Blog

How Component-Based Headless CMS Falls Short

Headless CMS simplified

The headless CMS, or headless Content Management System, broke through around the world in 2015 and its popularity rose as it made it easier, faster, and more flexible to develop webpages. A headless CMS is to act as a content repository, making content accessible with the help of API-drivenI on the back end, and without the need for built-in, front-end code, or a presentation layer.

Adding components to headless CMS allows developers to decide how content will be displayed. When we talk about component-based headless CMS, we're referring to the form-based, reusable content that's coded through interactive JSON frameworks. If you need a one-off landing page but would like to incorporate or transfer most of the text to

But components-based headless CMS does present some pitfalls for the user interface (UI). Namely, the UI isn't as rich, flexible, or autonomous. "It doesn't allow people to use design tokens to make designs from scratch bespoke," critiques Steve Sewell, CEO of Builder IO.

Visitors to your website crave a bespoke experience. An intuitive and customized user experience can exponentially increase the chances of a user's return. According to the Next in Personalization, 2021 Report, "Seventy-one percent of consumers expect companies to deliver personalized interactions. And seventy-six percent get frustrated when this doesn't happen."

Components-driven doesn't allow for flexibility and power to collaborate in the editor, all at once, and in true real-time. This is limiting and presents an incomplete end-to-end experience.

The issue with headless CMS

UIs operate in a tree structure, not a flat structure.

- Headless CMSs can only do flat structures and relational structures that are still flat. They can become complex and challenging to edit.
- Since components are what you install and are made of trees, for example, a hero with additional buttons, a product's grid, etc., you'll need to release, add, and modify components to make a change.
- The process will not be autonomous and changes won't happen in real time.
- The lack of flexibility and optimization capabilities creates a less intuitive and seamless environment for design.

Sub-optimal Developer Experience (DX)

- You have to manually connect all your component inputs via a Graphic User Interface (GUI) and its PITA, resulting in the constant de-synchronization of your code.
- Ultimately, component-based CMS doesn't fully embrace the design tooling needed to create a seamless web flow.
- Developers would need to take apart, examine, and rebuild components to make it compatible with the CMS.
- You'll have to write components with an end-to-end storyblok in mind.
- In addition, the limited configuration doesn't allow for room for customization and fine-tuning roles, permissions, etc., resulting in bulky, awkward components that are constantly out of sync with your code.

The solution

To optimize the experience, you'd have to rely on developers and release cycles to make changes if your component library is limited. In order to optimize the design experience, the best option would be to incorporate a visual CMS, like Builder, which would provide a complete end-to-end experience including a cycle of testing, optimization, integration, analytics, etc.

"Personalization drives performance and better customer outcomes. Companies that grow faster drive 40 percent more of their revenue from personalization than their slower-growing counterparts." Mckinsey & Company 2021

With more tweak ability, you'll have the flexibility and power to create a seamless, more intuitive, and aesthetically pleasing user interface— in one space, in unison, and in real time—without having to rely on developers and release cycles to manipulate the components.

To learn more about headless CMS and Builder IO's solution for creating stunning designs without experience in code, check out this blog on visual CMS.