# Bug 1: Event handler overwriting

|  |  |  |
| --- | --- | --- |
| **CODE SNIPPET DESCRIPTION** | | |
| WHAT THE CODE SNIPPET SHOULD BE DOING? | | WHAT IS THE CODE SNIPPET DOING INCORRECTLY? |
| Both functions - initializeGame() and setupEventListeners() – should be called in response to the load event of the window object. | | The cards are not displayed because multiple window.onload assignments overwrite each other. The cards aren't initialized since initializeGame() is skipped and overwritten by setupEventListeners() |
| **SPECIFIC TEXTBOOK SECTION EXPLAINING THE BUG CORRECTION** | | |
| PAGE NUMBERS AND SECTION TITLES | EXPLAIN HOW THE TEXTBOOK INFORMATION HELPED YOU UNDERSTAND WHY THIS BUG WAS A PROBLEM. | |
| Page Number:  43 | “…limitation of setting an event handler as object property is that only one function can handle an event at a time.  In the following code, the second event handler supersedes the first so that only the second function is run in response to the load event of the window object.    If your application requires multiple functions to be assigned to the same event, you can use an event listener.” | |
| Section Title:  2-2b Events as Object Properties |
| Page Number:  43 | Unlike the event handler approach, event listeners can attach multiple functions to the same event.  In the following code, both functions will be run in response to the load event of the window object: | |
| Section Title: 2-2c Event Listeners |
| Page Number:  44 | “You can also use anonymous functions with event handlers and event listeners.”      “…Anonymous functions can provide more flexibility in your code.” | |
| Section Title: 2-2d Events and Anonymous Functions |
| **BUGGY & FIXED CODE SIDE BY SIDE** | | |
| BUGGY CODE SNIPPET | | FIXED CODE SNIPPET |
|  | | **Fix1** |
| **Fix2** |
| **FIX EXPLANATION** | | |
| Explain why your solution works | **Fix1**: Use addEventListener to register multiple handlers  **Fix2:** Combine both initializeGame() and setupEventListeners() into a single anonymous functions with either event handler or event listener triggered by the load event of the window object. This ensures both initializeGame() and setupEventListeners() are executed after the page loads, cards are properly created and displayed (initializeGame()) and event listeners attached properly (setupEventListeners()). | |
| Describe which event handling concepts you applied | The limitation of setting an event handler as object property is that only one function can handle an event at a time.  Unlike the event handler approach, event listeners can attach multiple functions to the same event.  Using anonymous functions with event handlers and event listeners can provide more code flexibility. | |
| Justify your choice of event handler vs. event listener | We are dealing with multiple load events.  Since only one function can handle an event at a time, using an event listener would be a way to assign multiple functions to the same event. | |

# Bug 2: Incorrect event listener removal

|  |  |  |
| --- | --- | --- |
| **CODE SNIPPET DESCRIPTION** | | |
| WHAT THE CODE SNIPPET SHOULD BE DOING? | | WHAT IS THE CODE SNIPPET DOING INCORRECTLY? |
| Should run the function handling the click logic only once per interaction. | | Event listeners are added multiple times for the same event, causing duplicate execution. |
| **SPECIFIC TEXTBOOK SECTION EXPLAINING THE BUG CORRECTION** | | |
| PAGE NUMBERS AND SECTION TITLES | EXPLAIN HOW THE TEXTBOOK INFORMATION HELPED YOU UNDERSTAND WHY THIS BUG WAS A PROBLEM. | |
| Page Number:  43 | JavaScript manages that event through an [**event model**](javascript://) that describes how objects and events interact within the web page and web browser.  Under JavaScript’s event model, an event like click is first tracked in the [**capture phase**](javascript://), moving down the object hierarchy from the most general object (the browser window) down to the specific (the image itself).  The capture phase is followed by the [**bubbling phase**](javascript://) as the event moves back up the object hierarchy ending with the browser window.  Thus, the event listener is always listening for the event as it goes down the object hierarchy (being captured) or goes up (being bubbled).  To attach an event listener to an object, apply the following method:    where *object* is the object in which to listen for the event, *function* is the name of the function that is run in response, and ***capture*** is an optional value equal to true | |
| Section Title: 2-2c Event Listeners |
| **BUGGY & FIXED CODE SIDE BY SIDE** | | |
| BUGGY CODE SNIPPET | | FIXED CODE SNIPPET |
|  | |  |
| **FIX EXPLANATION** | | |
| Explain why your solution works | Ensure only one click event listener is added per card.  This way a card's click event is triggered only once per interaction, preventing handleCardClick function from running multiple times unnecessarily. | |
| Describe which event handling concepts you applied | The event listener is **always** listening for the event. Event listeners can attach multiple functions to the same event.  To attach an event listener to an object, apply the following method:    where *object* is the object in which to listen for the event, *function* is the name of the function that is run in response, and *capture* is an optional value equal to true. | |
| Justify your choice of event handler vs. event listener | We are dealing with multiple click events. Since only one function can handle an event at a time, using an event listener would be a way to assign multiple functions to the same event. | |

# Bug 3: Event propagation issues

|  |  |  |
| --- | --- | --- |
| **CODE SNIPPET DESCRIPTION** | | |
| WHAT THE CODE SNIPPET SHOULD BE DOING? | | WHAT IS THE CODE SNIPPET DOING INCORRECTLY? |
| Clicking a card triggers only card's click event without affecting the parent container's click event. | | The onclick handler on the gameContainer is also being triggered when you click on an individual card. |
| **SPECIFIC TEXTBOOK SECTION EXPLAINING THE BUG CORRECTION** | | |
| PAGE NUMBERS AND SECTION TITLES | EXPLAIN HOW THE TEXTBOOK INFORMATION HELPED YOU UNDERSTAND WHY THIS BUG WAS A PROBLEM. | |
| Page Number:  43 | Under JavaScript’s event model, an event like click is first tracked in the capture phase, moving down the object hierarchy from the most general object (the browser window) down to the specific (the image itself).  The capture phase is followed by the bubbling phase as the event moves back up the object hierarchy ending with the browser window.  Thus, the event listener is always listening for the event as it goes down the object hierarchy (being captured) or goes up (being bubbled). | |
| Section Title: 2-2c Event Listeners |
| **BUGGY & FIXED CODE SIDE BY SIDE** | | |
| BUGGY CODE SNIPPET | | FIXED CODE SNIPPET |
|  | |  |
| **FIX EXPLANATION** | | |
| Explain why your solution works | This event propagation issue happens because events in JavaScript bubble up from the child element (the clicked card) to its parent element (gameContainer) unless explicitly stopped.  This means that clicking a card triggers both the card's click event and the container's click event.  To stop event propagation from the card's click event we can update the handleCardClick function to include event.stopPropagation(). | |
| Describe which event handling concepts you applied | Understanding the difference between the capturing and bubbling | |
| Justify your choice of event handler vs. event listener | We are dealing with multiple click events. Since only one function can handle an event at a time, using an event listener would be a way to assign multiple functions to the same event. | |

# Bug 4: Incorrect event handling logic

|  |  |  |
| --- | --- | --- |
| **CODE SNIPPET DESCRIPTION** | | |
| WHAT THE CODE SNIPPET SHOULD BE DOING? | | WHAT IS THE CODE SNIPPET DOING INCORRECTLY? |
| Code should be checking if card is already flipped or matched. | | Cards can be clicked repeatedly even when already flipped or matched. |
| **SPECIFIC TEXTBOOK SECTION EXPLAINING THE BUG CORRECTION** | | |
| PAGE NUMBERS AND SECTION TITLES | EXPLAIN HOW THE TEXTBOOK INFORMATION HELPED YOU UNDERSTAND WHY THIS BUG WAS A PROBLEM. | |
| Page Number:  53 | A comparison operator, or relational operator, is used to compare two operands. In this particular case we need to see if we have the two flipped cards. The consecutive logic has to depend on the Boolean result of this check | |
| Section Title:  Comparison Operators |
| Page Number:  54 | Logical operators are often used within conditional and looping statements such as the if, for, and while statements. In our case we do not make a proper comparison in place, so we have to create a game logic. | |
| Section Title:  Logical Operators |
| **BUGGY & FIXED CODE SIDE BY SIDE** | | |
| BUGGY CODE SNIPPET | | FIXED CODE SNIPPET |
|  | |  |
| **FIX EXPLANATION** | | |
| Explain why your solution works | We need to check for whether a card is already flipped or matched before attempting to flip it. | |

# Bug 5: Incorrect matching logic

|  |  |  |
| --- | --- | --- |
| **CODE SNIPPET DESCRIPTION** | | |
| WHAT THE CODE SNIPPET SHOULD BE DOING? | | WHAT IS THE CODE SNIPPET DOING INCORRECTLY? |
| If two flipped cards do not match, they should be flipped back (hidden) after a brief delay. | | Missing else clause to handle non-matching cards.  There’s no handling for unmatched cards; they remain flipped indefinitely. |
| **SPECIFIC TEXTBOOK SECTION EXPLAINING THE BUG CORRECTION** | | |
| PAGE NUMBERS AND SECTION TITLES | EXPLAIN HOW THE TEXTBOOK INFORMATION HELPED YOU UNDERSTAND WHY THIS BUG WAS A PROBLEM. | |
| Page Number:  53 | A comparison operator, or relational operator, is used to compare two operands. In this case we need to see if we have the two matching cards. The consecutive logic has to depend on the Boolean result of this check  Logical operators are often used within conditional and looping statements such as the if, for, and while statements. In our case we have to create a game logic to handle a situation where the two cards do not match(they should be flipped back (hidden) after a brief delay). | |
| Section Title:  Comparison Operators |
| Page Number:  54 |
| Logical Operators |
| **BUGGY & FIXED CODE SIDE BY SIDE** | | |
| BUGGY CODE SNIPPET | | FIXED CODE SNIPPET |
|  | |  |
| **FIX EXPLANATION** | | |
| Explain why your solution works | We need to ensure that if we have two non-matching cards, they are flipped back. | |

# Bug 6: Missing win condition check

|  |  |  |
| --- | --- | --- |
| **CODE SNIPPET DESCRIPTION** | | |
| WHAT THE CODE SNIPPET SHOULD BE DOING? | | WHAT IS THE CODE SNIPPET DOING INCORRECTLY? |
| Game hould announce win and prevent further clicks | |  |
| **SPECIFIC TEXTBOOK SECTION EXPLAINING THE BUG CORRECTION** | | |
| PAGE NUMBERS AND SECTION TITLES | EXPLAIN HOW THE TEXTBOOK INFORMATION HELPED YOU UNDERSTAND WHY THIS BUG WAS A PROBLEM. | |
| Page Number:  41 | checkWin() function is never called, so even if the logic was present, the function would not execute.  By running it within the handleCardClick function, we ensure to run the win check after each card match situation. | |
| Section Title:  Calling a Function |
| **BUGGY & FIXED CODE SIDE BY SIDE** | | |
| BUGGY CODE SNIPPET | | FIXED CODE SNIPPET |
|  | |  |
| **FIX EXPLANATION** | | |
| Explain why your solution works | We need to:  - make sure the checkWin function is called after each card match  - track the number of matched cards  - correctly check if all cards have been matched - if the number of matched cards (.matched class) equals the total number of cards (cardSymbols.length).  - reset the game | |