Understanding OF THE CASE - EDP

1. index.html

This is the main HTML page.

- It shows:
    - A heading (Event-Driven Programming Demo)
    - A text box to type something
    - A "Submit" button
    - An empty paragraph to show output

2. style.css

This makes the page look better:

- Centers everything on the screen
- Adds space and padding to input and buttons
- Makes the output text bold

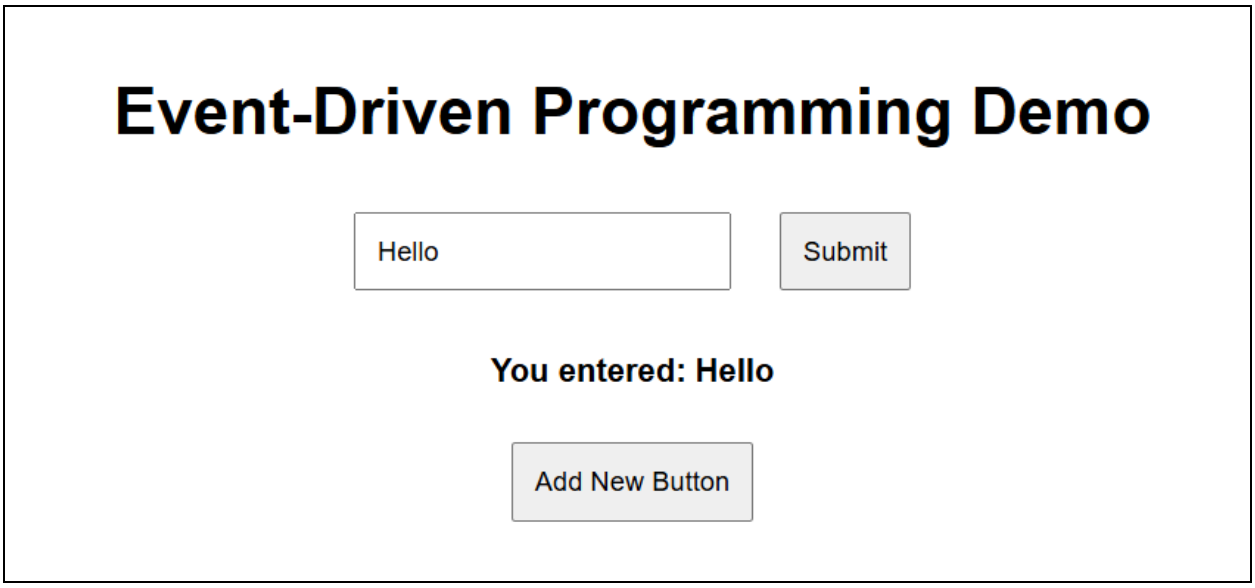
3. script-step2.js

This is the JavaScript section — it adds **event-driven behavior**.

```
document.getElementById('submitBtn').addEventListener('click', function() {
    let input = document.getElementById('inputField').value;
    document.getElementById('output').innerText = "You entered: " + input;
});
```

It handles **one event** — clicking the "Submit" button.

When clicked, it reads the input text and displays it on screen.



4.script-step3.js

```
document.getElementById('submitBtn').addEventListener('click', function() {
    let input = document.getElementById('inputField').value;
    document.getElementById('output').innerText = "You entered: " + input;
});
```

Event type: click
Event target: submitBtn (the "Submit" button)
When clicked: It reads the text from the input box and shows the message.

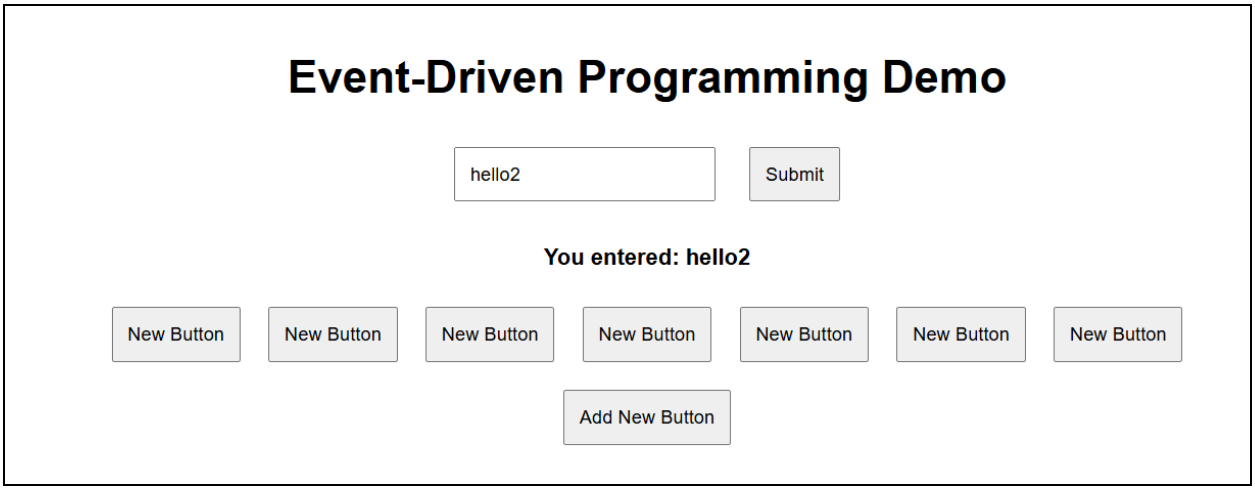
```
// Event delegation for dynamically added buttons
document.getElementById('addBtn').addEventListener('click', function() {
    let newButton = document.createElement('button');
    newButton.innerText = "New Button";
    document.getElementById('buttonContainer').appendChild(newButton);
});
```

Event type: click
Event target: addBtn (the "Add New Button")
When clicked: It creates a new button labeled “New Button” and adds it inside the button container.

```
document.getElementById('buttonContainer').addEventListener('click', function(e) {
    if (e.target.tagName === 'BUTTON') {
        alert('You clicked ' + e.target.innerText);
    }
});
```

Event type: click

Event target: buttonContainer



Differences between 2 and 3

Step 2 -

It only one event is used which is – **click on the Submit Button**. When clicked, it reads the text from the input field and displays it in the output area.

It handles only the static elements and shows the basic concepts of events-driven programming.

Step 3 –

A new **Add New Button** allows the user to create buttons dynamically. Using event delegation, a single event listener detects click on all new buttons and shows an alert when any of them is clicked.

It demonstrates handling events for dynamic elements.

Features / Concepts	**Step 2**	**Step 3**
Number of event listeners	1	3
Event type used	Click **(submitBtn)**	Click **(submitBtn, addBtn, buttonContainer)**
Can add new buttons	It cannot add new buttons	It can add new buttons
Uses event delegation	It doesn't use event delegation	It use event delegation
Handle dynamically created elements	It doesn't handles dynamically created elements	It handles dynamically created elements
Main focus	Basic event handling	Event delegation and dynamic events

Java Version (Swing GUI)

Index.html

```html
<!DOCTYPE html>
<html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Code Comparison</title>
    </head>
    <h2> Click Count: <span id="count">0</span></h2>
    <button id="clickBtn">Click me!</button>

    <script>
        let count = 0;
        document.getElementById("clickBtn").addEventListener("click", function() {
            count++;
            document.getElementById("count").innerText = count;
        });
        </script>
        </body>
    </style>
</html>
```

Swing.java

```java
import javax.swing.*;
import java.awt.event.*;

public class Swing {
    Run | Debug
    public static void main(String[] args) {
        JFrame frame = new JFrame(title:"Click Counter");
        JButton button = new JButton(text:"Click me!");
        JLabel label = new JLabel(text:"Click Counter: 0");

        button.setBounds(x:100, y:100, width:120, height:30);
        label.setBounds(x:100, y:50, width:120, height:30);

        frame.add(button);
        frame.add(label);
        frame.setSize(width:300, height:200);
        frame.setLayout(manager:null);
        frame.setVisible(b:true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        final int[] count = {0};
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                count[0]++;
                label.setText("Click Counter: " + count[0]);
            }
        });
    }
}
```

Output: