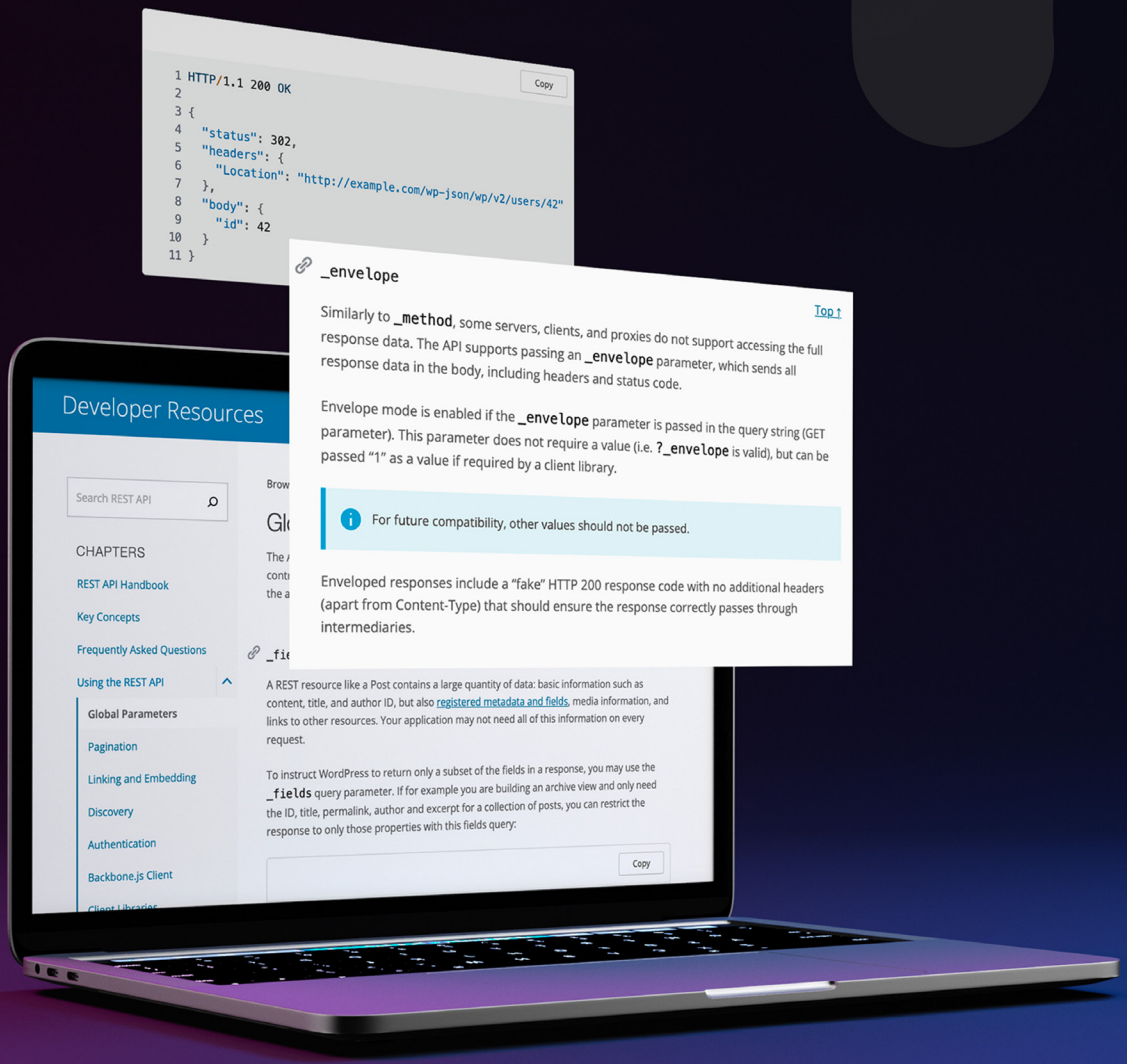
Human Made

The Human Made Guide to the WordPress REST API

Everything you need to know about the REST API, from the people who created it



Contents

Introduction

The WordPress REST API, created by Human Made's Ryan McCue, facilitated a major evolution, exposing WordPress content and data as JSON via a standardised RESTful API. For the first time, data was unlocked, and an explosion in the number and complexity of WordPress integrations began.

The REST API allows for a more powerful 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.

Read on to find out more about the REST API, and as always, please do reach out to us at hello@humanmade.com if you've got any queries about anything RESTful.

What is a REST API?

POST, GET, PUT, DELETE:
Dive into what a REST API is, why it's RESTful, and what makes it open.

What is REST?

Representational **S**tate **T**ransfer **(REST)** is a software architectural style for Application Programming Interfaces (APIs) that consists of guidelines and best practices for creating scalable web services.

REST uses simple HTTP to make calls between machines, which happen via a request/response mechanism between the server and the client. For example, a client, let's say an Android application, makes a request for the most recent posts from the website. The server knows how to interpret this request, through REST, and satisfies the response by providing the most recent posts in a format understood by the client.

REST requests interact with the resources in your application (e.g. a Post or Page). These interactions are typically Creating, Reading, Updating or Deleting (CRUD). Combined with HTTP, REST requests are formed using four verbs:

- POST: Create a resource
- GET: Retrieve a resource
- PUT: Update a resource
- DELETE: Delete a resource

The data retrieved is supplied in a machine-readable format, often JSON in today's web applications.

What makes an API RESTful?

An API must have the following architectural features to be considered RESTful:

- **Client-server:** the client is separated from the server. This means that clients are not concerned with data storage and servers are not concerned with display. This ensures that data is portable and can be reused in multiple clients, and servers are simpler and more scalable.
- **Cacheable:** clients can, and should, cache responses to improve performance, and avoid the server with every request.
- **Stateless:** the necessary state to handle the request is contained in the request itself, whether as part of the query parameters, URL, body, or headers.
- **Uniform interface:** information transferred via REST comes in a standardised form, creating a simplified, decoupled architecture.
- **Layered system:** the architecture is composed of hierarchical layers. Each component cannot "see" beyond its layer: a client cannot tell if it's connected to the server or to an intermediary.
- **Code-on-demand:** REST allows client functionality to be extended by transferring applets or scripts.

A separate but closely-related concept is hypermedia. Similar to how hyperlinks on the human-readable web enable discovering new sites and content, hypermedia allows a client to more fully discover a REST API without needing to know anything about the structure of the API.

Instead, the server provides whatever information the client needs to interact with it. This means that the client can interact with the server in complex ways without knowing anything about it beforehand.

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.

What is the WordPress 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.

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.

```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;
}
```

Authentication

For accessing private data or publishing new content, authentication allows users to establish who they are and whether they're authorised 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.

Why use the WordPress REST API?

From centralising your data and services to using WordPress as a module in a larger stack, discover the myriad uses for the WordPress REST API.

Create context-specific solutions

Over 43% of the world's websites use WordPress. These websites are PHP-based, with frontends built with the WordPress Theme system.

The API frees developers and allows them to use any technology that will solve a problem in their specific context. WordPress no longer has to be concerned with the frontend: it can deliver data to any frontend technology. A developer can take data from a WordPress website and display it using the technology of their choice.

Reusable, portable content

The content entered into WordPress isn't limited to being displayed on a WordPress website. A REST API-powered website has content which is infinitely portable, meaning your content authors only need to enter data in one place. Once it has been authored and published in WordPress, it's available via the API and can be delivered to websites, web apps, and mobile and desktop apps.

Separation of concerns

In traditional WordPress development where the frontend and backend are tightly coupled, frontend developers need to be familiar with at least some aspects of WordPress. This makes it difficult to hire and work with purely frontend developers.

This is no longer a problem in a decoupled environment; different teams can work on different project elements while having access to the same data. A backend team can work on WordPress and the database, a team can build the frontend in JavaScript or another web technology, and you could have an iOS team and an Android team.

Your JavaScript developer no longer needs to learn PHP to work with WordPress, and your WordPress developer no longer needs to tinker with JavaScript. This widens the pool of development talent available to work on your website or application, and streamlines project management.

Develop with live data

When data is exposed through the REST API, it can be used in a development environment. Content can be added to the CMS and is available to developers whether they are working on the frontend, the admin, or any applications.

Familiar backend for authors and publishers

One of the reasons WordPress has been so successful is because it provides an easy-to-understand interface for non-technical users.

With the REST API, you don't have to decide between using your frontend technology of choice and giving your authors the admin interface they want: authors and editors can work in the WordPress admin and the data is delivered to the frontend by the API. It's advantageous to have an admin interface many authors are already familiar with, as it reduces the need for training and retraining, letting authors quickly start adding content.

Integrate WordPress as one part of a content authoring workflow

WordPress may only be suitable for one aspect of your website or application. The REST API allows you to use WordPress for only those elements that it is suitable for. This allows WordPress to form just one module in a larger stack, making it more available to the wider web development community for smaller, specific tasks.

WordPress as a central repository

Much of the web is API-driven, with websites and services aggregating data. The REST API makes it possible for WordPress to be the central place that brings all of this data together.

This means that all of your services and data can be centralised, while providing your authors with a straightforward, familiar interface. This also provides a standard platform for further functionality with WordPress' plugin system.

ustwo

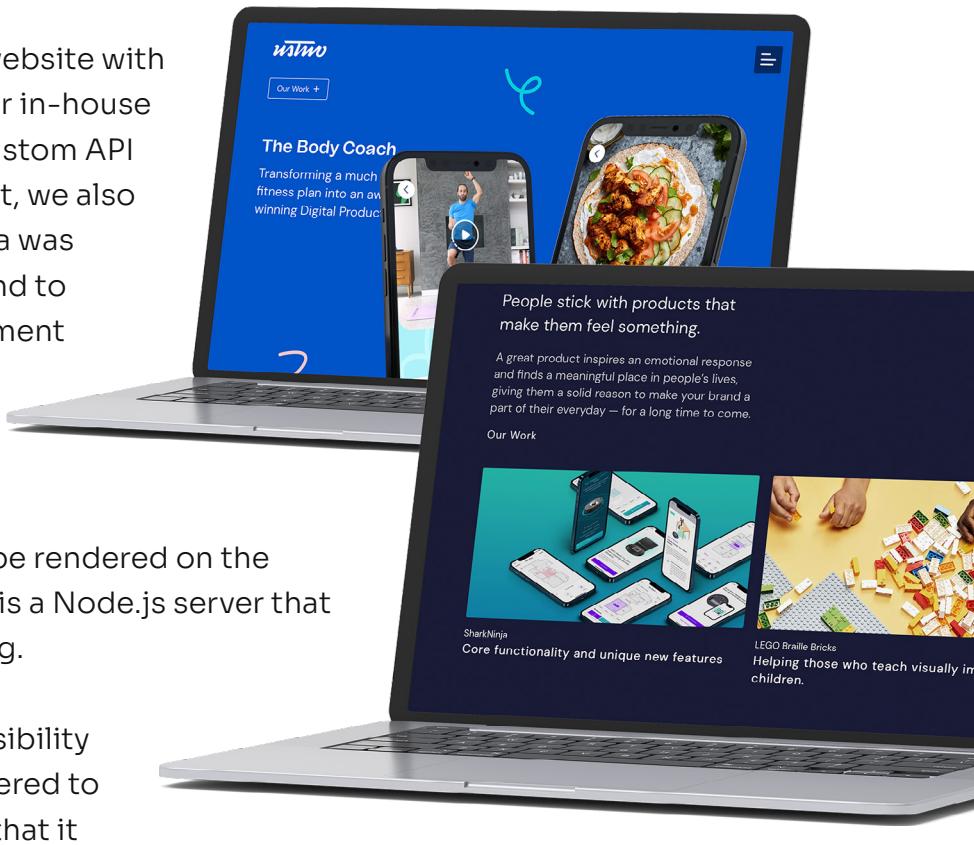
Case study

ustwo wanted a decoupled website with a WordPress backend and a frontend built with React.

Why WordPress & the REST API?

WordPress provides a straightforward interface that the company's global marketing team could use to author content. WordPress' free and open source nature meant the ustwo design team could focus entirely on the frontend, without wasting time rolling their own CMS.

The build



ustwo wanted a decoupled website with a React frontend built by their in-house team. As well as creating a custom API for delivering ustwo's content, we also had to ensure structured data was delivered to the front-end, and to enable progressive enhancement across platforms.

React was used because it allows for isomorphic rendering, where pages can be rendered on the server or by the client. There is a Node.js server that enables server-side rendering.

It was Human Made's responsibility to ensure that data was delivered to the frontend via the API and that it was available in a structured, reusable format.

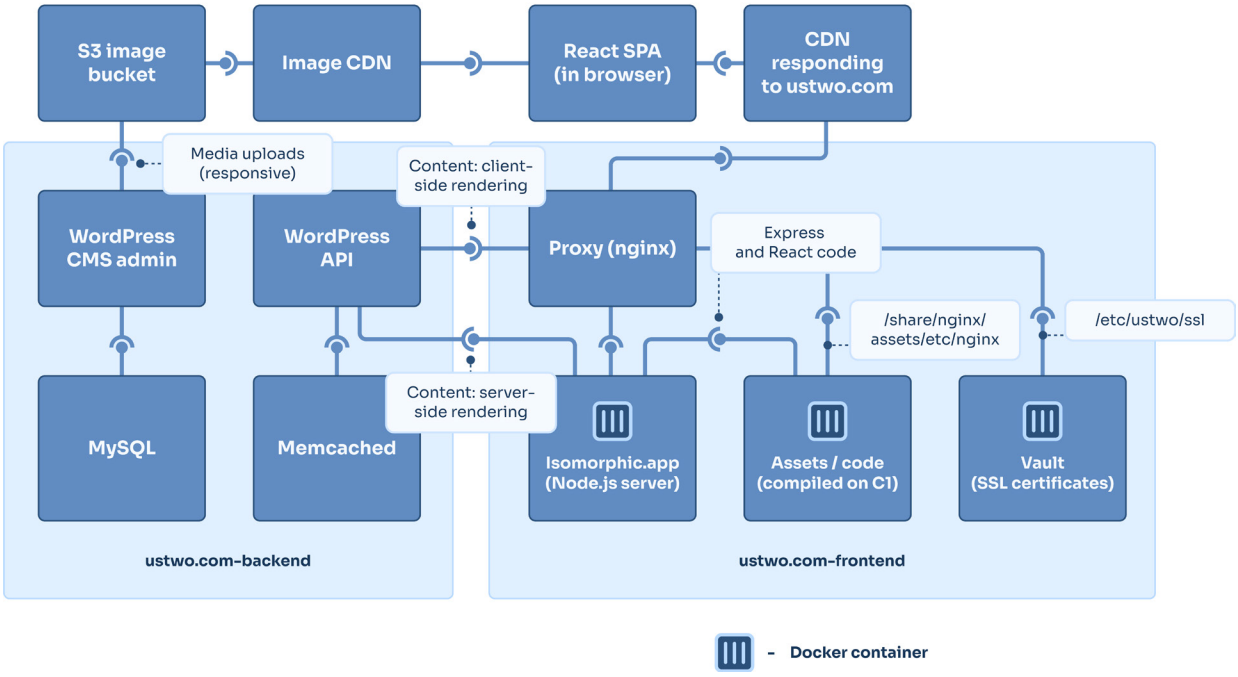
ustwo

Case study

We used our experience leading the WordPress REST API project to build a custom API that was tailored to the client's needs. This delivered all of the data from ustwo's custom post types.

On the WordPress side, a custom page builder plugin gets authors to enter content in a modular fashion. This ensures that the content is modular and portable to different contexts. The infrastructure for the REST API is used along with a bespoke API comprising custom endpoints that deliver content in JSON format to the frontend.

ustwo.com architecture



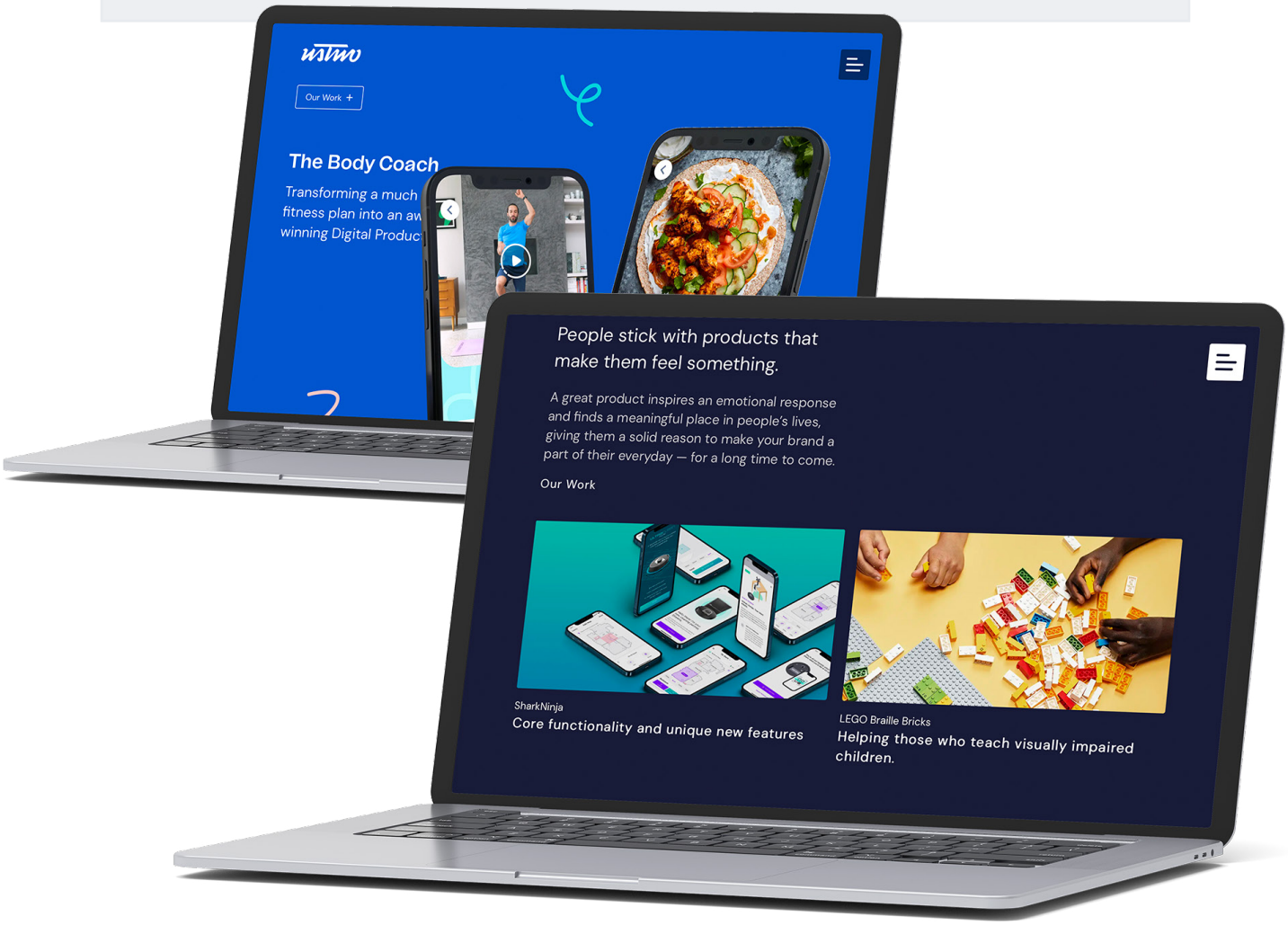
By collaborating effectively with ustwo's in-house design and frontend talent, we were able to focus solely on WordPress as a CMS for data collection and delivery, culminating in the production of ustwo's new WordPress-powered website that showcases their specialisms.

ustwo

Case study

We chose WordPress as we wanted to have an established open source CMS so that we can be confident that we'll never be left without support or ability to change. To fulfil our design ambitions we decided to build our frontend as a single-page application, which was made possible with the WP-API."

Daniel Demmel
Fullstack web developer, ustwo



How the REST API changed WordPress development

With the REST API, WordPress evolved from being a web development tool used in isolation to one module available in a web developer's toolkit.

WordPress as part of a larger stack

WordPress has a familiar and easy to use user interface for managing and publishing content. With the REST API, you can provide this interface to your authors without compromising on the rest of your stack.

As a single module in a larger stack, WordPress can also be used outside of its traditional community. The REST API allows developers to create websites and applications in any language without having to roll their own CMS.

New project approaches

As a knock-on effect, the separation of concerns that come with a REST API project can allow project management to be approached in a new way: developers can independently focus on different aspects of the website or application, working with live data retrieved using the API.

The role of admins has changed, too: developers can create funnelled administration experiences that focus on a particular user doing particular actions. These focused admins will remove the clutter and empower the user to do exactly what they need to do.

The enhancement of built-in WordPress functionality

The REST API makes it easier for developers to enhance functionality in the WordPress admin. Developers can create client-side features in the admin that are more advanced and more performant than can be achieved with PHP.

The separation of concerns means that frontend products that retrieve data from the API will not need to be GPL. However, there hasn't been a vast increase in API-powered themes due to the challenges of rebuilding native WordPress functionality on the frontend.

Gutenberg & the REST API: powering advanced applications

Gutenberg was the first project in WordPress core using the REST API, and it changed the way developers build on WordPress. For the first time, the REST API was being used to communicate data between the server and a JavaScript powered frontend.

> *Gutenberg's rapid development speed showed the power of the REST API, and how it changed the way developers build with WordPress. The rich, block-based data that Gutenberg provides pushed the developer experience of WordPress forward, while also providing a fantastic experience for users."*

Ryan McCue
REST API co-lead

Challenges presented by the REST API

The introduction of the REST API marked a new era in WordPress development. As with all technologies, there are a few drawbacks alongside the plus points.

Loss of core functionality

A REST API-driven website loses frontend features that are linked to the WordPress Theme system, like menu management and post previews.

Frontend developers need to take responsibility for reimplementing features that come for free with WordPress: if they are not rebuilt, users must do without them.

When writing project specifications for an API-driven project, it's necessary to be specific about the features the client needs - avoid assuming that because they are in WordPress they are available.

To solve this problem, we anticipate the emergence of REST API base themes that rebuild WordPress features on the frontend. These boilerplate themes will be written in different languages and will provide a starting point for frontend developers to build on.

Disempowers WordPress site builders

WordPress' strength, alongside its user-friendliness, is that it makes it very easy to set up a website.

Through WordPress, many people gain experience of PHP, CSS, and HTML, thereby gaining confidence to make changes to the frontend of their website. However, the REST API completely decouples the frontend from the backend, disempowering those users, and making the frontend only editable by developers.

For this reason, REST didn't cause a major disruption to the WordPress theme market. Instead, the REST API is of most significance to large custom builds and WordPress-based SaaS products.

The need for structured, portable data

A headless WordPress site requires data that can be used across multiple contexts, which means creating and storing it in a way that is completely frontend agnostic.

In the first instance, you may just be using data on a website, but you may want to make it available later to a native application. The focus here is on content management as opposed to web publishing. This data needs to be structured in a modular manner, separate to the CSS and HTML.

For this reason, REST API-driven sites will not use the WYSIWYG capability in TinyMCE for page layouts, instead using content structured by modular page builders.

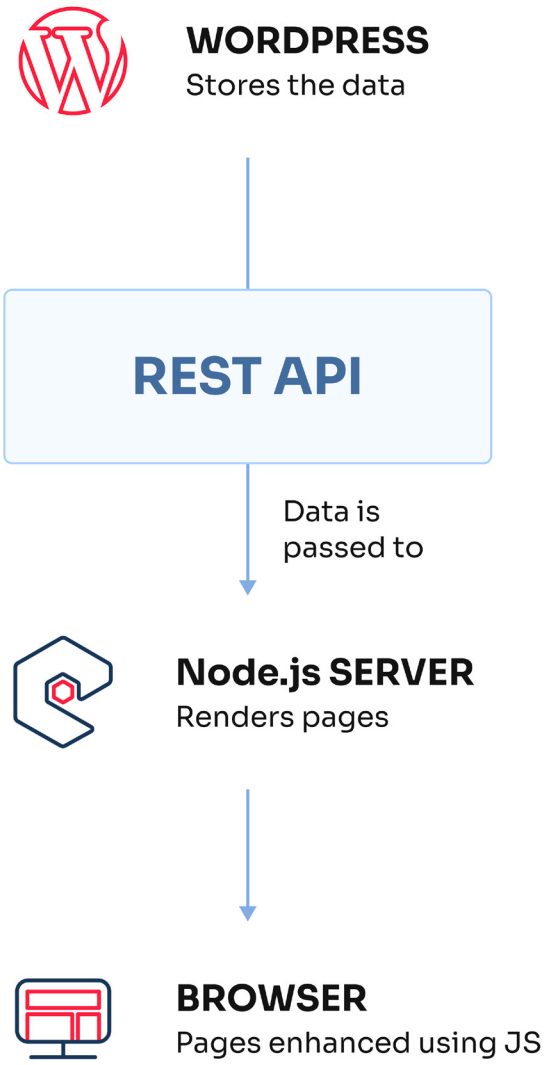
WordPress' commitment to backwards (and forwards) compatibility ensures that data produced by the API will continue to be readable and usable well into the future. This means that you can safely store it knowing that it will continue to be available by a well-supported API. In addition, the WordPress REST API is open, ensuring that your data can be moved out of your site using standard tools.

Dealing with progressive enhancement

In an increasingly JavaScript-driven world, progressive enhancement is a challenge that has to be addressed, particularly since some users have JavaScript disabled in their browser.

If content from a REST API-driven WordPress website is delivered to a JavaScript-powered frontend, these users will simply see a blank page. Developers need to address these issues to ensure that the web stays accessible.

One method is to render frontend templates on the server using a technology like Node.js, and then enhance the website on the frontend using client-side JavaScript. This setup, however, requires an additional server, and developers with the experience to implement it. Standalone services such as Vercel offer frontend Node.js rendering platforms, and many top-tier WordPress hosts now offer JavaScript server technology as part of their standard hosting.



“

Headless architectures provide technology teams with unique methods and approaches to tackle challenges. From decoupled frontends to mobile applications, reusable content to centralised 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 organisations 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 →