WHITE PAPER

The Complete Guide to Headless CMS

Decoupled and hybrid content delivery options with WordPress

One of the top trends in the world of content management is the emergence of headless CMS. In recent years, many organizations have begun evaluating new architectures to meet their content management needs. And emerging vendors have been more than willing to fan the hype around headless CMS.

But what exactly is headless CMS? Is it for you? And what's the best way to get the benefits of a headless architecture without creating unneeded complexity or reliance on untested technologies?

In our guide, we'll discuss the plus points of headless CMS, things to watch out for with headless architecture, and how a global media organization delivers great content to a worldwide audience via a headless implementation.

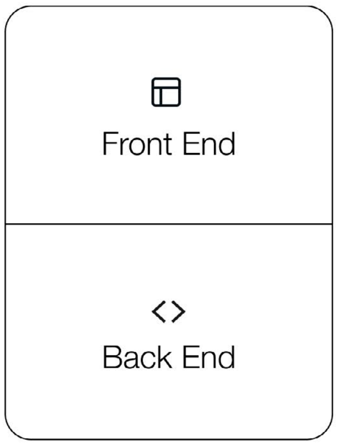
What headless is and why it's different

Traditionally CMS has contained two components:

- **The back end:** The CMS stores and manages all the content a company has produced. This included all the tools content producers use to create content, the taxonomy and nomenclature of the content and the storage and retrieval of that content.
- **The front end (or "head"):** Here the CMS manages the presentation of the content. This includes themes and templates that power the way the content is presented to the viewer.

In a typical CMS deployment, these two pieces of technology are delivered by a single solution. In this model, a full-featured Content Management System (CMS) directly renders the user experience of your site in a web browser. The most popular has been WordPress, which today powers more than 40% of the web. By delivering everything in one package, content marketers and other content producers were able to easily create content faster and take advantage of a rich ecosystem of themes to create a great customer experience.

Headless CMS operates differently. It removes the front end entirely. In headless architectures the CMS manages only the back end and exposes APIs to access content. It is then up to the organization to build their own custom front end. Because the back and front ends are "decoupled," the user experience can be built with any technology of your choice. Often this means using technologies like Node.js to produce dynamic web pages. Regardless of the technologies, the CMS is not involved directly in defining the presentation of the content.

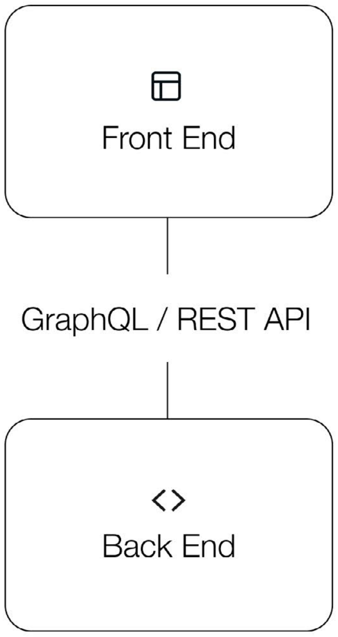


TRADITIONAL

Advantages of going headless

So if a headless CMS is similar to a traditional CMS but lacks a core component, why would anyone want to adopt this? Why is it such a hot topic? There are several key reasons why headless CMSes have become more popular recently:

- **Deeper control of the experience:** By necessity, even the most complex theme or template for a web page has its limitations around customizability. By putting full control of the experience in the hands of the developer, headless architectures basically let developers do whatever they want. This can lead to highly differentiated experiences.
- **Flexibility and iteration:** Developers can essentially take their pick of front-end technologies. They can more quickly make changes in their code based on their analysis of user behavior and potentially iterate faster.
- **Omnichannel support:** Customers no longer access content only from websites. They also use mobile apps, IoT devices, smart speakers, digital displays and much more. Decoupled applications can more easily be written with these new user touchpoints in mind.



HEADLESS

- **Data ingestion and use:** Headless architectures can improve performance in certain instances. For example, if your site needs to ingest content or data from multiple sources and then present it to users without making a stop at your CMS database, headless might be a good solution. Deeper control of the experience: By necessity, even the most complex theme or template for a web page has its limitations around customizability. By putting full control of the experience in the hands of the developer, headless architectures basically let developers do whatever they want. This can lead to highly differentiated experiences.

For these reasons headless architectures have become a favorite of development and other technical teams. Often the adoption of a headless CMS accompanies the growth of development teams and new investments in digital transformation.

Downsides of headless architectures

All of this comes with a cost. Over the years, a whole ecosystem developed around CMSes. Headless CMSes often abandon years of know-how and rock-solid off-the-shelf technologies in favor of having developers "reinvent the wheel." This can create challenges around:

- **Complexity:** The flexibility of headless deployments also typically means a more complex architecture, with multiple components necessary to build the experience. Each is another piece of infrastructure that has to be designed, managed, and maintained. Failure to do so effectively can cause performance, reliability, and scalability issues.
- **Expense:** All of this comes at a cost. Not only is it often expensive to maintain complex environments, but it typically takes more development resources to make changes. Instead of building from off-the-shelf themes, everything has to be built from the ground up.
- **Dependence on developers:** One of the reasons CMS emerged in the first place was to reduce dependence on developers. Headless CMS puts developers back in the mix when it comes to relatively simple configuration or customization of the customer experience. But this can create bottlenecks that limit content agility.
- **Weaker tools for content creators:** Because new headless CMSes were built with a developer-first mindset, many lacked good and

intuitive tools for content creation. This means non-technical staff waste time trying to create the content an organization needs to fuel growth, partially negating the agility promised by many headless vendors.

Each of these is a serious pitfall and why many headless CMS projects ultimately fail to live up to their hype.

Four things to consider when choosing the right architecture

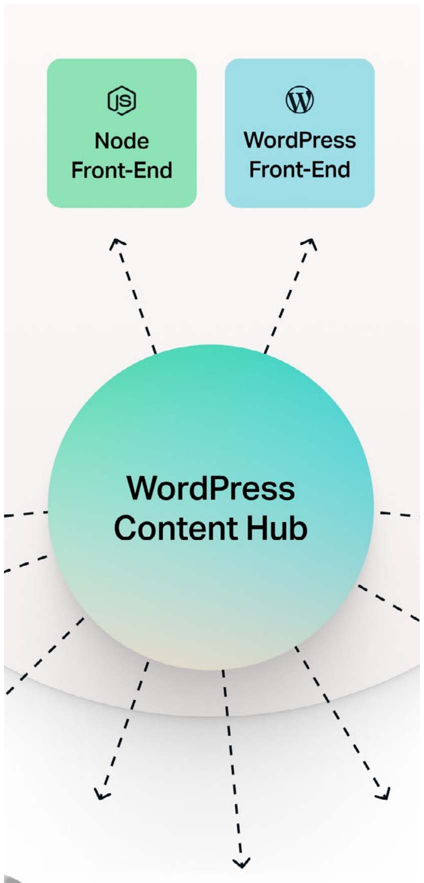
1. What problem are you trying to solve? If it's speed or security, in most cases we'd recommend optimizing your codebase before changing to a headless architecture. On the other hand, if you have unique experience needs across multiple channels, headless might be a good fit.
2. How are you currently measuring the impact of that problem? Whether you are using a tool to measure performance, measuring the time it takes your development team to ship new features, or understanding what percentage your customer base values various experiences, use data to inform your decision.
3. What additional resources will your choice require and do you have them at hand? Do you need to hire new developers to build and maintain the front end of a headless solution? Have you thought about the requirements for your DevOps team, when it comes to managing separate repositories for your CMS and your front-end application?
4. Where will you host your front end? Will your current or future provider of CMS technology also be able to host your application? If not, where will you host it and how will you plan to ensure the right performance and integration?

Decoupled delivery via Agile CMS: The best of both worlds

Once upon a time, headless and single-stack were an either-or choice and if your answer to these questions changed over time, it could require a rip-and-replace type of shift. Thankfully, new technologies are emerging that bring the advantages of headless CMS without many of the pitfalls. Forrester Research recently defined a new category—Agile CMS—that has moved the market beyond headless.

Agile CMSes are built around a central content hub. Required capabilities include:

- **Decoupled architecture:** Instead of forcing one model or the other, organizations can build their own front end or use the existing CMS-provided front end. They can even choose to build different experiences using different models—for example, building their website with an existing theme while using an API-first approach for a mobile app.
- **One central content hub:** No matter where the content goes or how experiences are developed, it pulls from a single content repository that supports access from REST and GraphQL APIs.
- **Continued access to ecosystem:** Years of existing plugins, themes, and templates remain available and can be used where appropriate, without losing flexibility.
- **Superior content authoring tools:** Agile CMS gives content creators all the intuitive content creation tools they expect from leading CMSes such as WordPress, even if the front end is something custom.



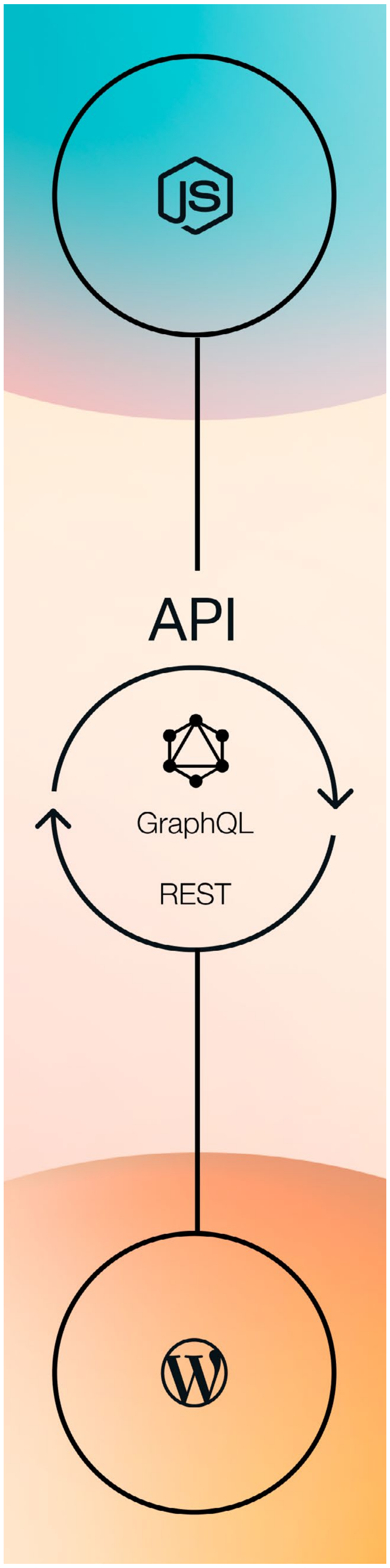
Decoupled delivery with WordPress

WordPress has frequently been used in a decoupled state by publishers, whose sites ingest content from WordPress JSON feeds into native mobile applications and other user experiences that aren't necessarily conventional websites. From popular iPhone apps for news organizations to networks of digital signage in hotels, WordPress is behind the scenes of a large range of user experiences.

In a traditional delivery model, WordPress themes include templates to manage the display of content on the front end. Content production, a templating engine, third party integrations, even design and user experience, all come in one handy package.

In a headless architecture, WordPress is separated from the templating engine, design, and user experience. In this model there are several pieces that may be required:

- **A Javascript Framework:** While not strictly required, JavaScript frameworks and libraries have become a popular way to build the front end of websites, independent of the underlying CMS. While it is possible to write their Node.js applications from scratch, it's popular to use a framework like Next.js, Gatsby, Frontity, etc.
- **APIs:** Content will be accessed via APIs.
  - **REST** APIs are popular for reliable, extensive, and extensible source of JSON feeds of content and data from your WordPress admin. Many approaches to headless WordPress consume REST API and parse the requested data into React templates for display.
  - **GraphQL** APIs are an increasingly popular alternative. It is a Query Language built to craft more precise (and lightweight) queries to bring only the data you need from your CMS into your front-end application.



How WordPress VIP simplifies headless WordPress

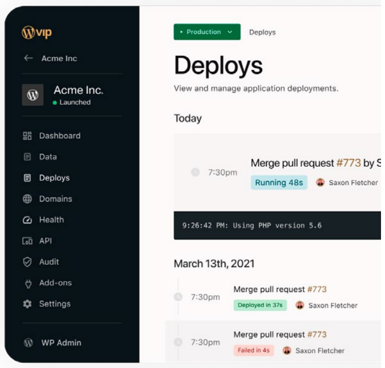
While there are many reasons organizations want to deploy their CMS in a headless architecture, one of the inherent disadvantages of headless CMS is complexity. It involves more components, not least the additional code for one or more front-end applications.

More pieces mean more things to manage and more things that can go wrong. Some, but not all, of this problem is unavoidable. However, with recent enhancements, the WordPress VIP platform reduces this complexity for your teams, ensuring the right foundation for headless success.

Hosting of node applications

A key component of many headless architectures is Node.js apps. With no built-in front end, a company's developers have to develop their own, composed of one or more Node apps. But those apps have to live somewhere.

We've spent years developing a solid platform designed for security, performance, and management. Furthermore, it's the same platform, run from the same data centers that host our CMS product, WordPress. By running on this platform, Node applications take advantage of all these core platform services, reducing the number of suppliers and services that can cause failures, and easing troubleshooting.



Deploys

View and manage application deployments.

Today

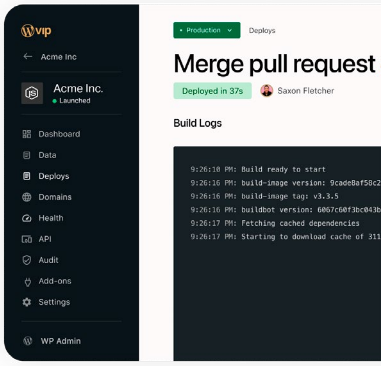
7:30pm Merge pull request #773 by S
Running 48s Saxon Fletcher

```
9:26:42 PM: Using PHP version 5.6
```

March 13th, 2021

7:30pm Merge pull request #773
Deployed in 37s Saxon Fletcher

7:30pm Merge pull request #773
Failed in 4s Saxon Fletcher



Merge pull request

Deployed in 37s Saxon Fletcher

Build Logs

```
9:26:10 PM: Build ready to start
9:26:16 PM: build-image version: 9cade8af58c2
9:26:16 PM: build-image tag: v3.3.5
9:26:16 PM: buildbot version: 6067c60f3bc043b
9:26:17 PM: Fetching cached dependencies
9:26:17 PM: Starting to download cache of 311
```

Bundling of everything needed

Headless architectures often require multiple pieces or components, including CMS plug-ins, API packages, and databases,which take time and effort to deploy and manage.

We've packaged all this into a single headless bundle that can be instantly deployed on our platform. Getting started takes a fraction of the time.

Databases on-demand

Many Node applications in a headless architecture tend to be stateless. But not all.

Where storage is needed, a Redis or similar in-memory database can be a good choice. However, it is important to ensure the security and reliability of the connection to this data store. Some choose to connect these components with third-party VPNs, which can introduce even more complexity. We've simplified this challenge by making Redis available right in our solution. If and when you need a data store, it's easy to simply turn it on alongside your application in a private network. Everything works securely and with high performance right away.

Next.js Quickstart

Once you've got all the pieces in place, you need to start writing your own code.

Each application will be unique based on your specific needs and desired experience. That's probably part of why you chose a headless architecture in the first place. However, most applications have similar requirements like content preview, content querying, etc.

We've included sample skeleton code that provides good solutions to each of these needs. This saves time, ensures performance, and lets you focus on your unique needs rather than the mundane nuts and bolts.

WordPress VIP: a strong hybrid option

While you may be looking for a headless architecture today, many organizations find over time it makes sense to combine the flexibility of headless with the out-of-the-box functionality of a single-stack CMS.

For example, they may have multiple web properties and want to use existing WordPress themes for some while focusing their developers on only a few key sites. Others may find they want to serve the web with a more traditional technology stack while powering other digital experiences with an API-driven headless architecture.

With WordPress VIP, you don't have to choose headless or traditional single stack deployment. We provide a single content hub that can power a hybrid deployment, fixing and matching architectures to meet your diverse needs. No matter how you deploy today and in the future, it's all managed from a single administrative console, and content creators can build content blocks using our easy and intuitive Gutenberg editor.

Going headless—where to get started

WordPress VIP customers Al Jazeera, News UK, Quartz, TechCrunch, and Fortune are all running decoupled solutions today. Each of these sites consumes performant and reliable feeds from the WordPress API, produced by their installation of WordPress running on the WordPress VIP platform, and using a variety of headless configurations. Overall, WordPress VIP delivers a powerful, easy and flexible option for headless deployments. Contact us to learn more.

About WordPress VIP

We are WordPress VIP, the agile content platform leading a powerful enterprise ecosystem. As we fit seamlessly across your organization, enjoy the ease, flexibility, and freedom you need to scale the digital experiences that drive your growth.

WordPress VIP offers content management, enterprise commerce, and content analytics. Our platform provides best-in-class digital creation, combined with the insights to understand and optimize the impact of those digital experiences.

Our team has been delivering WordPress at scale for more than a decade, and will guide your team with architecture, integrations, plugins, agencies, and more.

Learn more at wpvip.com.