Code Analysis Report: No-Code Form Builder

Case Study Report Subject: No Code Form Builder. **Author:** Biswajeet Rout, Neel Singh, Saish Bhujbal, Manthan Shirke **Group:** Team 2 —

Index

---

1. Introduction

This report provides a comprehensive code analysis of a "No-Code Form Builder." The project is designed to allow users to construct forms visually without writing code. The analysis focuses on the application's architecture, specifically its clean three-pane layout, drag-and-drop capabilities, and the recursive rendering logic used to handle nested elements.

2. Problem Statement & Requirements

Abstract: The objective was to create a visual interface that requires no coding from the end-user. The system needed to support complex features such as nesting elements inside containers and generating real-time HTML exports.

Project Requirements Check: The following table outlines the core requirements and their implementation status:

Requirement	**Implementation Status**
Drag & Drop elements	Native HTML5 API
Customize properties	Properties panel
Recursive nesting	Container element with recursive render
Live preview	Real-time HTML generation
Export HTML	Textarea with generated code
No coding required	Visual interface only

---

3. Case Study Design

Architecture Analysis The application follows a clean three-pane layout designed for usability:

```
- **Left Sidebar (200px):** Contains draggable form elements, including Text Input, Checkbo
- **Centre Canvas:** The main area where form elements are placed and arranged.
- **Right Properties Panel (220px):** Used for editing element properties and viewing the H
```

UI/UX Design: The design uses simple, functional styling with borders and basic spacing to clearly delineate different sections. It features a minimal visual hierarchy with slight background variations and relies entirely on pure vanilla CSS without external dependencies.

4. Methods & Technology

Technology Stack

```
- **Core Logic:** Pure Vanilla JavaScript (~120 lines).
- **Styling:** Native CSS (No external frameworks).
- **APIs:** Native HTML5 Drag-and-Drop API.
```

Key Concepts Demonstrated The implementation demonstrates three advanced technical concepts: 1. **DOM Traversal:** Direct manipulation of `canvas.innerHTML`, and event delegation on dynamically created elements. 2. **Recursion:** Used for handling arbitrarily deep nesting structures, where `renderElement()` calls itself for child elements. 3. **Object Storage:** Separation of data and presentation, where each element is stored as a plain JavaScript object within an array-based tree structure.

---

5. Implementation Details

1. Data Structure The application uses a simple object structure for form elements, allowing for unique identification and nesting.

JavaScript

```
{
  id: Date.now() + Math.random(), // Unique IDs generated using timestamp [cite: 20, 26]
  type: "text",
  label: "Text Input",
  placeholder: "",
  children: [] // Children array enables nesting [cite: 24, 27]
}
```

1. Drag & Drop System The system uses the native HTML5 API with clean event handling.

JavaScript

```
// Simple drag setup with native HTML5 API
item.addEventListener('dragstart', e => {
    e.dataTransfer.setData('type', item.dataset.type);
});
```

1. Recursive Rendering (Core Feature) Recursion is properly implemented to handle arbitrary depths of nesting within containers.

JavaScript

```
// Recursive function that handles nested structures
function renderElement(el) {
    // ... renders current element
    // RECURSION for nested elements
    if (el.type === "container") {
        el.children.forEach(child =>
            wrapper.appendChild(renderElement(child))
        );
    }
    return wrapper;
}
```

1. Live Preview & HTML Export The HTML updates automatically as the form changes, using a similar recursive pattern for generation.

JavaScript

```
// Real-time HTML generation
function exportHTML() {
    // output value
    generateHTML(formTree);
}
```

---

6. Results & Conclusion

Performance Analysis

```
- **Strengths:** The codebase is minimal (~120 lines) and has no dependencies. The recursion
- **Limitations:** The system is limited to 4 basic elements and lacks element deletion func
```

Suggested Improvements Future iterations should include: - - **Enhanced Elements:** Add dropdowns, radio buttons, and email fields. - **Functionality:** Add delete buttons, undo/redo history, and validation rules. - **UI Polish:** Improved visual feedback for drop zones.

Conclusion: This implementation successfully demonstrates key concepts like DOM traversal, recursion, and object storage. The code is clean, well-structured, and provides a functional foundation that focuses on core requirements without unnecessary complexity.

7. References

- Native HTML5 Drag and Drop API Documentation.
- Code Analysis Report: No-Code Form Builder.