Human Made

The Human Made Guide to Headless WordPress

The definitive guide to using headless WordPress in enterprise organisations

paragraph

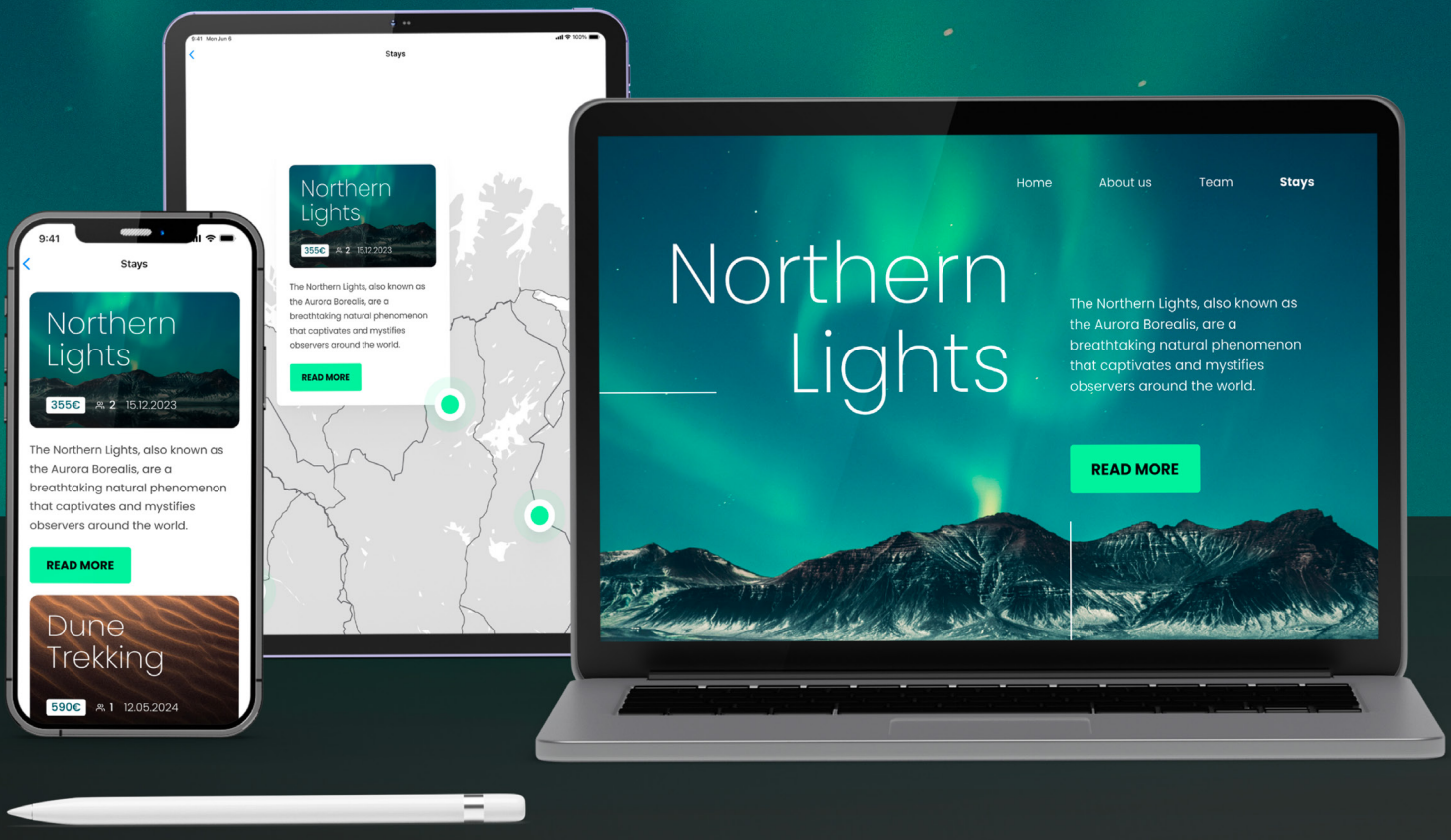
The Northern Lights, also known as the Aurora Borealis, are a breathtaking natural phenomenon that captivates and mystifies observers around the world. Occurring in the Earth's polar regions, these mesmerizing displays of colorful lights illuminate the night sky with vibrant hues of green, blue, pink, and purple.

header

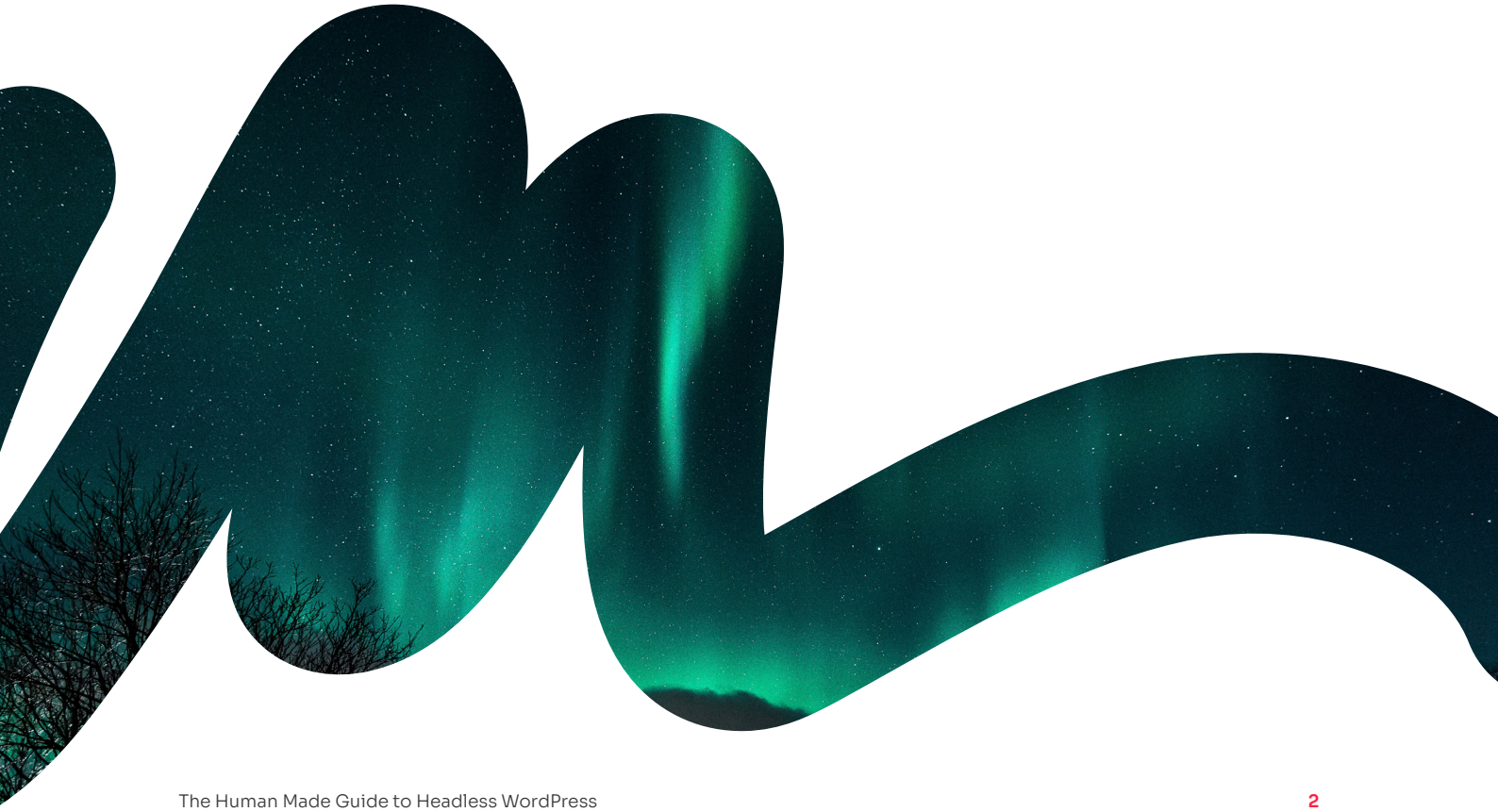
Northern Ligh

button

READ MORE



Contents



A word from our CEO, Tom Willmot

I'm proud to say that Human Made has been a leading contributor to WordPress for well over a decade now, having been actively involved in developing the software since version 3.0.

Our experience in WordPress spans years of modifications and evolutions and we have seen dramatic changes to the way WordPress has been adopted across a range of industries and enterprises.

WordPress' move beyond being a monolithic CMS to acting as a central hub was facilitated in no small part by the REST API. Exposing WordPress content and data as JSON via a standardised RESTful API helped unlock data and enabled an explosion in the number and complexity of integrations.

Tom Willmot
CEO, Human Made

Some of our team are particularly involved in the projects and technologies vital for creating performant and engaging headless experiences with WordPress.

Joe Hoyle
CTO, Human Made

Joe leads our development team and the overall technical direction of the company. He is a member of the WordPress REST API development team.

Ryan McCue
Director of Product, Altis

Ryan joined Human Made in 2014. He's been involved with WordPress for over a decade and is also the founder of and focus lead for the WordPress REST API.

KAdam White
Principal Web Engineer, Human Made

Principal Web Engineer, Human Made: KAdam joined Human Made in 2017 after several years of working with Joe and Ryan on the WordPress REST API. He is a component maintainer for the WordPress REST API the 'wpapi' package on npm.

I hope this Human Made guide to headless WordPress will help smooth your path towards building a high performing headless experience, but please do reach out if we can provide your business with more support or advice on headless architecture.

Introduction

It's no secret that WordPress **powers 43% of all websites on the internet.**

An open source content management system (CMS), WordPress' usability, extensibility, and deep talent pool makes it a popular choice for sites of all shapes and sizes, and it is gaining increasing traction among enterprise organisations.

WordPress, like many other CMSs, is monolithic. Also known as traditional or coupled, a CMS with monolithic architecture offers everything needed to store, manage, and publish content under one roof. It can also be extended with third-party plugins and themes.

Nothing stands still for long, however; CMSs have evolved to act as a central hub, consuming and aggregating content and data from other tools and services, and exposing their own content and data via APIs in turn.

The WordPress REST API, founded by Human Made's Ryan McCue, facilitated such an evolution, exposing WordPress content and data as JSON via a standardised RESTful API which unlocks data and enabled an explosion in the number and complexity of integrations.

The REST API allows for an easier experience in:

- Separating frontend delivery from the CMS
- Powering multiple frontends from the same content (e.g. websites, apps, etc.)
- Using WordPress as part of a complex multi-service workflow (e.g. pushing content to a translation service then pulling translations back into WordPress)

For large custom builds and applications in particular, the benefits of the REST API are far-reaching. Upon the API's release back in 2016, engineers were able to work on discrete parts of a larger project independently, empowering development teams to move faster and build iteratively.

The stability of third-party integrations was also improved with the REST API, and WordPress was no longer a web development tool used in isolation. Today, it's one module within a web developer's toolkit; a building block to be used across myriad applications.

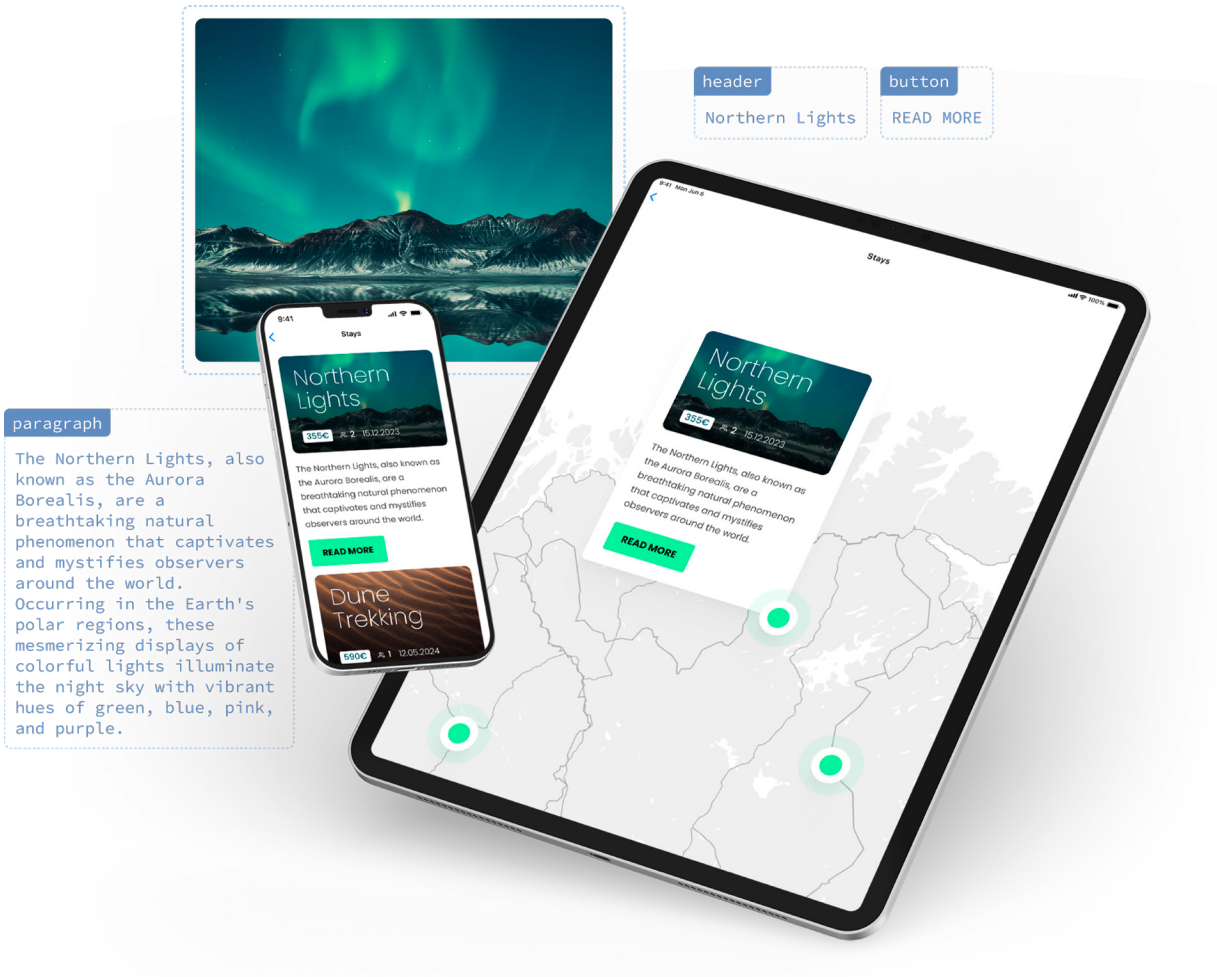
In this guide to headless WordPress, we'll explore the ins and outs of the headless CMS and the REST API, as well as taking a look at how some organisations have successfully adopted headless architecture for the benefit of their business and their teams.

If you've got questions or would like more support with headless architecture, don't hesitate to reach out to us at hello@humanmade.com.

The Headless CMS

A headless CMS is used only for data capture, storage, and delivery, making it frontend agnostic. Its data can be displayed using any frontend technology, whether in a browser, mobile application, syndication, or elsewhere.

A traditional CMS deals with data collection, delivery, and display.



WordPress, for example, has a backend where users can enter data. This data is stored in a MySQL database, retrieved from the database using PHP, and then displayed in the browser using the Theme system.

A headless CMS decouples the Theme system, allowing it to replace it with any preferred frontend technologies. What's left is the data storage method and web application for authors and editors, while the data is delivered to the frontend using an API.

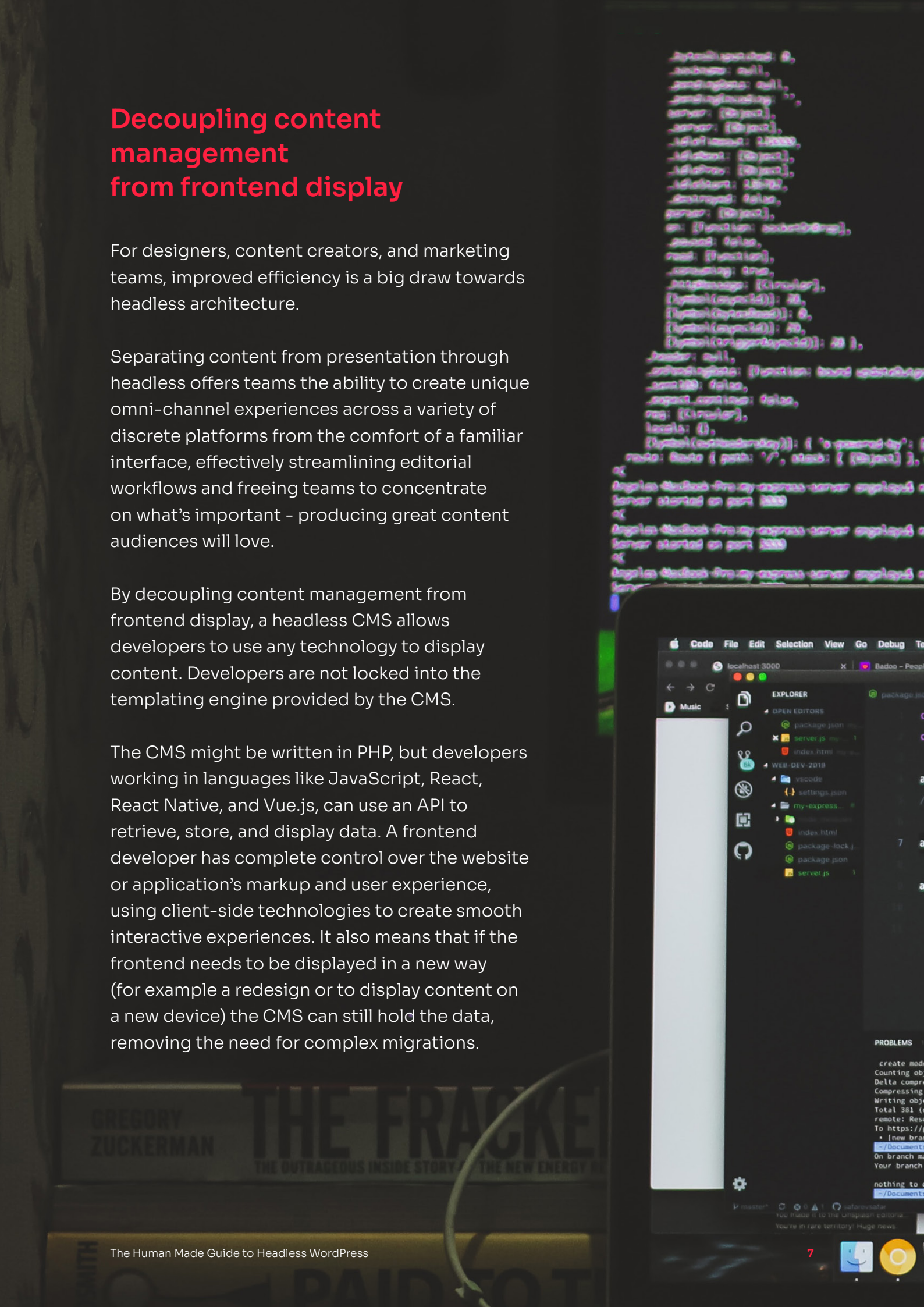
Decoupling content management from frontend display

For designers, content creators, and marketing teams, improved efficiency is a big draw towards headless architecture.

Separating content from presentation through headless offers teams the ability to create unique omni-channel experiences across a variety of discrete platforms from the comfort of a familiar interface, effectively streamlining editorial workflows and freeing teams to concentrate on what's important - producing great content audiences will love.

By decoupling content management from frontend display, a headless CMS allows developers to use any technology to display content. Developers are not locked into the templating engine provided by the CMS.

The CMS might be written in PHP, but developers working in languages like JavaScript, React, React Native, and Vue.js, can use an API to retrieve, store, and display data. A frontend developer has complete control over the website or application's markup and user experience, using client-side technologies to create smooth interactive experiences. It also means that if the frontend needs to be displayed in a new way (for example a redesign or to display content on a new device) the CMS can still hold the data, removing the need for complex migrations.



Fast, interactive experiences

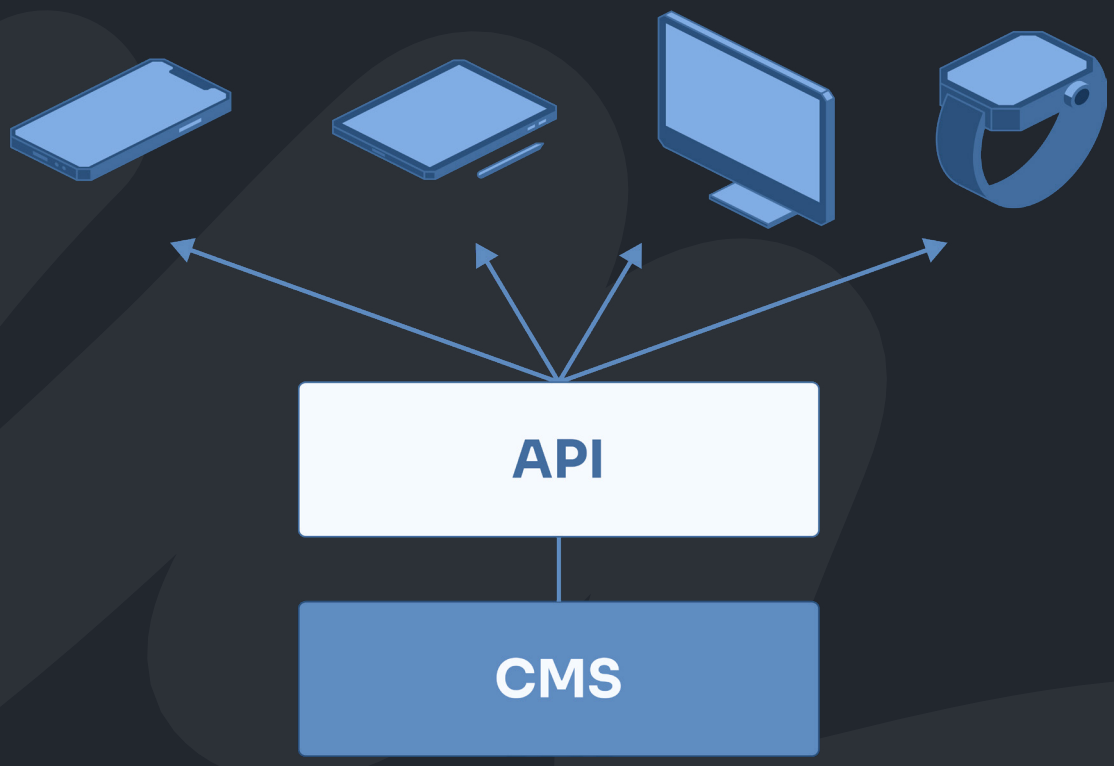
There are two components to a headless CMS: the CMS itself, and the frontend display. Splitting your website or application in this manner means you can improve performance and provide a superior user experience.

The CMS focuses only on content management, without having to assemble formatted responses, while the client-side technology can quickly display that data in the browser. Using client-side technologies for display means that in-browser experiences are fast, acting in real-time, without having to wait for PHP to generate entire pages.

There is a significant increase in performance when using JavaScript vs PHP: Node.js, for example, can handle many more requests than PHP due to its asynchronous, event-driven nature. This can be especially useful when an application requires many connections open simultaneously.

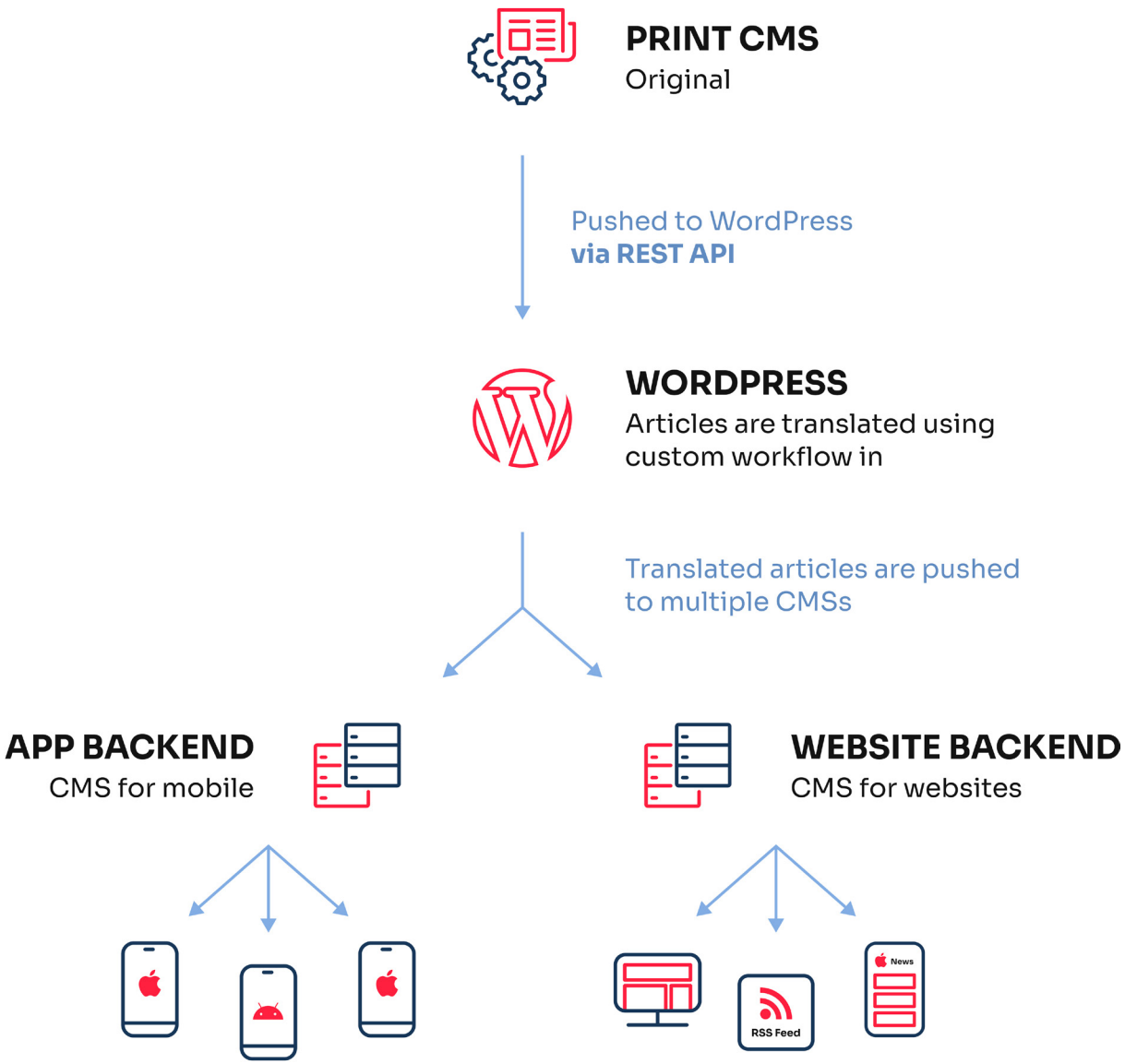
One content management system, multiple frontends

Gone are the days of burning audiences through poor UX on their channel of choice - building unique experiences across all channels means consumers can interact with a brand in the way that works best for them. Marketers can sleep soundly knowing their content will be able to shine no matter where users find it.



Multi-service content pipelines

A headless CMS can be used to store all of the data for one site or application, or it can be one element of a larger application that retrieves and aggregates data. This means that data can be integrated into existing workflows as just one layer. It could be used as a layer for translating content which is then pushed to another CMS, for example.



TechCrunch

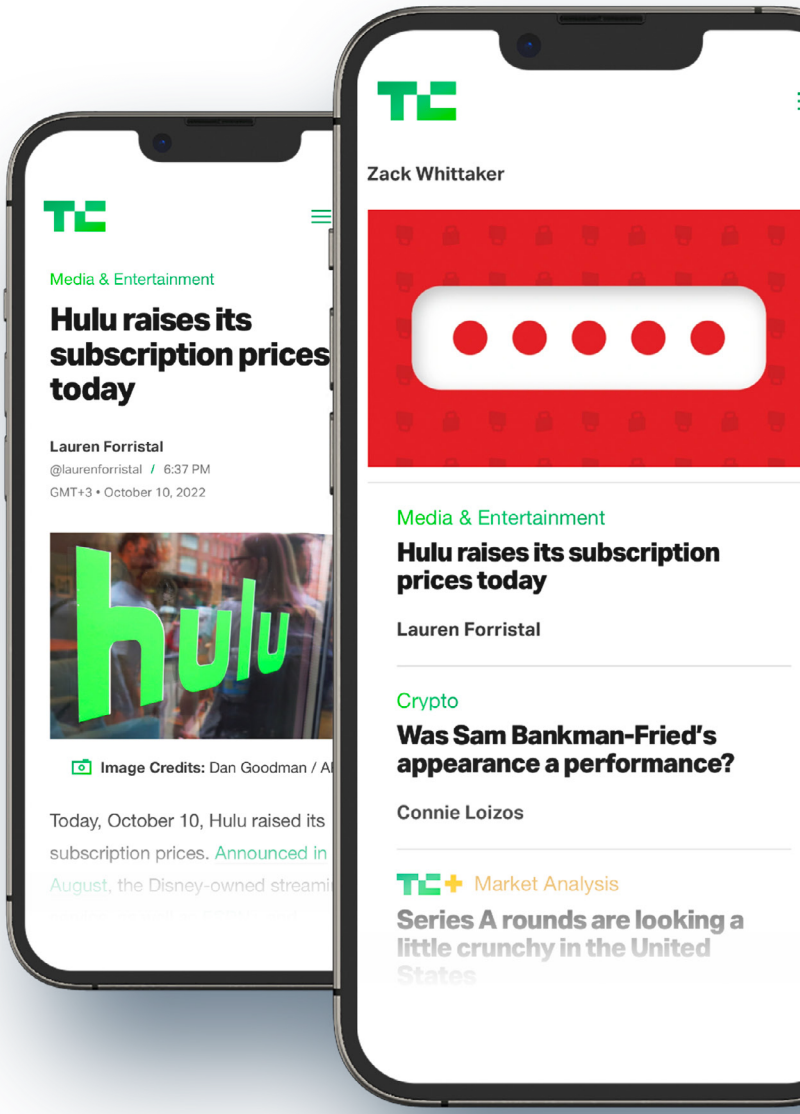
Case study

As TechCrunch's technology partner for their renovation project, we delivered a full site rebuild and a WordPress CMS with a headless React frontend on traditional managed WordPress hosting.

Why WordPress and the REST API?

TechCrunch wanted to decentralise their publishing experience: a backend with editorial simplicity, and a user-friendly frontend were their goals.

Maintaining WordPress' backend on their system enabled TechCrunch's non-technical teams to independently run individual pages, while making use of React on the frontend enabled editors and authors to create tailor-made solutions for each new page, empowering more people in their team to write and publish content effectively.



TechCrunch

Case study

The build

We implemented a headless CMS on a PHP-only hosting infrastructure, hosted with WordPress.com VIP Go. We introduced our own VIP Go Builder which was later adopted by WordPress.com VIP, adding a build process that only has to deploy the master-build branch to send all merged changes live to the production website.

For our local development environment, we extended the React app tool with WordPress-specific customisations. This meant we could load the application and the stylesheets from WordPress.

We used React on the frontend to display data, which required us to write the same boilerplate code repeatedly to retrieve data from WordPress. The repetitive nature of this task inspired us to create Repress, a Redux Library for the WordPress REST API.

Repress enables developers to retrieve data from the REST API, and add it to the store with just a few lines. Unlike many other React libraries for WordPress, Repress can be added to an existing store, allowing progressive adoption, and can even be combined with other methods of retrieving data from the WordPress REST API.

Repress not only enabled us to facilitate the process of retrieving data from WordPress; it also introduced a higher order component to interface with React, making it easy to add data management to presentational React components.

On the frontend we created specific page functions, like the river homepage experience that creates a seamless and responsive user experience. It enables users to scroll and click on an article, which then snaps closed to resume their exact browsing position on the homepage.

TechCrunch

Case study

We also integrated their partner database for startup news, Crunchbase. Our team of engineers integrated the Crunchbase API with the React frontend to display detailed information on companies, investors, founders, and more, as hover cards within the articles. Read more about Our React Tools for WordPress.

> *We wanted our redesign to be ambitious, and it was critical for us to choose a development partner who could deliver it without compromising on our aspirations.*
>
> *Our design focused on fluid and interesting interactions over bold visual statements, heavily influenced by our decision to use the WP REST API with a JavaScript frontend. As such, it made sense to choose a partner that brought expertise in that realm, which Human Made certainly did."*

Nicole Wilke
Head of Product, TechCrunch

What is an Open API?

Open APIs are publicly available APIs that give developers access to proprietary software information that they can make use of in their own software and applications. REST is the ideal architecture for creating an Open API for the web because, by using HTTP, it is built on the principles of the open web. To leverage an open REST API a developer just needs to make a HTTP request.

By making data available for developers to use in their own applications, open APIs have transformed the internet. Developers can access data across services, creating applications that aggregate information from different providers, and leveraging that data to their own needs.

This aggregation of public data across different platforms enables the creation of feature-rich, powerful applications that do more than any individual product or service could do on its own.

API options for headless builds

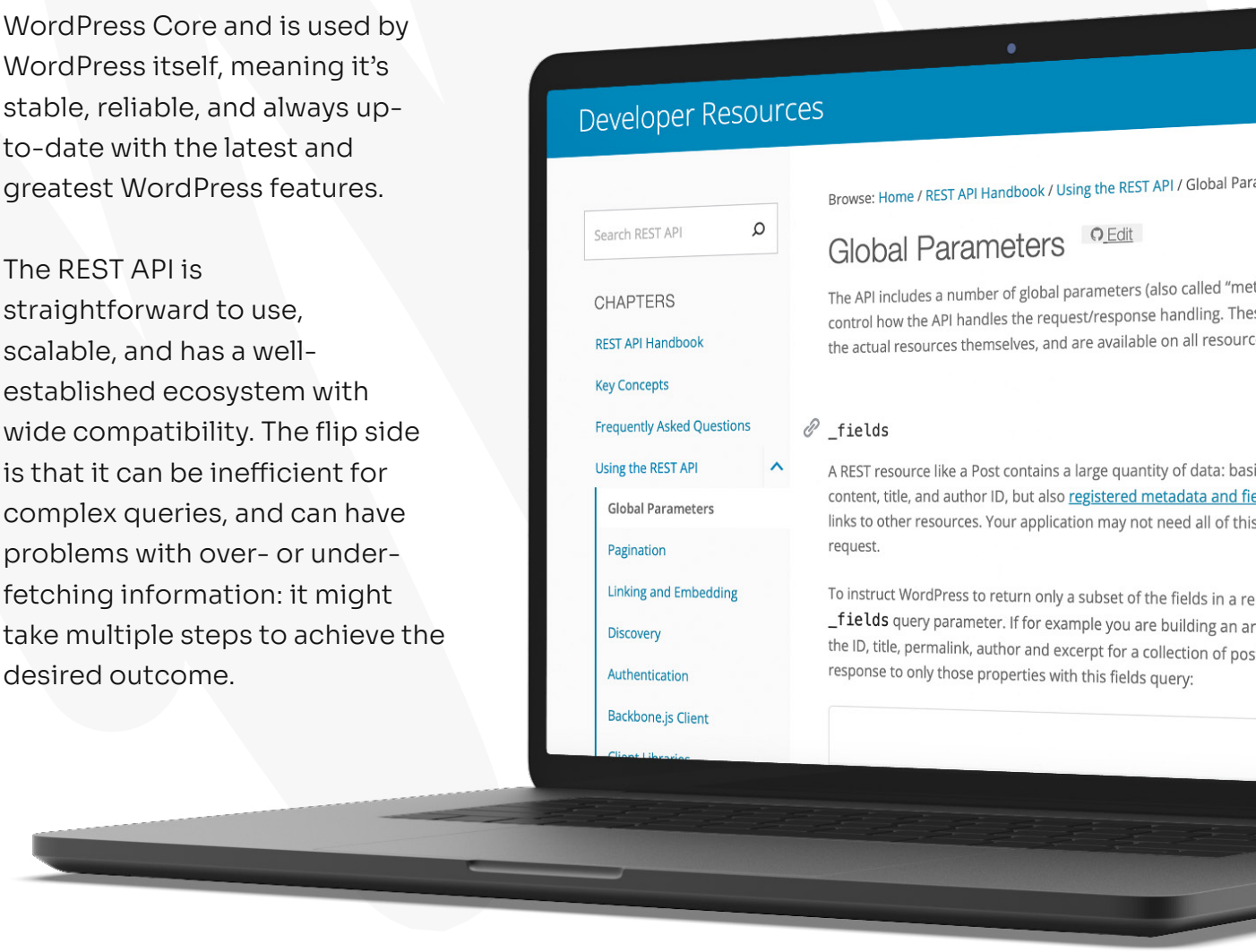
Deciding which API best fits the needs of a site depends on individual requirements. Headless development can use many potential approaches, with mature solutions using both REST APIs and GraphQL available for WordPress.

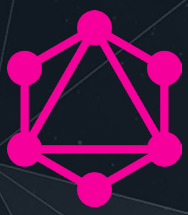
REST API

The WordPress REST API provides an interface for applications to interact with WordPress sites by sending and receiving data as JSON objects.

Developed by Human Made colleagues including Ryan McCue, Joe Hoyle and KAdam White, the REST API is built into WordPress Core and is used by WordPress itself, meaning it's stable, reliable, and always up-to-date with the latest and greatest WordPress features.

The REST API is straightforward to use, scalable, and has a well-established ecosystem with wide compatibility. The flip side is that it can be inefficient for complex queries, and can have problems with over- or under-fetching information: it might take multiple steps to achieve the desired outcome.





GraphQL

GraphQL, originally created by Meta, is a query language API that allows a client application to request just the needed data from a remote server. This can include a complex query with specific fields and connected objects.

Compared with the REST API, a big plus of GraphQL is that it can be more efficient in dealing with large amounts of data. It's fast, allows for rapid development, and can be less error-prone. It is, however, more complex to work with, lacks standardised tooling, and doesn't have the full functionality of the REST API.

Unlike the REST API, GraphQL is not supported out of the box by WordPress, and requires third-party plugins instead. Users should also be aware that its third-party status means GraphQL may not always reflect the latest features or functionality of WordPress, and it may not be updated as regularly as an API built into WordPress.

GraphQL can also be prone to issues like malicious queries or resource exhaustion attacks, and site developers should pay close attention to the security of the system.

The hybrid approach

Combining the strengths of GraphQL and the REST API, hybrid design patterns can result in flexible and efficient API solutions.

One of the more common hybrid patterns sees GraphQL used for bulk data fetching, with REST APIs used to augment this where complex interactions are necessary. This approach minimises requests and data over the wire, without being constrained by either API's limitations.

What is the WordPress REST API?

Transforming WordPress into a headless CMS with the REST API.

The WordPress REST API allows access to a website's data, including users, posts, and taxonomies. In the past, developers needed to use WordPress' built-in Theme system and administration panel to display and edit content on a site.

The REST API decouples the WordPress backend from the frontend, allowing developers to use it as an application platform: WordPress is used for data entry and storage, and the frontend can be built in any programming language. The REST API transforms WordPress into a headless CMS.

```php
<?php
/**
 * Grab latest post title by an
 *
 * @param array $data Options fo
 * @return string|null Post titl
 */
function my_awesome_func( $data
  $posts = get_posts( array(
    'author' => $data['id'],
  ) );

  if ( empty( $posts ) ) {
    return null;
  }

  return $posts[0]->post_title;
}
```

Endpoints

Endpoints are functions that are available through the API: they're the places where developers can do something with the CMS, whether that's creating, retrieving, updating or deleting (CRUD) data. Developers building a website, application, theme, or plugin can leverage the API by adding their own custom endpoints.

Authentication

For accessing private data or publishing new content, authentication allows users to establish who they are and whether they're authorized to perform actions. WordPress includes support for Application Passwords, as well as allowing plugins to add other systems.

Application Passwords

Included within WordPress are Application Passwords, which allow users to create a password specific to an application or tool using the API, rather than sharing their full password. These require a setup process that the user performs, but are available in every installation of WordPress, and are reliably available on every site.

OAuth

Third-party plugins provide support for OAuth, an authentication standard supported by many existing tools and libraries across every programming language. Users are directed to click a simple "Authorize" button, with all of the work happening in the background. This is a much simpler flow for users, but requires installing third-party plugins, such as the OAuth 2 plugin by the REST API team.

JSON Web Tokens (JWT)

An increasingly common web standard, JSON Web Tokens (JWT), can also be supported through the plugin ecosystem. Users enter their username and password once, and applications exchange this for a JWT, ensuring user data is kept private. Many plugins provide this functionality, including the popular JWT Authentication plugin.

Fairfax Media

Case study

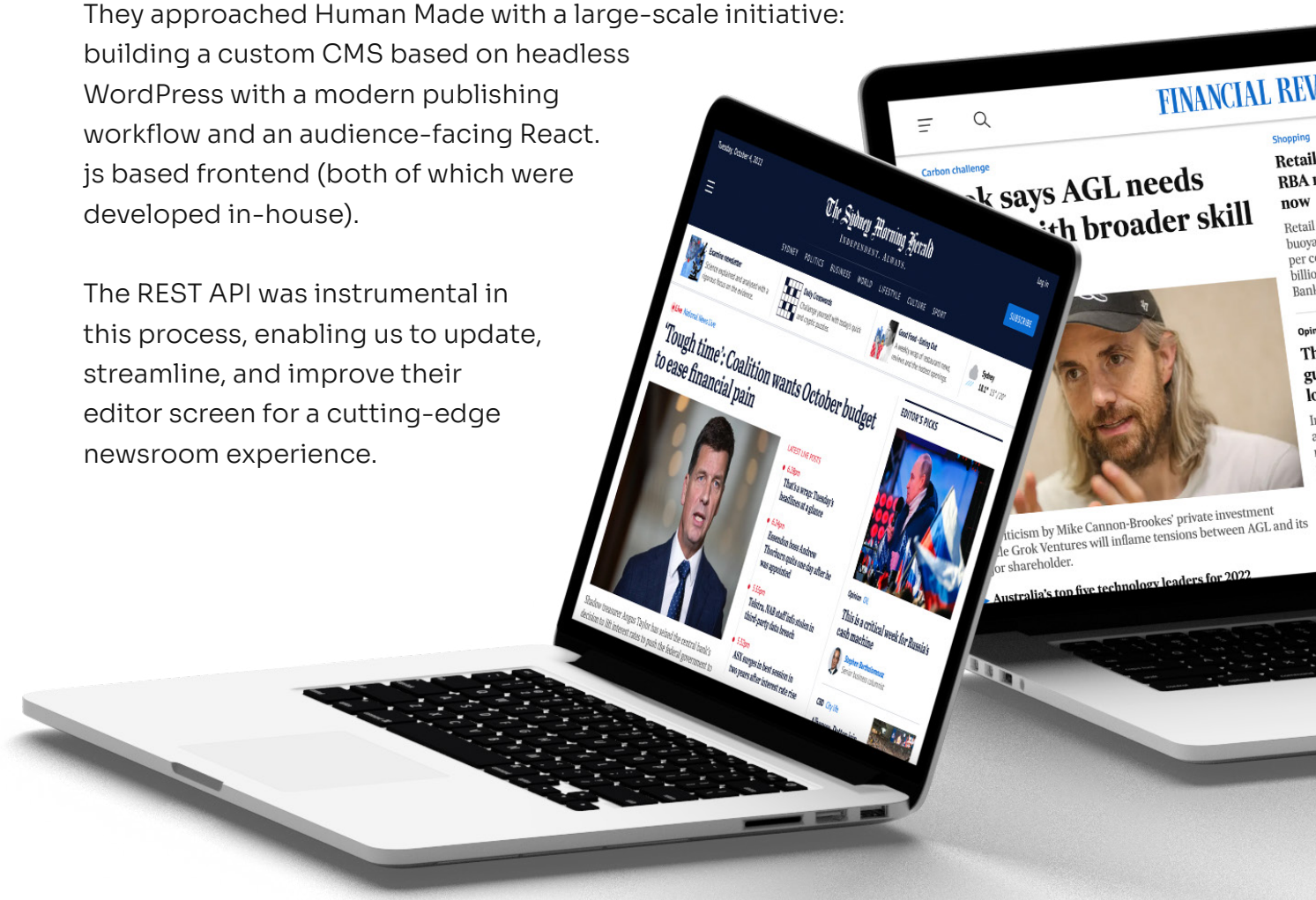
We joined Fairfax Media for an end-to-end newsroom transformation, using the REST API to improve editorial workflows and publishing processes for Australia's leading media brand.

Why WordPress & the REST API?

Fairfax Media wanted a tech partner to support them through their digital evolution, involving streamlining and improving their editorial and publishing experience.

They approached Human Made with a large-scale initiative: building a custom CMS based on headless WordPress with a modern publishing workflow and an audience-facing React.js based frontend (both of which were developed in-house).

The REST API was instrumental in this process, enabling us to update, streamline, and improve their editor screen for a cutting-edge newsroom experience.



Fairfax Media

Case study

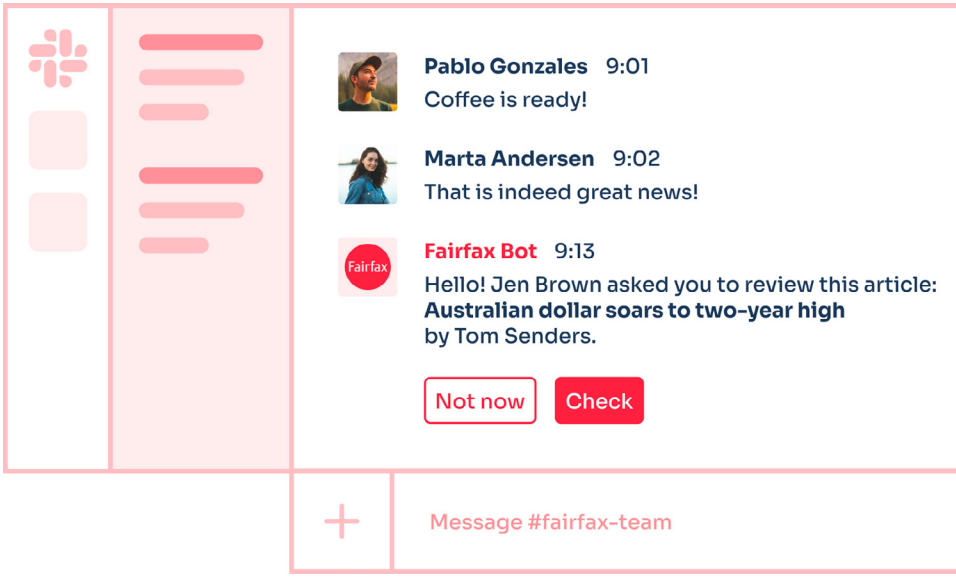
The build

Fairfax wanted to move quickly towards new and more effective publishing workflows for their team of editors and journalists.

We created a custom feature to allow journalists to search the Fairfax Content API for wire feeds from other news sources, allowing journalists to select an article that is then injected into WordPress for editing, customisation, and publishing.

Once we'd built the wire feed importer, we began to refine other areas of the editorial process through modifications on the edit screen. The WordPress edit screen can be slow, and doesn't offer the publishing experience required by a large media organisation where numerous people need to publish content around the clock.

One of the most impactful changes we made was using the REST API to help the edit screen load faster, and prevent the lag which is caused by the 'Update' (post) button, resulting in the page refreshing in order to effect an editorial change. We went with the REST API because we were extensively customising the edit screen and wanted it to feel modern, responsive, and dynamic.



Fairfax Media

Case study

We achieved this by enforcing changes directly in the browser to prevent delays, and the most effective way to make this happen in WordPress is to communicate with the REST API.

At Fairfax Media, WordPress is used as an editorial interface only and the system by which reporters' create and file stories. But it is WordPress' capacity to communicate with the wire feed, content, and media APIs that enabled us to create a flexible and streamlined tool.

It enabled the organisation to centralise its data, providing more consistency and convenience to the newsroom workflow, and created a clean workspace they could use as a place for data entry, making their admin area much more professional and efficient.

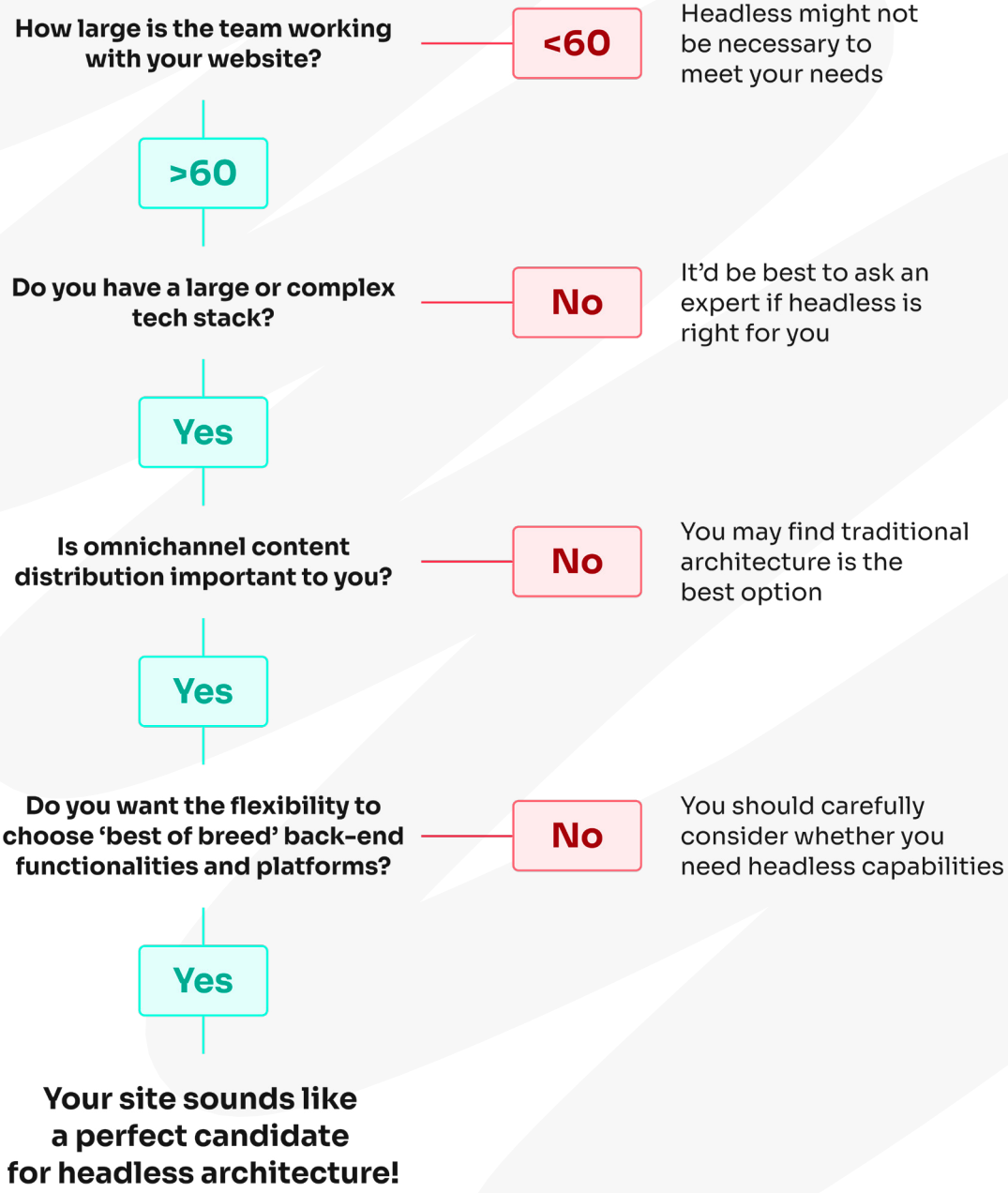
> *Human Made was instrumental in developing our editorial experience. Their expertise in WordPress was key to a successful outcome."*

Damien Cronan
CTO, Fairfax Media

Download our full Fairfax Media white paper to explore how we improved publishing workflows for Australia's leading media brand.

Is headless right for you?

There's a lot to consider before deciding whether or not headless architecture is the best choice for your specific requirements, but our handy decision tree can help you tick some of the biggest boxes.



We'd always recommend getting tailored advice from an expert before committing to changing your site's architecture - why not reach out to us at hello@humanmade.com to chat about your options?

“

Headless architectures provide technology teams with unique methods and approaches to tackle challenges. From decoupled frontends to mobile applications, reusable content to centralized content stores, the REST API provides powerful capabilities to deliver innovative experiences. I created the WordPress REST API to allow WordPress to meet the standards demanded by modern web users.

This innovation is not without challenges, and headless approaches need to be carefully considered and balanced against concerns like gaps in functionality and accessibility. Navigating these pitfalls and deciding on which challenges to tackle takes careful consideration and guidance.

At Human Made, we’ve guided many organizations through the journey of evaluating, deciding, engineering, and launching complex sites using headless, decoupled, and hybrid approaches. Our years of experience, including building the REST API itself, mean we have the experience to know how to tackle these challenges head on.”

Ryan McCue
Director of Product, Altis

We’re always happy to chat about which architecture options may be right for your site.

Get in touch →