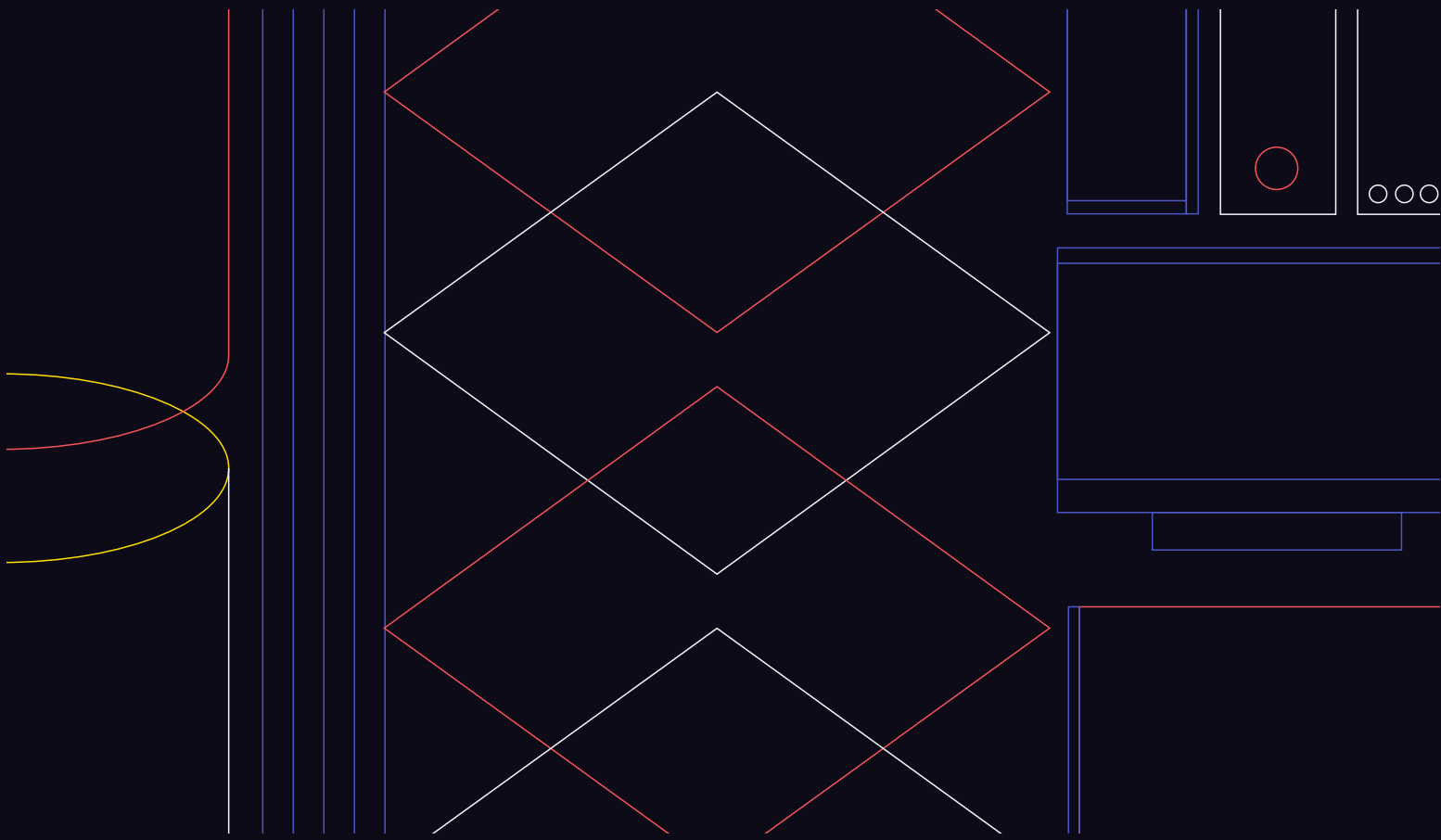
THE ULTIMATE GUIDE TO

Headless CMS in 2022



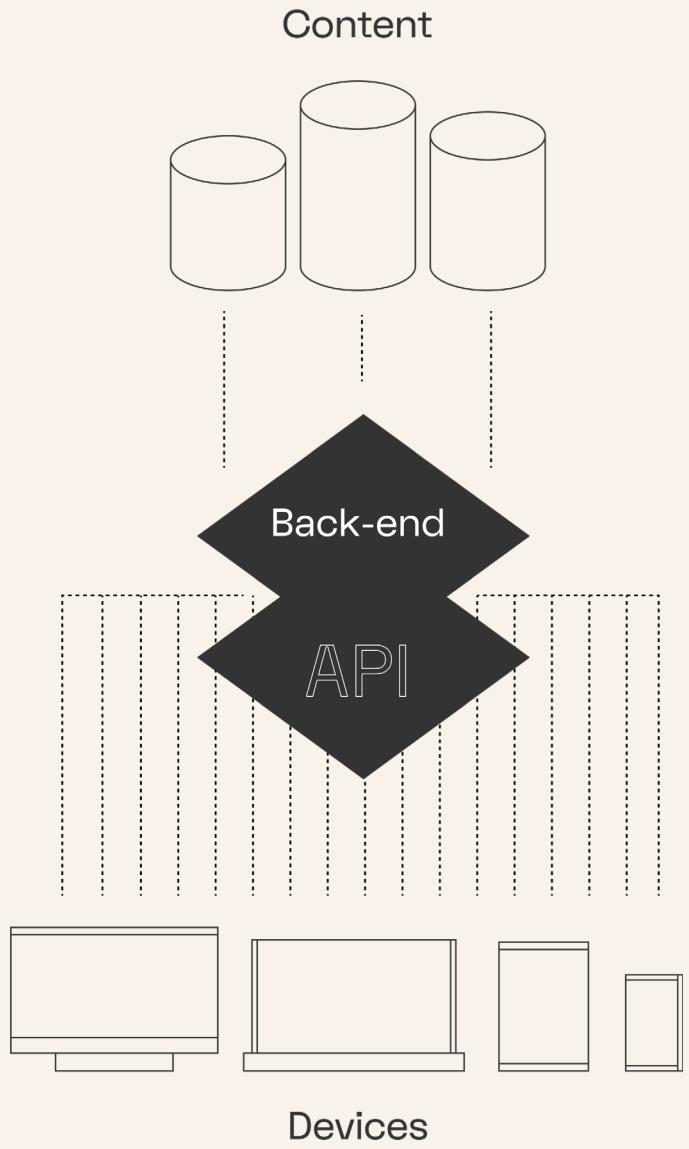
The definitive guidebook to Headless CMS: what a headless CMS is, their pros and cons, examples of real-life use cases, and the key criteria for which headless CMS is right for your organization.

Stackbit

What is a headless CMS?

A Headless CMS is a content management system (like a database for your content). It sits in the backend of your website, mobile app, or other digital property decoupled from the presentation layer or “head”.

This decoupling means your content can be served into whatever head or heads you want. This provides huge productivity benefits for technical and business users alike.



Important Definitions

As more companies adopt headless CMSes, here's your cheat sheet to understand what's going on.

CMS

A content managment system (CMS) helps companies manage and publish content digitally. Used to create, edit, and organize

Traditional CMS / Monolith

A platform that connects the frontend and the backend of a website into one product. Handles the end-to-end website creation process without much flexibility. Examples: WordPress, Acquia, Sitecore, and AEM.

Headless CMS

A backend platform that manages digital content. It is detached from the frontend of a website. This gives companies powerful control over where and how content gets delivered to their users.

Body

Body is used to describe where you content is stored and authored. Aka, the backend.

Head (Presentation Layer)

Head is used to describe where your content ends up — the visual presentation of digital properties like websites or mobile apps.

Code Ownership

With every Traditional CMS or site builder today, you marry your company's digital properties to their ecosystem. With headless CMS, you maintain full ownership of your digital properties.

Tech Stack

The technologies a company uses for a project. Consists of programming languages, frameworks, databases, and applications connected via APIs.

Architecture

The concept of how your tech stack is organized. Important for scalability, flexibility, and ease of use.

Technical Debt

The result of prioritizing short-term technical needs over scalable implementations. Creates situations where tech teams need to rework old projects rather than work on new features.

Reusable Content

Content stored in one location that is published to many different formats or platforms.

Microservices

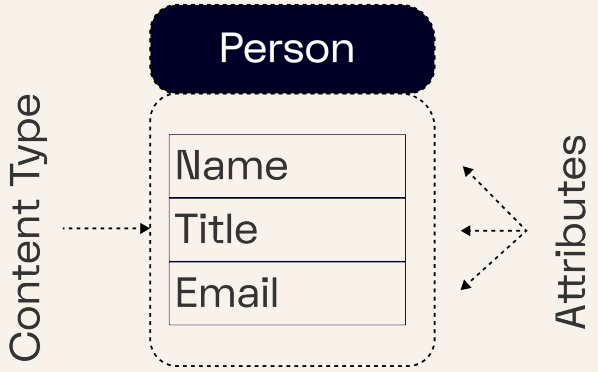
When an application or website is built using many independent, interchangeable, and interconnecting services. Microservices allow cross-functional teams to update services independently, leading to faster deployment and troubleshooting turnaround times

Content Field

The atomic fields that store content in a Headless CMS. The content in each field can be a string, image, array, object, and more.

Content Type

A grouping of content fields in a Headless CMS. Content Types are reusable and usually structured by common purpose.



Content Model

A representation of all the different content types and their interrelationships within a Headless CMS.

Reference Field

A content field that links to another a content field or type.

Structured Content

Content organized in a predictable way outside of the presentation layer so that it's ready to be consumed by any interface.

API

Application Programming Interface." An intermediary interface that allows two applications to talk to each other.

Think of an API like a menu in a restaurant. The menu provides a list of dishes and description that you can order. When you order, the restaurant's kitchen pieces together the ingredients and provides you with a finished dish. The customer is your business. The menu is the API. The kitchen is the service maintaining the API.

Vendor Lock-In

When a business is essentially forced to continue using a product or service regardless of quality, because the time- or monetary-cost of switching is not practical.

Omnichannel

A term for companies managing experiences on many platforms. Any combination of apps, websites, brick & mortar stores, etc make up "omnichannel."

WYSIWYG

"What you see is what you get." Typically describes a website builder that lets you edit a website in a visual way.

Headless CMS vs Traditional CMS

CATEGORY	HEADLESS	TRADITIONAL
Choice of technologies	Limitless	Restricted
Backend architecture	Microservice, best-in-class	Monolithic, all-in-one
Omnichannel support	Limitess	Restricted
Content model	Built for many formats and products	Built for a single page
Speed	Faster load times	Longer load times
Developer experience	Modern	Legacy
Use of plug-ins	Custom solutions	Yes
Technical debt	Managed	Inherent to provider
Localization	Yes	Yes
Hosting & delivery	Your choice	Provider
Hosting & delivery	Your choice	Provider specific
Workflow	Agile	Waterfall
Integration and deploy-ment	Continuous	Punctuated
Updates	Continuous	Scheduled
Vendor lock-in	Freedom of choice	Locked to the provider

Headless CMS for Modern Businesses

TECHNICAL BENEFITS

- Adapt to any speed and performance needs
- Fit your exact tec h stack and architecture needs
- Reduce tech debt by decoupling presentation from code
- Work faster in your preferred frameworks and code languages
- Unblock hiring bottlenecks caused by small, decreasing talent pools of legacy CMS experts

EASIER SCALING

Headless lets you manage your content across multiple platforms from a single source of truth, change developer tools at any time, and benefit from sending your content to high-performance cloud-based hosting.

ENHANCED SECURITY

Because a headless CMSes content is separated from the presentation layer it's a smaller area of attack.

LOW LEARNING CURVE

With a headless CMS, developers don't need to learn all the tech built around the CMS as they would with a traditional CMS. This makes your organization's time-to-value much faster.

DEVELOPER FREEDOM

Since headless CMSes are tech-agnostic, developers can choose their frontend tooling. They can interchange parts of your stack or move from one framework to another without affecting the CMS.

BUISINESS BENEFITS

- Easily expand to into new markewith new digital products
- Create and edit content once, publish in all channels
- Work independently of development cycles
- Work faster in your preferred frameworks and code languages
- Improve site performance, SEO, and marketing capabilities
- Eliminate publishing bottlenecks and inconsistencies

MORE CHANNELS, BIGGER AUDIENCES

A headless CMS isn't tied to a single presentation layer, say one specific website. This means it can reach audiences across multiple digital channels (ex. multiple websites, apps, VR/AR headsets, TVs, smartwatches, internal admin websites, etc).

UPDATE ONCE, UPDATE EVERYWHERE

When you need to update a product, service, or campaign across every instance, it's tedious. With a headless CMS, you update it in one place. And when ready, it publishes to every instance, even across multiple platforms.

FASTER AUTHORING EXPERIENCE

With a Headless CMS, you don't have to manage website concerns every time you publish content. This substantially increases the pace at which your business team and website team can work.

FUTURE-PROOFING YOUR BUSINESS

Since a Headless CMS can deliver content to any channel, you're set for new platforms that appear in the future. Adopting new platforms like smartwatches means you won't have to reauthor content. You simply access the content through their API and display it in their format.

THE FULL CHECKLIST FOR

Choosing a headless CMS

☐ Do you need an API-based or Git-based CMS?

With a Git-based CMS, you access your content by querying files stored in your repository. If all you need is a website(s), it's incredibly flexible and less expensive — there are no bandwidth limits, data caps, or vendor lock-in as you fully own everything.

With an API-based CMS, you access your content via the CMS API. If you need to power multiple digital platforms or have complex roles and permissions, this is the solution for you.

☐ Consider, is it industry-specific?

Some CMS solutions are meant only for e-commerce or content applications. Before you commit to a CMS like this, are you confident that being locked into an industry-specific tool will allow you to scale?

☐ Do you have to learn new frameworks?

A CMS that lets your team work in their preferred coding languages will speed up your time-to-value and make maintenance easier. If you have to write special frameworks or wrappers to make a certain CMS work, consider that your time-to-value will likely be slower.

☐ Do you have to learn new frameworks?

Someone will need to create and publish new pieces of content. Does the CMS you're looking at have a simple, visual way to achieve this? Make sure you can understand how the publishing flow will work for your business like before you commit!

☐ Can you create structure around how content is created and published?

As your company grows, you'll want to align the ever-growing creation of content with your brand identity. Does your CMS allow you to set up editorial workflows for what can be published, as well as what that content will look like on different platforms?

☐ Do you own the code underlying the website?

Code ownership will allow your company to truly control your company's website and digital properties fate. Ask if the headless CMS requires any dependencies, wrappers, or plug-ins that will force you to marry your digital properties to their system.

☐ What integrations can you use with the tool?

Some industry-specific CMS or "easy site builder" solutions have limited integration libraries. Before you commit, make sure you are confident they have the integrations for the tools you need. Look for solutions that allow you to own the code beneath the website and other digital properties.

THE FULL CHECKLIST FOR

Selecting key features

☐ Simultaneous editing

Think: a writing experience like Google Docs

☐ Content structuring

Can you set frameworks for how content should appear?

☐ Preview in multiple formats

When you hit publish, can you see what it will look on the site? On the app?

☐ Self-hosting

Do I own our company's destiny with hosting or am I limited by the tools op-

☐ Customizable workflow

What happens when you hit publish? Can you change that logic?

☐ Security

How is my content stored? Who can access it?

☐ Comment & collaborate

like editing a word doc, can an editor suggest changes?

☐ Open Source

Can our team edit and create add-ons that only we might need?

☐ Restoring content

what happens if something accidentally gets deleted? Can you save it?

☐ Ecosystem support

Is there a set, limited integration library? Who supports their technology?

The 3 Honest Downsides of Using a Headless CMS

1

REQUIRES A DEVELOPER

A developer needs to be on hand to implement a Headless CMS. For future maintenance, you'll also want a developer as customizations are too technical for a business user to enact.

2

DEVELOPER BOTTLENECK

Because a headless CMS's UI looks a lot like a database, all too often marketers end up dependent on developers to implement even the simplest of updates. This can overflow the tech team's ticketing queue with marketing requests; frustrate developers because they waste time on low-value requests rather than writing scalable code; and frustrate marketing because they can't independently ship against their KPIs.

3

THERE IS NO VISUAL EDITING EXPERIENCE

Marketers today are used to using visual site builder when editing their website. With a headless CMS, marketers are forced to use a UI/UX that can be very hard to navigate.

(Solutions to these problems listed in section 9.)

What to consider before you decide to go headless

IS A DEVELOPER OR AGENCY INVOLVED?

A developer needs to be on hand to implement a Headless CMS. For future maintenance, you'll also want a developer as customizations are too technical for a business user to enact.

NEED TO MANAGE MULTIPLE DIGITAL PLATFORMS?

Because a headless CMS's UI looks a lot like a database, all too often marketers end up dependent on developers to implement even the simplest of updates. This can overflow the tech team's ticketing queue with marketing requests; frustrate developers because they waste time on low-value requests rather than writing scalable code; and frustrate marketing because they can't independently ship against their KPIs.

THERE IS NO VISUAL EDITING EXPERIENCE

Marketers today are used to using visual site builder when editing their website. With a headless CMS, marketers are forced to use a UI/UX that can be very hard to navigate.

Stackbit: A Headless CMS & more

With all their benefits, headless CMSes still have one big problem.

Because a headless CMS's UI looks a lot like a database, all too often marketers end up dependent on developers to implement even the simplest of updates.

This overflows the tech team's ticketing queue with marketing requests; frustrates developers because they waste time on low-value requests rather than writing scalable code; and frustrates marketing because they can't ship against their KPIs.

Stackbit, out of the box, provides you with a headless CMS *and* a visual website editor.

Need to swap in a specific headless CMS? It's no problem. Stackbit's visual page builder continues empowering marketing to create new pages and add components.

And unlike other visual headless tools that force your to use their frameworks, Stackbit uses an annotation-only methodology to turn your custom website into a visual page builder.

```
<Link href={url} data-sb-field-path={annotations}>
  <span>{label}</span>
</Link>
```

The only code tied to Stackbit to make content editable is lightweight data attributes. And these attributes can be ripped out at build time to

---

With no set frameworks, wrappers, or plug-ins — you get an unparalleled combination of limitless development flexibility *and* marketer independence.

Architecting a new website or digital experience? Looking to unblock your marketers who are already using a headless website?

We can help.

Book a free architecture consultation

Book a demo, no strings attached

Make your headless website visually editable *without code dependencies*.
Empower your marketers to independently create pages, add components, and adjust layouts — while developers get to build without adding any new frameworks to your code.

Stackbit

Book a free architecture consultation

Book a demo, no strings attached