|  |  |
| --- | --- |
| ****Document Ready Event (****DOMContentLoaded****)****  * **Nature**: Asynchronous * **Explanation**:   + The DOMContentLoaded event is fired when the HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.   + The document.addEventListener("DOMContentLoaded", ...) is an asynchronous operation because the browser waits for the document to load before invoking the callback function.   + The code inside this callback will only execute once this event is triggered.  |  | | --- | | let button = document.getElementById("myButton");  button.addEventListener("click", () => {  document.getElementById("output").textContent = "Clicked!";  });   **Nature**: Synchronous   **Explanation**:   * The statements inside this block are executed synchronously, meaning they run sequentially and immediately when the DOMContentLoaded event fires. * The event listener button.addEventListener("click", ...) is set up synchronously — this means the browser registers the click handler at this moment in the event loop, but the function itself does not execute immediately. It simply "registers" the callback to be triggered later. | |
| Click Event on the Button   |  | | --- | | button.addEventListener("click", () => {  document.getElementById("output").textContent = "Clicked!";  }); |  * **Nature**: Asynchronous * **Explanation**:   + The click event is asynchronous because it only fires when the user interacts with the button.   + When the button is clicked, the callback function registered for the click event is added to the JavaScript **event queue**. The browser processes this event and executes the callback after completing all currently executing tasks in the main thread.  ****Execution Flow****  1. The browser starts parsing the HTML. 2. The DOMContentLoaded event listener is registered. 3. Once the DOM is fully loaded (but before images, stylesheets, etc., are fully loaded), the DOMContentLoaded event fires:    * The code inside the DOMContentLoaded listener executes synchronously, registering the click event listener on the button. 4. The script then "waits" for a user interaction (button click). 5. When the user clicks the button:    * The registered callback is added to the event queue.    * Once the main thread is idle, the callback executes and updates the text content of the output element.  ****Synchronous vs Asynchronous Summary****  * **Synchronous**:   + Code inside the DOMContentLoaded callback executes sequentially.   + Event listeners are registered synchronously. * **Asynchronous**:   + The DOMContentLoaded event itself is asynchronous.   + The click event is asynchronous, and its callback is executed only when the event occurs.   This interplay between synchronous and asynchronous behavior is a fundamental aspect of JavaScript's event-driven, non-blocking nature. |
|  |