

JavaScript – Event Handling

Vijesh M. Nair
Assistant Professor
Dept. of CSE (AI-ML)

Events

- **Events** are fired by actions that happen in a document, e.g. a user clicks a button or submits a form
- JavaScript programs are **event-driven**
 - The program waits for some user action
 - Clicking a link or a check box
 - Submitting a form
 - Loading a page
 - An action triggers an event – e.g. click, submit, or load events
 - Then a function is executed as a result of the event firing

Event Handlers

- **Event handlers** are the functions that are executed when a particular event fires
- We add GUI controls and event handlers to a Web page to make it interactive
 - The user interacts with a control – e.g., by changing the value in a form field or clicking on a submit button
 - The event handler function is notified of this event
 - Then the function executes

Some Events

- **Mouse events** are available on all HTML elements
- **Interface events** are fired after certain mouse or key actions, e.g., the submit event fires after a user clicks on a submit button

| <i>Mouse events</i> | <i>Interface events</i> | <i>Keyboard events</i> |
|---------------------|-------------------------|------------------------|
|---------------------|-------------------------|------------------------|

| | | |
|------------------|---------------|----------|
| click | blur | keydown |
| dblclick | focus | keypress |
| mousedown | change | keyup |
| mousemove | load | |
| mouseout | unload | |
| mouseover | submit | |
| mouseup | resize | |
| scroll | | |

Some Commonly Used Events

click

- An element is clicked once

mouseover

- The mouse moves over an element

change

- The value of an element changes

load

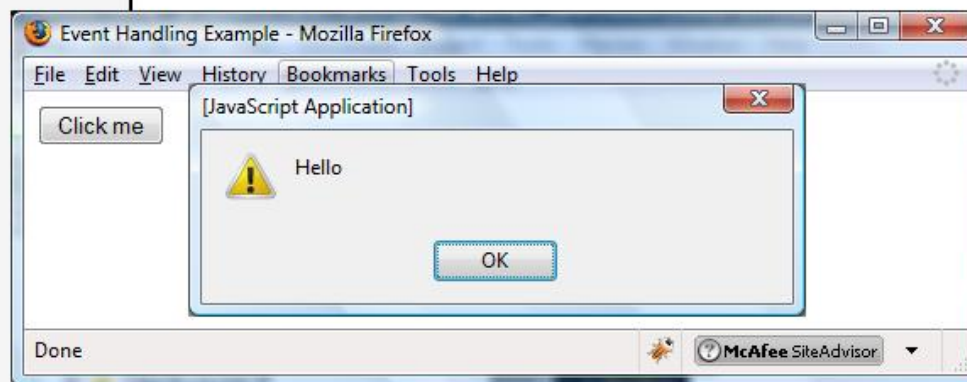
- A page has been completely loaded in the browser

submit

- A user submits a form

A Simple Example

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Event Handling Example</title>
  </head>
  <body>
    <form method="post" action="hello.aspx">
      <input type="button" value="Click me" onclick="alert('Hello');"/>
    </form>
  </body>
</html>
```



Add "on" in front of the event name

Event Handler Registration

Method 1: Inline event handlers

- Add an attribute to an HTML element
- Works in all browsers
- Deprecated because behaviour is not separated from structure
- The syntax to recognize an HTML event handler looks like this:

`<element attribute = "functionName()">`



The screenshot shows a web browser window with a single tab titled 'evnt1'. The address bar is empty. The page content is displayed in a light blue theme. The HTML code is as follows:

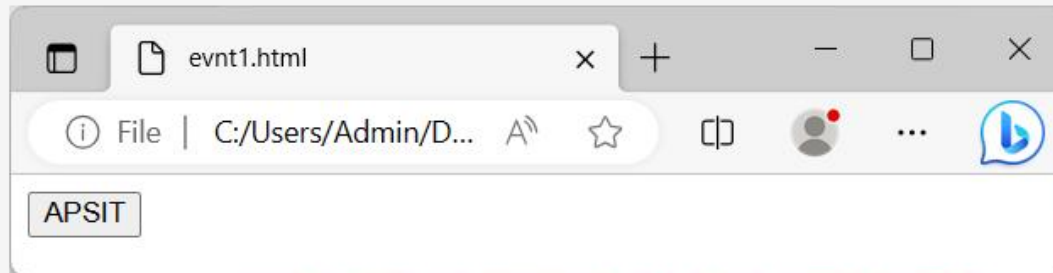
```
<html>
<head>
<body>
  <button onclick = "changeColor()"> APSIT </button> <!--function call-->
  <p></p>
<!--javascript code-->
<script type="text/javascript">
  function changeColor(){
    document.querySelector("button").style.backgroundColor = "blue"; //change background color
    document.querySelector("button").style.color = "white"; //change font color
    document.querySelector("p").innerHTML = "Great! The button changed its color." //add text
  }
</script>
</body>
</html>
```

The status bar at the bottom of the browser window shows 'Ln 5, Col 12', '100%', 'Windows (CRLF)', and 'UTF-8'.

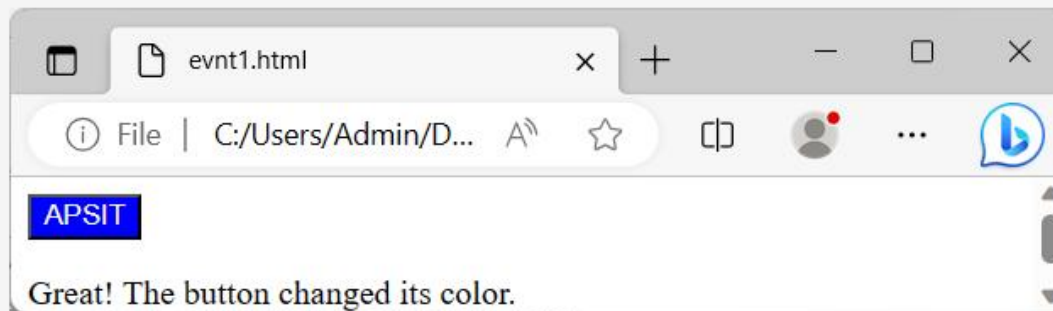
Event Handler Registration

Method 1:

Before Click



After Click



Event Handler Registration

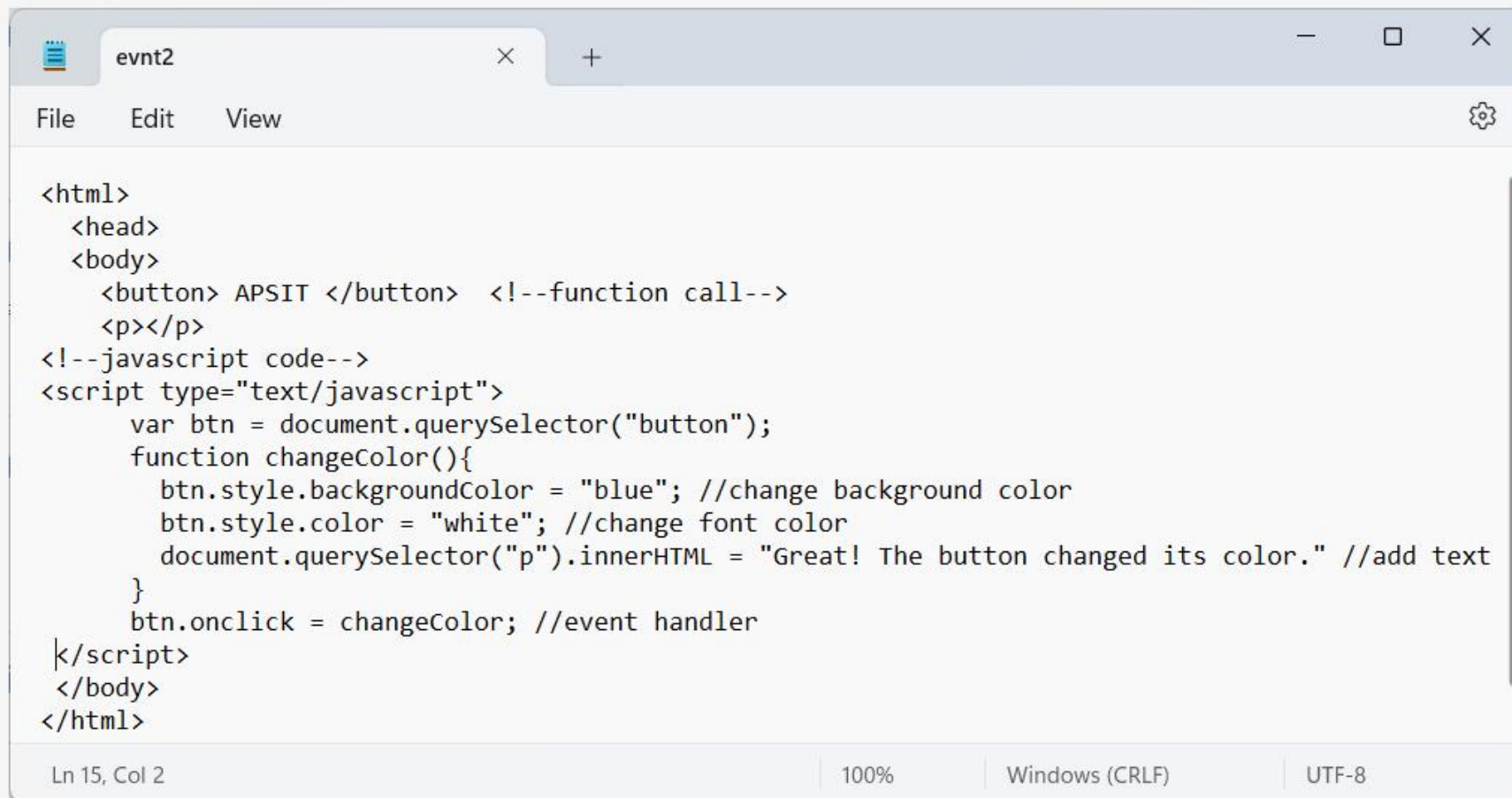
Method 2: Register event handlers in the script

- Best practice
- All the major browsers support this approach.
- The drawback is that for any event, you can attach only one function.
- As a result, if a page uses **more than** one script, and both scripts **respond to the same event**, then **one or both of the scripts may not work** as intended.
- Syntax:

```
element.onevent = functionName;
```

Event Handler Registration

Method 2: Register event handlers in the script



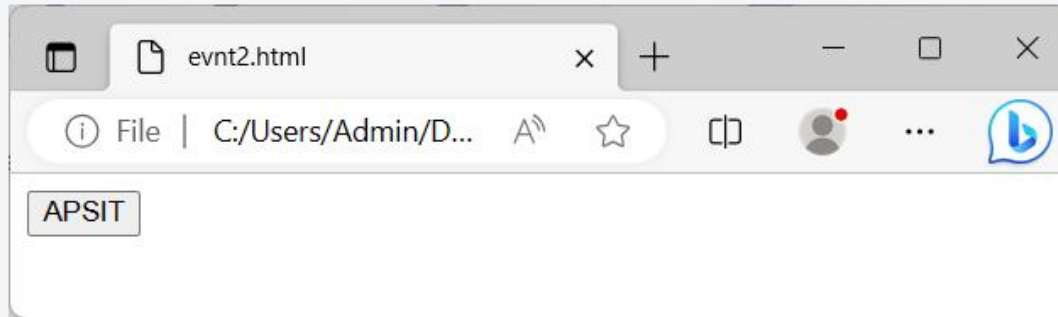
The screenshot shows a web browser window titled 'evt2'. The address bar is empty. The page content includes a button labeled 'APSIT' and a paragraph. A JavaScript script is embedded in the page, which registers an event handler for the button's click event. The script defines a function 'changeColor' that changes the button's background color to blue, its text color to white, and updates the paragraph's content to 'Great! The button changed its color.'.

```
<html>
  <head>
  <body>
    <button> APSIT </button> <!--function call-->
    <p></p>
  <!--javascript code-->
  <script type="text/javascript">
    var btn = document.querySelector("button");
    function changeColor(){
      btn.style.backgroundColor = "blue"; //change background color
      btn.style.color = "white"; //change font color
      document.querySelector("p").innerHTML = "Great! The button changed its color." //add text
    }
    btn.onclick = changeColor; //event handler
  </script>
  </body>
</html>
```

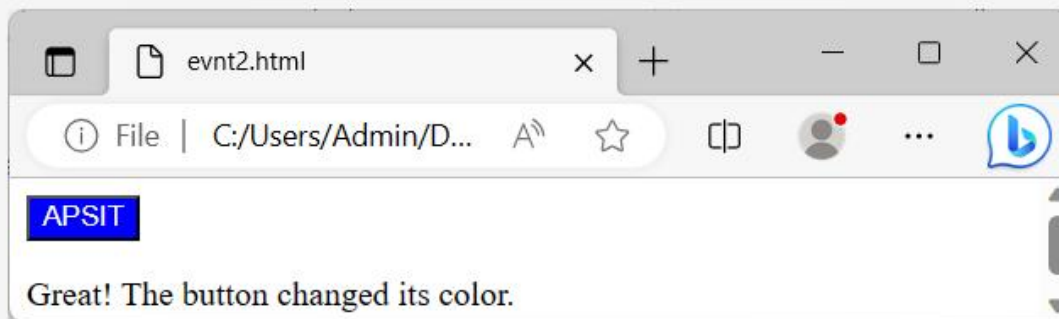
Ln 15, Col 2 | 100% | Windows (CRLF) | UTF-8

Event Handler Registration

Method 2: Before Click



After Click



Advanced Event Handler Registration

- The script-based event handler registration, as described so far, is considered the “**traditional**” approach because it is part of the Netscape 3 standard
- *PROBLEM:*
 - With the traditional approach, only one function can be registered for any one particular event
- **Advanced event handler registration** allows many event handlers for the same event on the same element
 - W3C event handler registration

W3C Event Handler Registration

- Supported by Mozilla and Safari, but not by Microsoft IE8 or earlier version
- These differences in browser support are resolved by using jQuery.

myscripts.js

```
function initForm() {  
    var theForm = document.getElementById("sandwichform");  
    theForm.onsubmit=addSandwich;  
}  
  
window.addEventListener("load",initForm,false);
```

Use the “**addEventListener**”
method of the object that gets
the event handler

Three arguments:

1. The event name as a string
2. The event handler function
3. A boolean

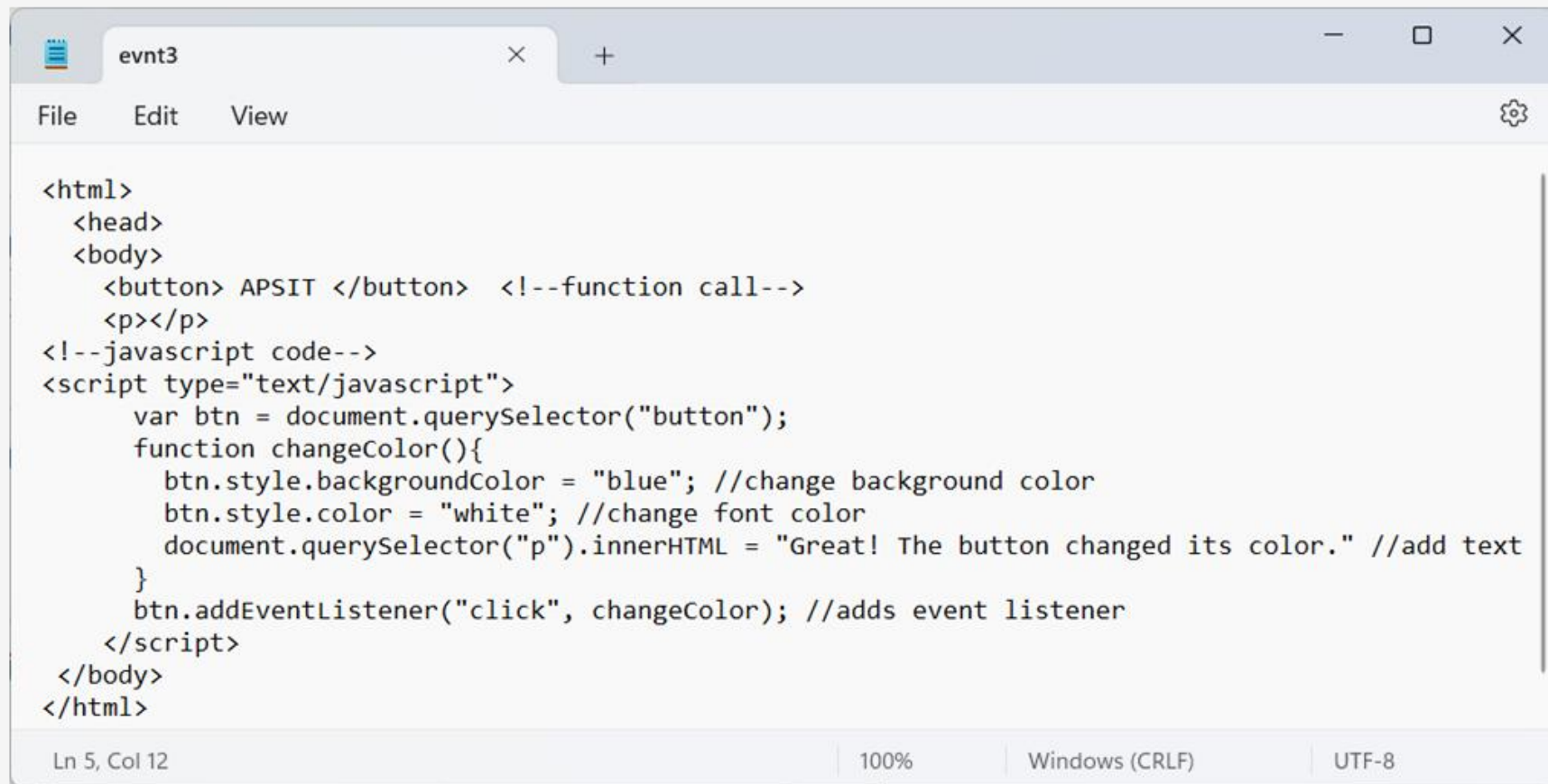
false = the event is captured (the usual choice)

true = the event bubbles up

W3C Event Handler Registration

- You can add an event listener to your event with the following syntax:

```
element.addEventListener("event", functionName [, Boolean]);
```

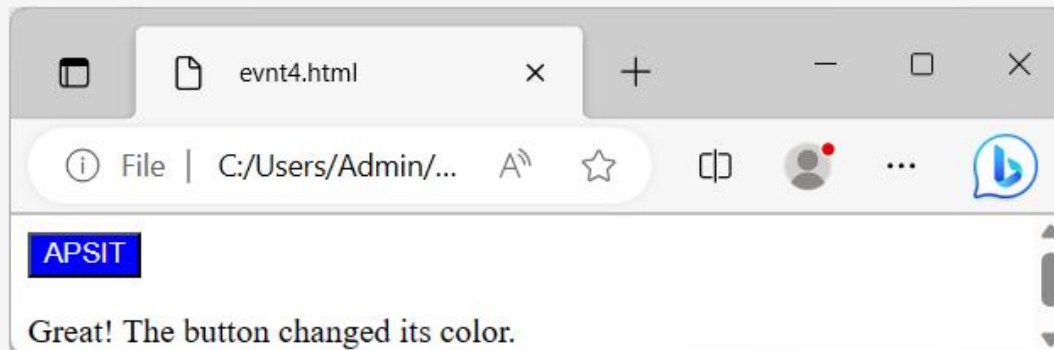


The screenshot shows a web browser window titled 'evnt3'. The browser's address bar is empty. The page content includes a button labeled 'APSIT' and a paragraph. A JavaScript script is embedded in the page, which uses the `addEventListener` method to register a click event listener on the button. The script defines a `changeColor` function that changes the button's background color to blue, the font color to white, and updates the paragraph's text to 'Great! The button changed its color.'

```
<html>
  <head>
  <body>
    <button> APSIT </button> <!--function call-->
    <p></p>
  <!--javascript code-->
  <script type="text/javascript">
    var btn = document.querySelector("button");
    function changeColor(){
      btn.style.backgroundColor = "blue"; //change background color
      btn.style.color = "white"; //change font color
      document.querySelector("p").innerHTML = "Great! The button changed its color." //add text
    }
    btn.addEventListener("click", changeColor); //adds event listener
  </script>
  </body>
</html>
```

Ln 5, Col 12 | 100% | Windows (CRLF) | UTF-8

W3C Event Handler Registration



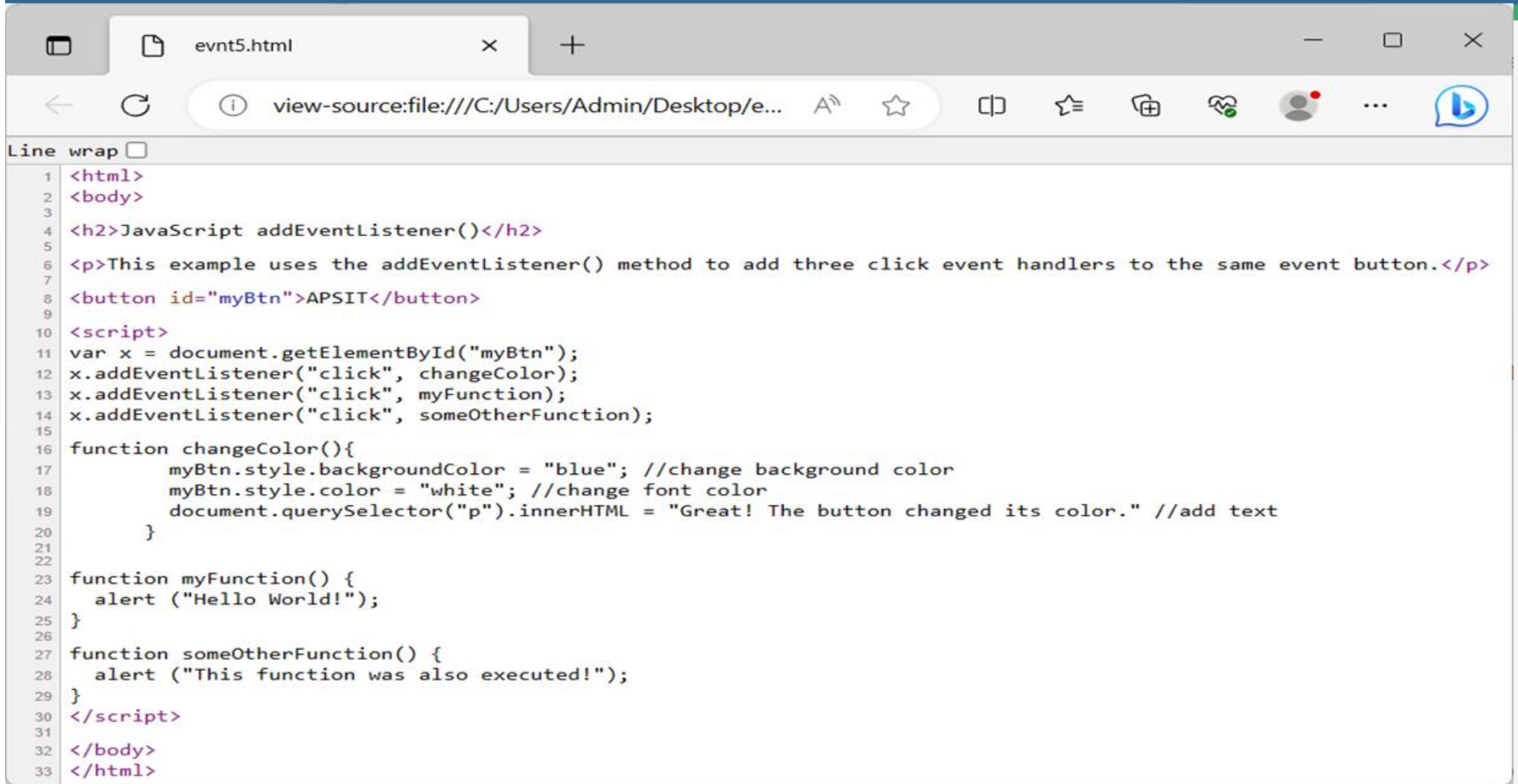
- You can remove the event listener by using the following syntax:

```
element.removeEventListener("event", functionName [, Boolean]);
```

- To remove the event listener from your event, use the following code:

```
btn.removeEventListener("click, changeColor");
```

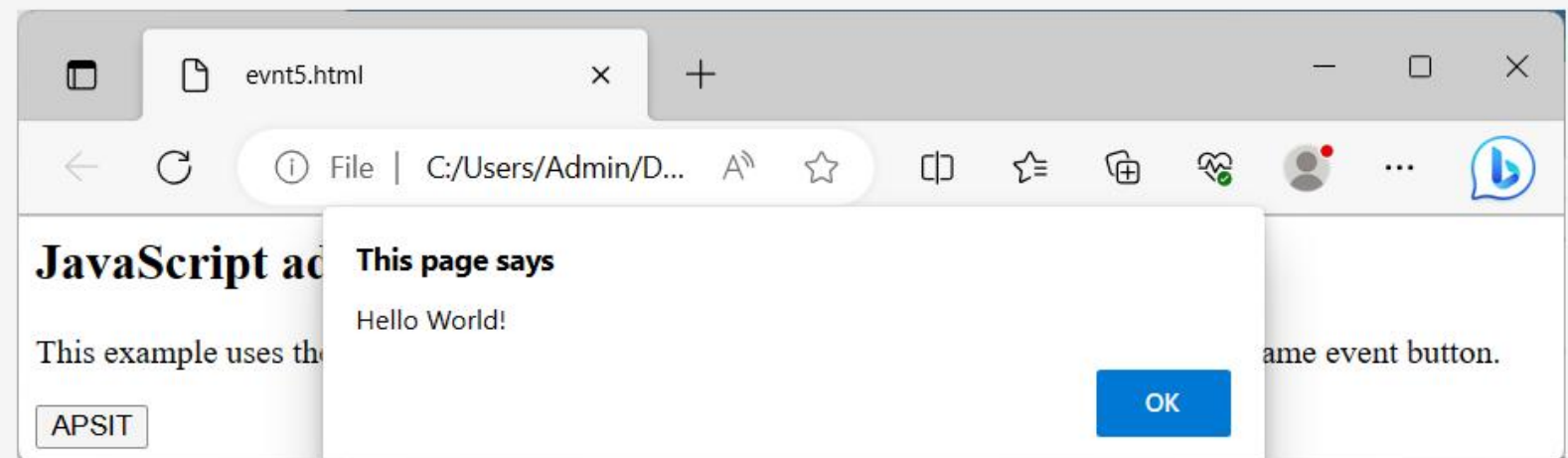
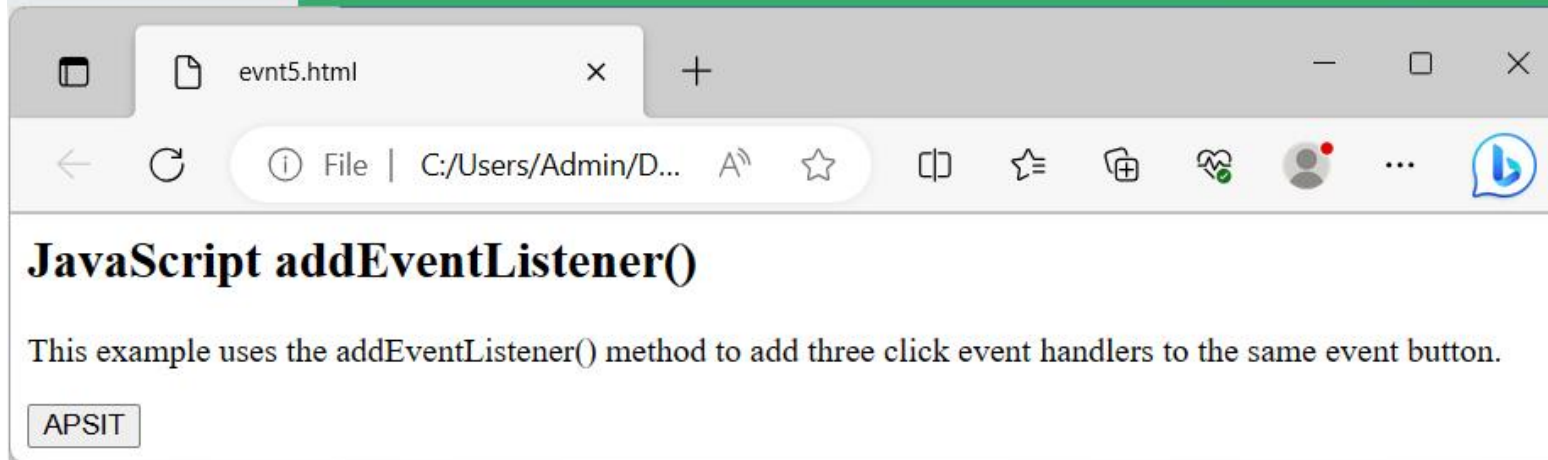
W3C Event Handler 3 Event Handler on same event



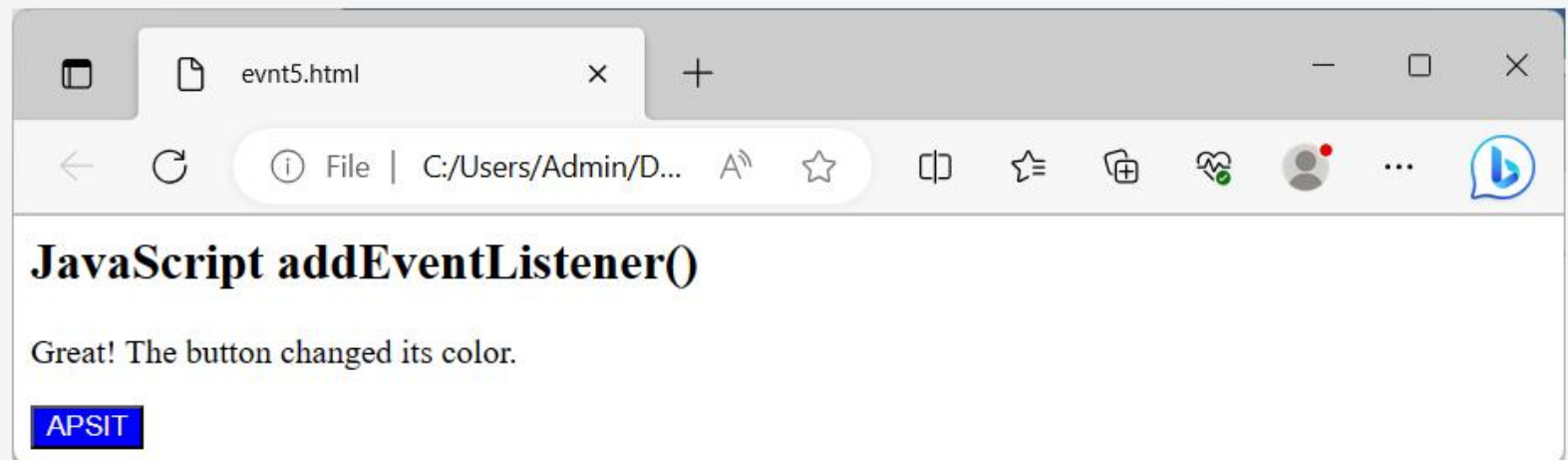
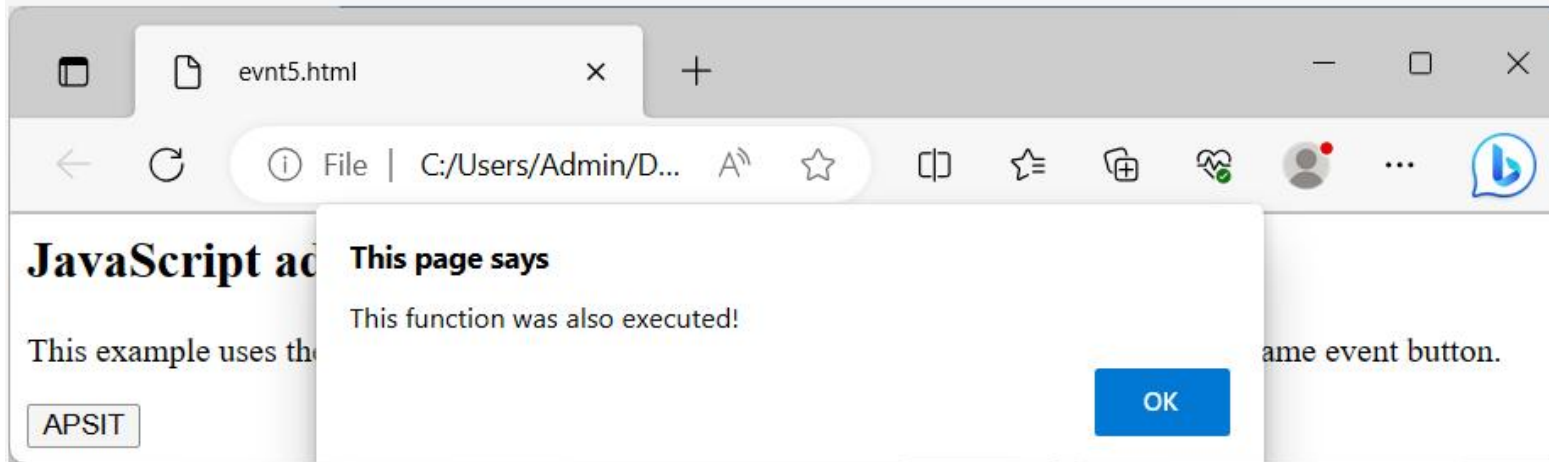
The screenshot shows a web browser window with a single tab titled 'evnt5.html'. The address bar displays 'view-source:file:///C:/Users/Admin/Desktop/e...'. The browser's developer tools are open, showing the source code of the file. The code is an HTML document with a single button and three JavaScript event listeners attached to its 'click' event. The first listener, 'changeColor', changes the button's background and font color and updates the page text. The second, 'myFunction', shows an alert. The third, 'someOtherFunction', also shows an alert. The code is as follows:

```
1 <html>
2 <body>
3
4 <h2>JavaScript addEventListener()</h2>
5
6 <p>This example uses the addEventListener() method to add three click event handlers to the same event button.</p>
7
8 <button id="myBtn">APSIT</button>
9
10 <script>
11 var x = document.getElementById("myBtn");
12 x.addEventListener("click", changeColor);
13 x.addEventListener("click", myFunction);
14 x.addEventListener("click", someOtherFunction);
15
16 function changeColor(){
17     myBtn.style.backgroundColor = "blue"; //change background color
18     myBtn.style.color = "white"; //change font color
19     document.querySelector("p").innerHTML = "Great! The button changed its color." //add text
20 }
21
22
23 function myFunction() {
24     alert ("Hello World!");
25 }
26
27 function someOtherFunction() {
28     alert ("This function was also executed!");
29 }
30 </script>
31
32 </body>
33 </html>
```

W3C Event Handler 3 Event Handler on same event



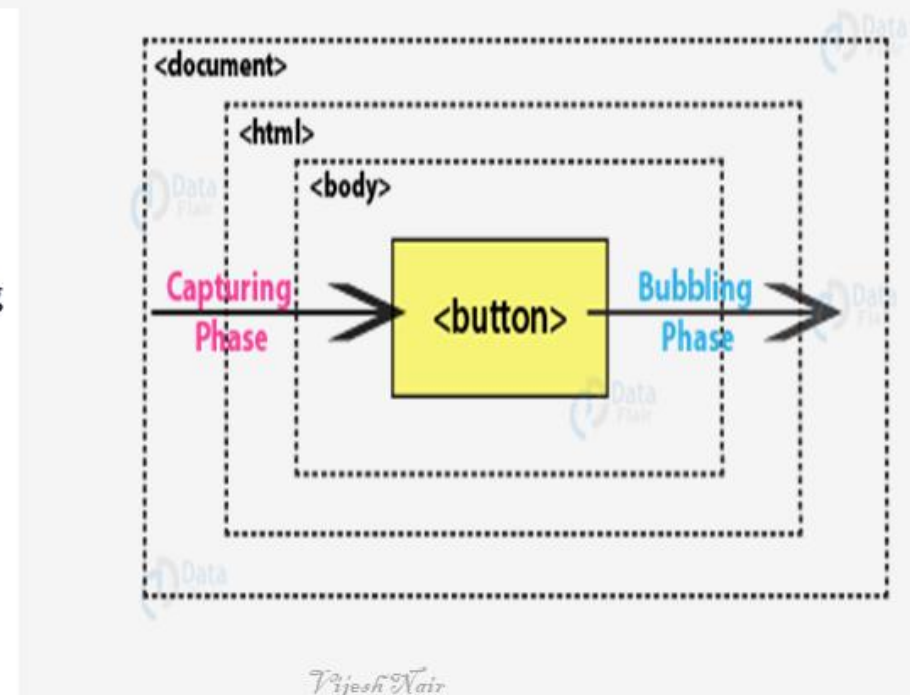
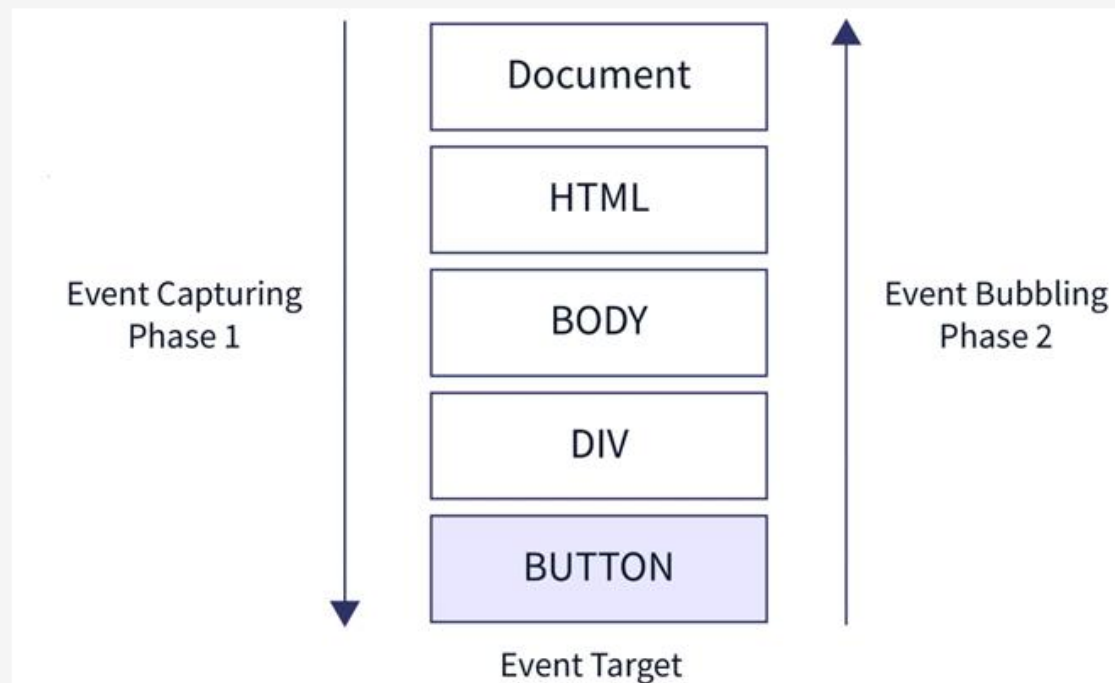
W3C Event Handler 3 Event Handler on same event



JavaScript Event Flow

The lifecycle of a JavaScript event contains three different phases of events:

- **Capturing Phase:** In the capture phase, generally known as the trickling phase, the event "trickles down" to the element that caused the event.
- **Target Phase:** It starts with the element and handler at the top level and works its way down to the element. When the event arrives at the target, the capture phase is over.
- **Bubbling Phase:** The event is "bubbled" up to the DOM tree during the bubble phase.



Thank You!