Visual Editor i

The **Visual Editor** is a platform that allows developers and content teams to collaboratively manage, edit, and synchronize content across websites and applications in real-time. It provides a user-friendly interface for managing content, enabling teams to preview changes in the visual context of the website's design, reducing friction between developers and non-technical users.

Key Features of Visual Editor

1. **Two-Way Content Syncing**
   The Visual Editor uses the **Content Source Interface (CSI)** to sync content between various **content sources** (CMS, databases, files, etc.) and itself in real-time.
   - Edit content in the Visual Editor, and the changes are synchronized back to the source instantly.
   - Make updates to the content source directly, and changes are reflected in the Visual Editor without delay.
2. **Content Source Support**
   The Visual Editor integrates with a variety of **API-based CMS** platforms and custom content sources through the **Content Source Interface**.
   - **Officially Supported Sources**: Contentful, Sanity, DatoCMS, Git CMS.
   - **Experimental Support**: Strapi, Shopify, Figma.
   - **Bring Your Own Source (BYOS)**: You can integrate non-supported sources like databases or markdown files by building custom modules using the CSI.
3. **Live Content Editing**
   - Teams can edit content in the context of the actual site design and preview changes live before publishing.
   - This eliminates the guessing game of how updates will appear on the frontend.
4. **Developer-Friendly Setup**
   - Developers control the Visual Editor's behavior by defining configurations in the `stackbit.config.js` file.
   - Dependencies and content syncing integrated via the CSI are for development use only, leaving the site's build process lightweight and unaffected.
5. **Collaborative Workflow**
   - Supports linking accounts for team members.
   - Synchronizes changes made by different contributors to avoid conflicts.
6. **Versioning and Migration**
   Switching between content sources or migrating from one source to another (e.g., migrating from one CMS to another) is simplified. The CSI makes it possible to maintain content continuity while gradually transitioning sites.

How the Visual Editor Works

1. Connecting Content Sources

The core of the Visual Editor revolves around connecting to **content sources** using the **Content Source Interface (CSI)**.

- Content sources could be CMS platforms, databases, or file-based systems.
- Developers must configure these sources in the `stackbit.config.js` file by instantiating the respective content module. (E.g., adding Contentful).

Example Setup for Contentful:

```
import { ContentfulContentSource } from "@stackbit/cms-contentful";

export default {
  contentSources: [
    new ContentfulContentSource({
      spaceId: process.env.CONTENTFUL_SPACE_ID,
      environment: process.env.CONTENTFUL_ENVIRONMENT,
      previewToken: process.env.CONTENTFUL_PREVIEW_TOKEN,
      accessToken: process.env.CONTENTFUL_MANAGEMENT_TOKEN,
    }),
  ],
};
```

2. The Content Panel

Once properly connected, the Visual Editor displays the content of the source in its **Content Panel**.

- Models and objects from the content source become visible for editing.
- Content can be previewed and adjusted visually.
- Updates reflect both in the Visual Editor and at the source.

3. CRUD Operations

The Visual Editor enables **CRUD (Create, Read, Update, Delete)** operations to modify content directly:

- Edit existing entries to adjust text, images, or other media.

- Add new entries with the correct schema, and they save to the connected source.
- Delete content from the content source with appropriate syncing.

4. Live Preview

- The Visual Editor provides full visual context for content updates.
- Users can view exactly how their content will appear on the page or application they manage.

Important Components of Visual Editor

1. **Content Source Interface (CSI)**
   The CSI is a bridge between the Visual Editor and the content source. It ensures that the edits done visually are synchronized with the source and enables working with any system capable of supporting API read/write operations.
   - Loads content schema and content data.
   - Manages CRUD operations.
   - Handles asset uploading and syncing (e.g., images, videos).
2. `stackbit.config.js` **File**
   This configuration file defines content sources and other Visual Editor behavior. Developers can add support for multiple sources, switch between experimental tools, and personalize the Visual Editor workflow here.

Supported Use Cases

1. **Integrated CMS Workflows**
   For example, syncing **headless CMS** content (Contentful, Sanity, DatoCMS, etc.) with the editor, letting editorial teams modify their web content directly.
2. **Database Management**
   If your content resides in a remote or local **database**, the Visual Editor can provide a GUI for non-technical users to view and update the content.
3. **Custom Sources and BYOS**
   For unsupported platforms, developers can build custom CSI connectors to manage their source content through the Visual Editor.
4. **Content Migration**
   Seamlessly migrate content between two sources without disrupting live content or requiring significant code restructuring.
5. **Multi-Source Management**
   If your site pulls content from multiple sources, they can all be integrated within the Visual Editor for centralized management.

Example Scenarios

Scenario 1: Connecting a CMS (Contentful)

1. Install the relevant content source NPM module:

```
npm install @stackbit/cms-contentful --save
```

2. Configure Contentful in the `stackbit.config.js` file as shown earlier.
3. Open the Visual Editor, link your account, and begin managing content directly from the editor.

Scenario 2: Integrating a Custom API or Markdown Files

1. Build a custom CSI module for the source.
2. Implement required CSI methods like `loadContent`, `saveContent`, and `deleteContent`.
3. Add the CSI module to the `stackbit.config.js` file.
4. Open the Visual Editor and start editing the custom content.

Advantages of the Visual Editor

Feature	Benefit
Real-Time Sync	Live editing and synchronization between CMS/database and the Visual Editor.
Preview-in-Context	Teams can see exactly how content will look on live sites before changes are deployed.
Support for Various Sources	Offers official modules for CMS platforms and supports BYOS (Bring Your Own Source).
Developer-Friendly	Configurations are stored in development files without affecting live site performance.
Scalable Workflows	Supports multiple sources and migration between them seamlessly for scalability.
Collaborative Editing	Allows teams to link their accounts to manage content collaboratively.

Limitations

1. **Experimental Features**: Certain integrations (e.g., Strapi, Shopify, Figma) are not yet fully developed.
2. **Hands-on Development**: Custom integrations (BYOS) require developers to write and manage custom modules.

In Summary

The **Visual Editor** serves as a powerful bridge between content sources and final delivery, enabling real-time visual editing and management. It minimizes the gap between developers and content teams, resulting in faster workflows, fewer errors, and a more efficient content management process.