Part II: Designing Your Script

When users interact with elements on a webpage—like clicking a button or typing in a form—JavaScript can respond to those actions. This process is called event handling, and it involves three main steps:

1. Select the element(s) you want the script to respond to.

   In my script, the elements I want the code to respond to are the dynamically created decision-tree buttons. These buttons are generated inside the `createButtons()` function, and each one represents a possible answer the user can choose (for example: “Short,” “Long,” “Curly,” or “Straight”).

2. Specify which event on that element will trigger the response.

   The event that triggers the response is the **click** event. Each button created in the loop receives an event listener using `btn.addEventListener("click", ...)`, so whenever the user clicks one of the buttons, the event activates.

3. Define the code (or function) that will run when the event occurs.

When the click event occurs, the callback function inside the event listener runs. This function sends the text of the clicked button back to the main decision-tree logic. That code then updates the displayed question, removes the old buttons, loads new answer options, or shows the final haircut recommendation depending on the user’s choice.

4. Describe Your Event

Write a short paragraph that describes an event-handling process you plan to implement on your product website. Your description must include:

- The element the user will interact with.
- The event that will trigger your response (e.g., click, mouseover, submit, etc.).
- The code or behavior that should occur when the event happens.

  On my product website, I use event handling to make my haircut decision tree interactive. The user interacts with a set of buttons that are created dynamically for each step of the quiz. Each of these buttons has a **click** event listener attached to it so the script knows when an option has been selected. When a button is clicked, the code runs a callback function that reads the text of the button and uses it to decide what should happen next. This might include updating the question, removing the old buttons, showing new choices, or displaying the final haircut recommendation. This setup makes the decision tree

feel responsive and allows the page to update smoothly based on what the user chooses.

5. Draw an Event Handling Flowchart

Create a flowchart that visually represents your event-handling process. Use the examples above as a reference.

Follow these conventions for your diagram:

Symbol	Meaning	Color
Hexagon	Event (e.g., “click,” “submit”)	Blue
Rectangle	Start/End points (terminal nodes)	Red
Box	Code to run when event occurs	Pink border
Box	Selecting and using elements	Purple
Diamond	Decision nodes (e.g., condition checks)	Yellow

Connect your shapes with labeled arrows showing how the logic flows between steps and branches.