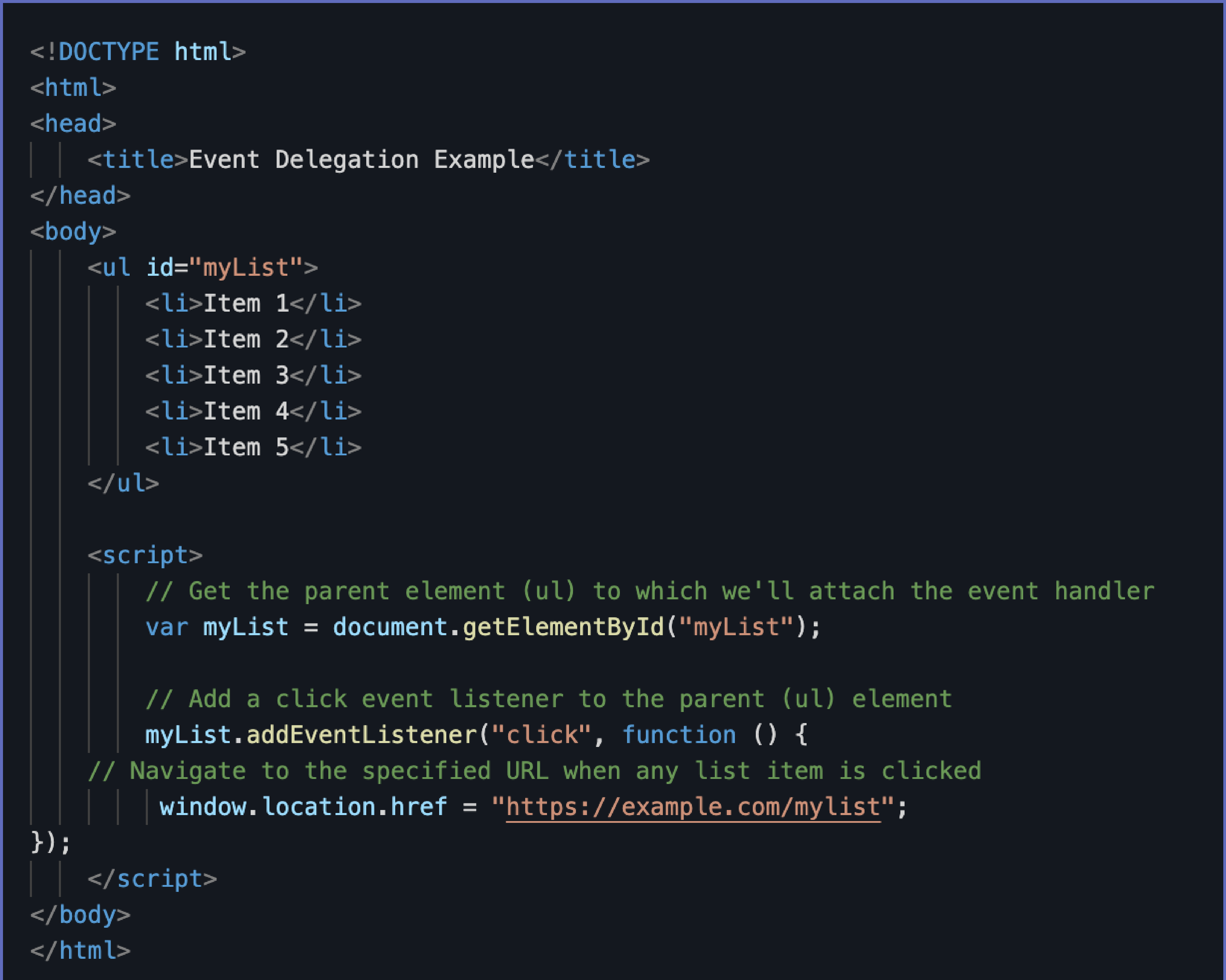
output on click of button



```
Inner button clicked!
Outer div clicked!
```

What is the need of bubbling ?

- Bubbling allows you to attach a single event handler to a common ancestor of multiple elements, which can be more efficient than attaching individual handlers to each element. This is useful for lists, tables, or any group of elements that share a common behavior.



```html
<!DOCTYPE html>
<html>
<head>
    <title>Event Delegation Example</title>
</head>
<body>
    <ul id="myList">
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
        <li>Item 5</li>
    </ul>

    <script>
        // Get the parent element (ul) to which we'll attach the event handler
        var myList = document.getElementById("myList");

        // Add a click event listener to the parent (ul) element
        myList.addEventListener("click", function () {
    // Navigate to the specified URL when any list item is clicked
         window.location.href = "https://example.com/mylist";
});
    </script>
</body>
</html>
```

we need to know something specific to bubbling which is event delegation. And we have used it in above example

Event Delegation

- Event Delegation is a pattern based upon the concept of Event Bubbling.
- It is an event-handling pattern that allows you to handle events at a higher level in the DOM tree other than the level where the event was first received
- In the example above we have used event on ul instead of each li. It‘s possible with event bubbling with delegation

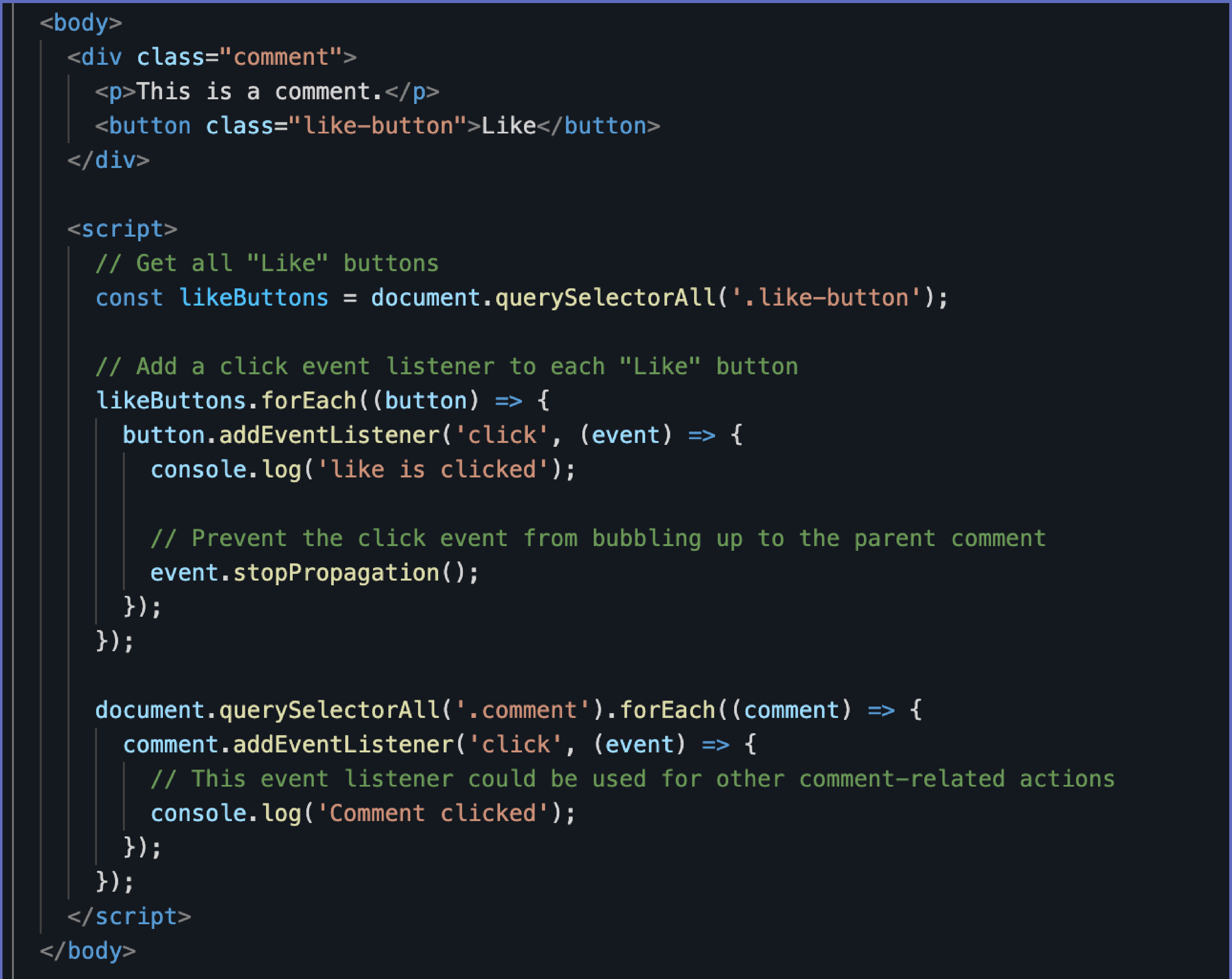
where to avoid bubbling?

Preventing Multiple Actions:

Example: Imagine a "Like" button within a comment section. Clicking the button should toggle the like state for that specific comment. If bubbling is not stopped, clicking the "Like" button may also trigger actions on the parent comment or the entire comment section, causing unintended behavior.

Solution:
To avoid this, you can call event.stopPropagation() or event.stopImmediatePropagation() within the event handler for the "Like" button to prevent the event from further bubbling up the DOM tree.

- Here with example below you see when like is clicked , the like element event listener is called and it’s parent (comment) event listener is not called as we added event.stopPropagation() in like listener call
- In this way you can avoid bubbling in the places we don’t need



```
<body>
  <div class="comment">
    <p>This is a comment.</p>
    <button class="like-button">Like</button>
  </div>

  <script>
    // Get all "Like" buttons
    const likeButtons = document.querySelectorAll('.like-button');

    // Add a click event listener to each "Like" button
    likeButtons.forEach((button) => {
      button.addEventListener('click', (event) => {
        console.log('like is clicked');

        // Prevent the click event from bubbling up to the parent comment
        event.stopPropagation();
      });
    });

    document.querySelectorAll('.comment').forEach((comment) => {
      comment.addEventListener('click', (event) => {
        // This event listener could be used for other comment-related actions
        console.log('Comment clicked');
      });
    });
  </script>
</body>
```

Follow on

@Duvvuru Kishore