**Event-Driven Programming (EDP) Lab Answers**

**Step 2 – Basic Event Handling**

**What is Event-Driven Programming (EDP)?**  
Event-Driven Programming means the program waits for events (like clicks, typing, or mouse moves) and reacts when they happen using event handlers.

**What HTML elements are used in this step? Write their purpose.**

* <input id="inputField"> → User types something.
* <button id="submitBtn"> → Triggers the event when clicked.
* <p id="output"> → Displays the result.

**What event does the JavaScript wait for?**  
It waits for a click event on the submitBtn.

**What happens when the user clicks the button?**  
The code reads the text from the input field and displays it as “You entered: …” inside the paragraph.

**In simple words, how does this example show event-driven behavior?**  
Nothing happens until you click the button. When you do, the page reacts — that’s event-driven behavior.

**Complete the table below:**

| **Concept** | **Example** |
| --- | --- |
| Event | Button click |
| Event Source | submitBtn |
| Event Listener | addEventListener('click', …) |
| Event Handler | Function that updates output |
| Response | Displays “You entered: …” on the page |

**Step 3 – Dynamic Buttons & Event Delegation**

**What new feature was added in this step?**  
Dynamic button creation and event delegation were added.

**What is the purpose of the Add New Button?**  
Each time it’s clicked, it creates a new button inside the buttonContainer.

**What is event delegation?**  
Event delegation is a technique where one parent element listens for events on all its child elements instead of adding separate listeners to each one.

**How does the code detect which button was clicked?**  
It checks e.target.tagName to see if the clicked element is a BUTTON and uses e.target.innerText to identify which one.

**Why do we need only one event listener for multiple buttons?**  
Because the parent container (buttonContainer) handles all click events for its child buttons using event delegation.

**Complete the table below:**

| **Concept** | **Example** |
| --- | --- |
| Event | Button click (Add Button / New Button) |
| Event Source | addBtn and buttonContainer |
| Event Listener | addEventListener('click', …) |
| Event Handler | Functions that create new buttons or show alert |
| Response | New button added / alert “You clicked New Button” |
| Extra Concept | Event Delegation – handling dynamic elements |

**Difference between Step 2 and Step 3**

| **Feature** | **Step 2** | **Step 3** |
| --- | --- | --- |
| Main Functionality | Single Submit button event | Dynamic buttons + event delegation |
| Event Source | One fixed button | Multiple buttons |
| Event Delegation | Not used | Implemented using parent container |
| Behavior | Displays typed text | Creates buttons and responds to each click |
| Complexity | Basic event handling | Advanced event handling with delegation |

**Conclusion**  
Step 2 introduces the basic concept of event-driven programming — waiting for an event and reacting.  
Step 3 extends it by adding event delegation, where one listener can handle many dynamic elements.