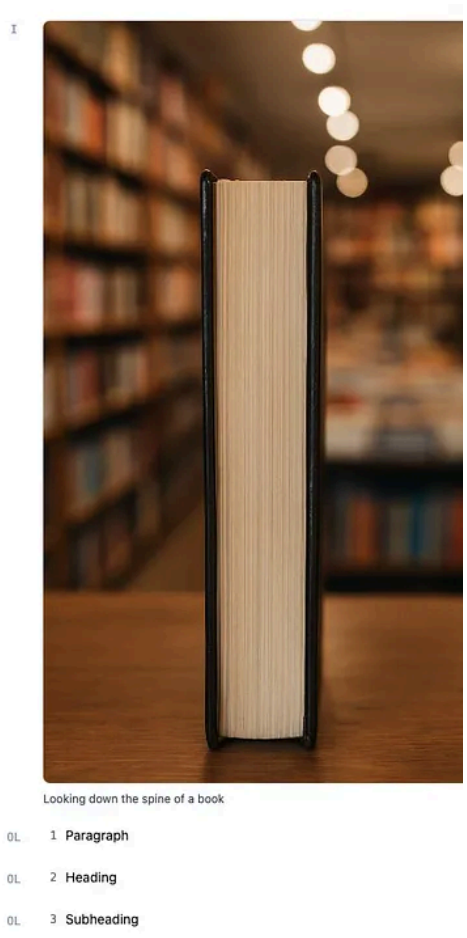

Introducing Notebook-js: A Simple, React-Friendly Block-Based Text Editor

3 min read · Jun 22, 2025

Ashwin Abraham

Listen Share More

As a developer, finding the right text editor for React projects can be a journey. Over time, I've experimented with popular solutions like Editor.js and Draft.js. While both are powerful, I ran into roadblocks that led me to build my own: notebook-js.

The Problem with Existing Solutions

Editor.js: Modern But Not React-Friendly

Editor.js is a popular block-based editor that's visually appealing and extensible. However, it's built with vanilla JavaScript in mind, and integrating it seamlessly with React can be surprisingly tricky. Managing lifecycle events, refs, and state synchronization often felt hacky and fragile. I craved a solution that felt native to the React ecosystem — with predictable data flow and simple component patterns.

Draft.js: Powerful Yet Overly Complex

Draft.js, created by Facebook, is robust and highly customizable. But with power comes complexity. Every time I set up Draft.js in a new project, I found myself writing boilerplate code, wrangling with editor states, and navigating a steep learning curve just to achieve basic block-based editing. For most of my use-cases, this felt like a lot.

Introducing notebook-js

notebook-js is a React component for block-based text editing, designed with simplicity and developer experience in mind.

My goals were:

- Easy integration into any React project
- Minimal setup
- Intuitive block editing — create, move, and edit text blocks naturally
- Modern, extensible architecture — customize if you want, but sensible defaults work out-of-the-box

Why notebook-js?

- React-first: Built from the ground up for React, so you won't battle with refs or hack around lifecycle methods.
- Simple API: Add the editor as a component and start editing. No elaborate configuration required.
- Customizable: Supports custom block types and styles, but doesn't force complexity on you.

Getting Started

Install via npm:

```
npm install @tamatashwin/notebook-js
```

Add to your component tree:

```
import react, { useState, useCallback } from "react";
import {
    NotebookJS,

    // Import blocks
    ParagraphBlock,
    HeadingBlock,
    SubheadingBlock,
    SubheadingBlock,
    UnorderedListBlock,
    OrderedListBlock,
    ImageBlock,

    // Import tools
    BoldTool,
    HighlightTool,

    // Import editors
    SetNumberingEditor,
    ImageEditor
} from "@tamatashwin/notebook-js";
import "@tamatashwin/notebook-js/styles.css";

function MyEditor() {
  let [blocks, setBlocks] = useState([]);
    let handleChange = useCallback((newBlocks) => setBlocks(newBlocks), []);

    return (
        <div className="mb-6 rounded-md border border-gray-300">
            <NotebookJS
                onChange={handleChange}
                readOnly={false}
                blocks={[
                    ParagraphBlock,
                    HeadingBlock,
                    SubheadingBlock,
                    UnorderedListBlock,
                    OrderedListBlock,
                    ImageBlock
                ]}
```

```
                tools={[
                    BoldTool,
                    HighlightTool
                ]}
                editors={[
                    SetNumberingEditor,
                    ImageEditor
                ]}
            />
        </div>
    );}
```

That's it! You have a block-based editor up and running in your React app.

Inbuilt Blocks, Tools and Editors

Till now the following block types are built in:

1. Paragraph Block
2. Heading Block
3. Subheading Block
4. Unordered List Block
5. Ordered List Block
6. Image Block

There are also these tools:

1. Bold Tool
2. Italic Tool
3. Underline Tool
4. Superscript Tool
5. Subscript Tool
6. Highlight Tool
7. Link Tool

And finally, the following editors:

1. Edit Props Editor — if you want to change the properties.
2. Set Numbering Editor — for the ordered list blocks
3. Image Editor
4. Canvas Editor
5. Table Editor
6. Latex Editor

Contribute or Try It Out

I'm actively developing notebook-js and welcome feedback, feature requests, or contributions. Check out the GitHub repo for the latest updates, documentation, and examples.

If you've struggled with integrating rich text editors into React, I hope notebook-js can save you time and frustration. Give it a try and let me know what you think!

Thanks for reading. You can find the source code and contribute to notebook-js here and the npm package here

Coding React Reactjs JavaScript

Edit profile

Written by Ashwin Abraham

1 follower · 8 following