

TALKING TO 28% OF THE WEB

*An in-depth report and analysis
on the WordPress REST API*

Human
Made.

Contents

03

Executive Summary

05

The Headless CMS

10

What is a REST API?

18

*What is the
WordPress REST API?*

23

*Why Use the
WordPress REST API?*

30

*How the REST API
Will Change WordPress
Development*

33

*Challenges Presented
by the REST API*

40

Resources

About the Authors



Tom Willmot is the co-founder and CEO of *Human Made*. He has over 10 years experience bringing open source technology to enterprise. He lives in the UK with his wife and daughter.



Joe Hoyle is the co-founder and CTO of *Human Made*, where he leads the development team and the overall technical direction of the company. He is a member of the WordPress REST API development team

Contributors

Ryan McCue - WordPress REST API Team Co-Lead

Daniel Bachhuber - WordPress REST API Team

Siobhan McKeown - Writer & Editor

Michael Pick - Designer

Executive Summary

WordPress is an open-source content management system which is used to power millions of websites and blogs, and an increasing number of web applications.

[It currently powers more than 28% of the top 10 million websites on the Internet.](#) WordPress' usability, extensibility, and mature development community make it a popular choice for web projects of all sizes.

WordPress, like many other CMSes, is monolithic. It provides everything you need to run a website, and can be extended further with third-party plugins and themes. But on today's web, we are moving beyond the monolithic CMS. WordPress, along with others, is leading the charge to a future where the CMS acts as a central hub, consuming and aggregating content and data from other tools and services and in-turn exposing its own content and data via APIs.

The WordPress REST API is a huge step towards this future. Exposing WordPress content and data as JSON via a standardised RESTful API unlocks your data and will enable an explosion in the number and complexity of integrations.

By embracing the WordPress REST API, you can more easily:

- separate your frontend delivery from the CMS
- power multiple frontends from the same content (think a website, app, Apple News, etc.)
- and use WordPress as part of complex multi-service workflows (like pushing content to a separate service for translation before pulling those translations back into WordPress).

The potential implications for your business are far-reaching, particularly for large custom builds and applications.

WordPress provides content management and content capture, while making the data available to other frontend technologies. This permits engineering teams to work independently on discrete parts of a larger project, and allows for more stable third-party integrations.

With the REST API, WordPress stops being a web development tool used in isolation. **It is one module that is available in a web developer's toolkit;** a building block to be used in many kinds of applications.

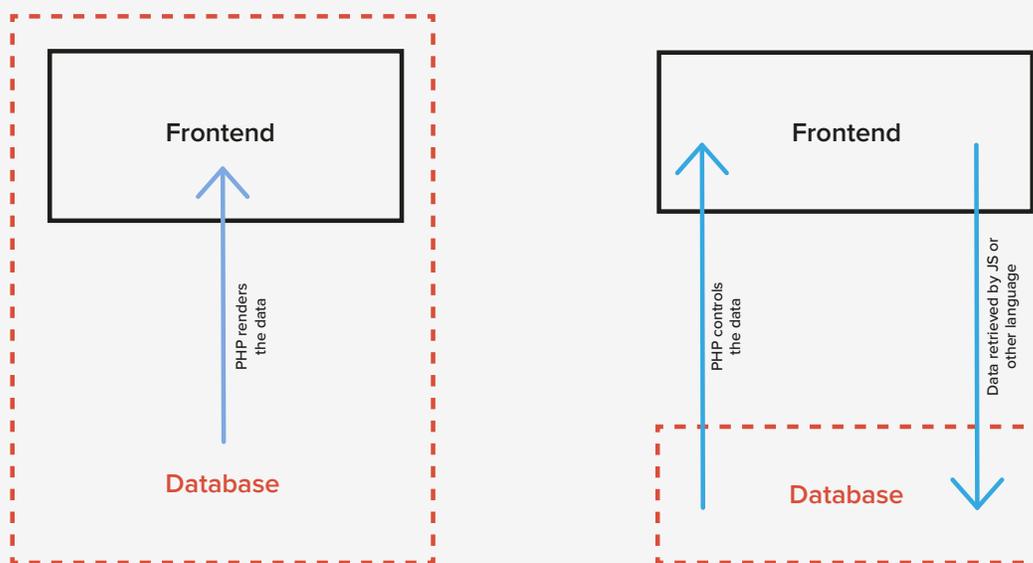
The potential implications for your business are far-reaching, particularly for large custom builds and applications.

The Headless CMS

We look at the difference between a traditional and a headless CMS, how a headless CMS slots into web development, and discuss some of the advantages of using a headless CMS for your web project.

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.

Monolithic CMS vs Headless CMS



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 you to replace it with the frontend technologies of your choice. 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.

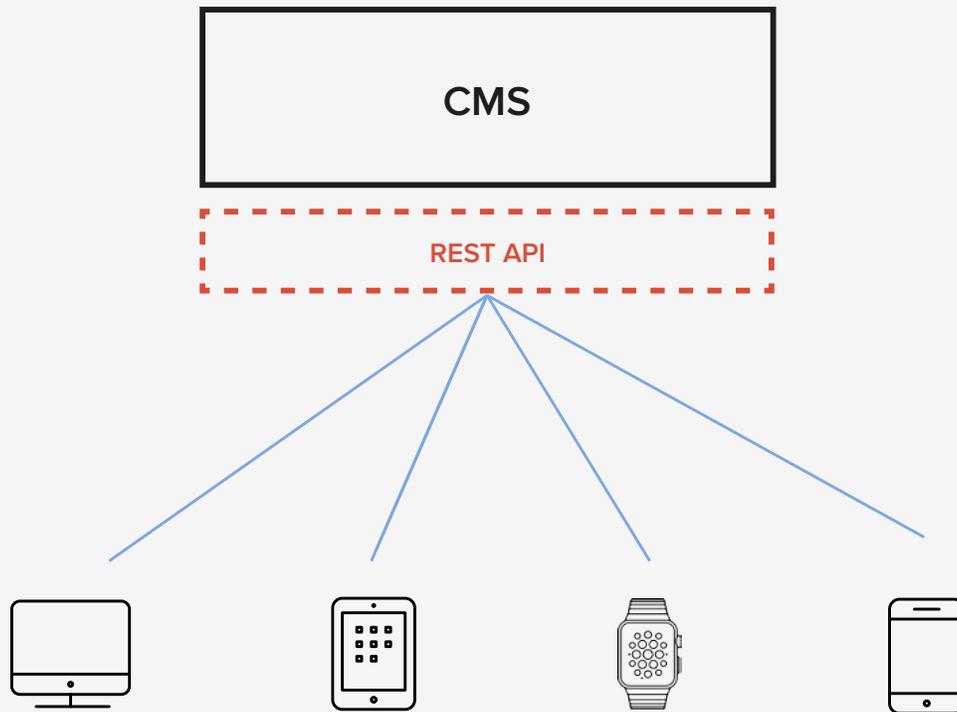
Decoupling content management from frontend display

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, Java, Ruby, and Swift 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 store the data, removing the need for complex migrations.

Fast, in-browser experiences, in real-time, without having to wait for PHP queries.

Fast, interactive experiences

When you use a headless CMS there are two components: the CMS itself and the frontend display. 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 queries to retrieve information from the database. 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.

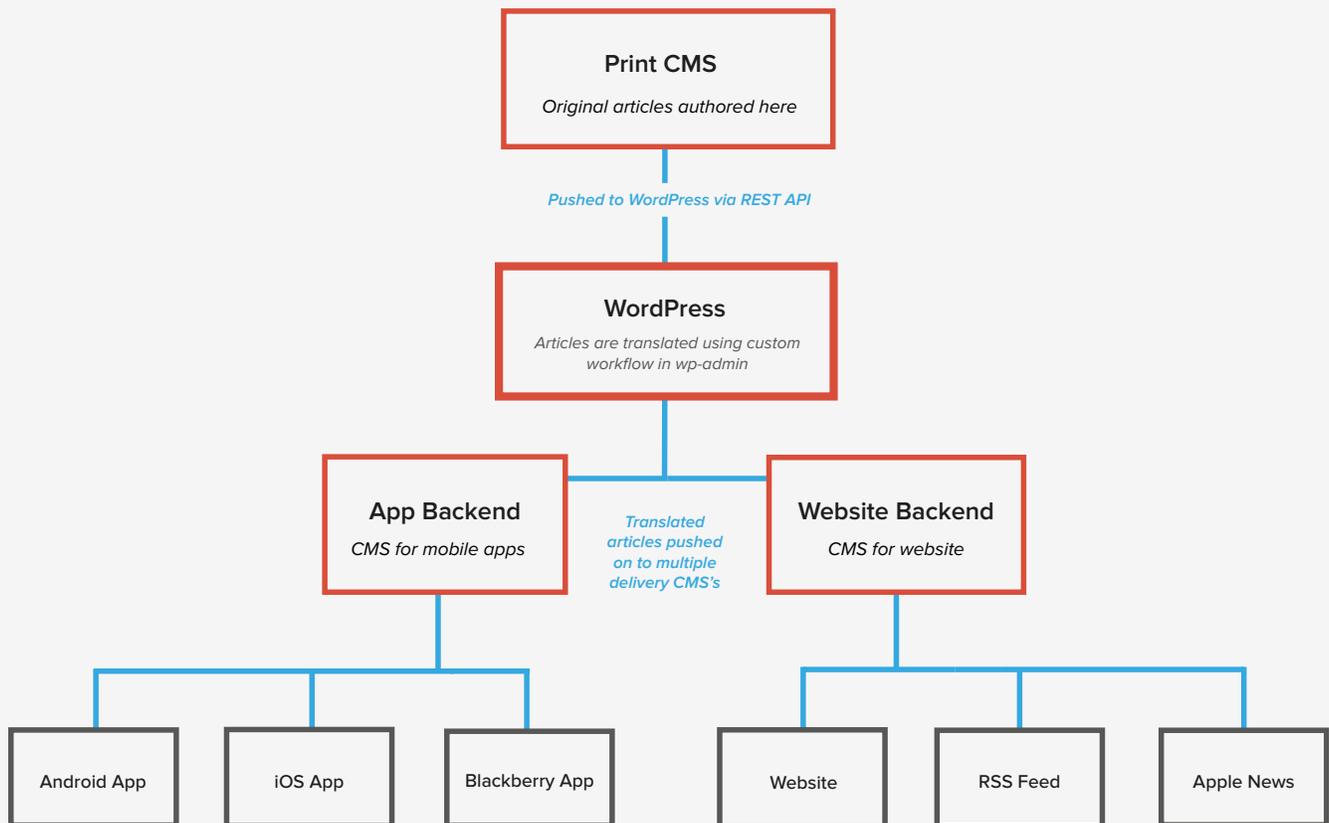


One content management system, multiple frontends

With a traditional, monolithic CMS, data is simply displayed by the CMS itself. **Data stored in a headless CMS is available for display in any context.** You may want to use it for a website now, but later you may decide to use the same data for a desktop or touch screen application. The stored data is always available via the API.

Multi-service content pipelines

A headless CMS can be used to store all of the data for one site or application, or it can **just 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. For example, it could be used just as a layer for translating content which is then pushed to another CMS.



What is a REST API?

We look at what the REST architectural style is, explore the elements that make an API RESTful, and consider some of the ways in which open APIs are changing the internet.

What is REST?

Representational State Transfer (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.

This happens 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 Reading, Creating, Updating, or Deleting. 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 modern web applications.

What makes an API RESTful?

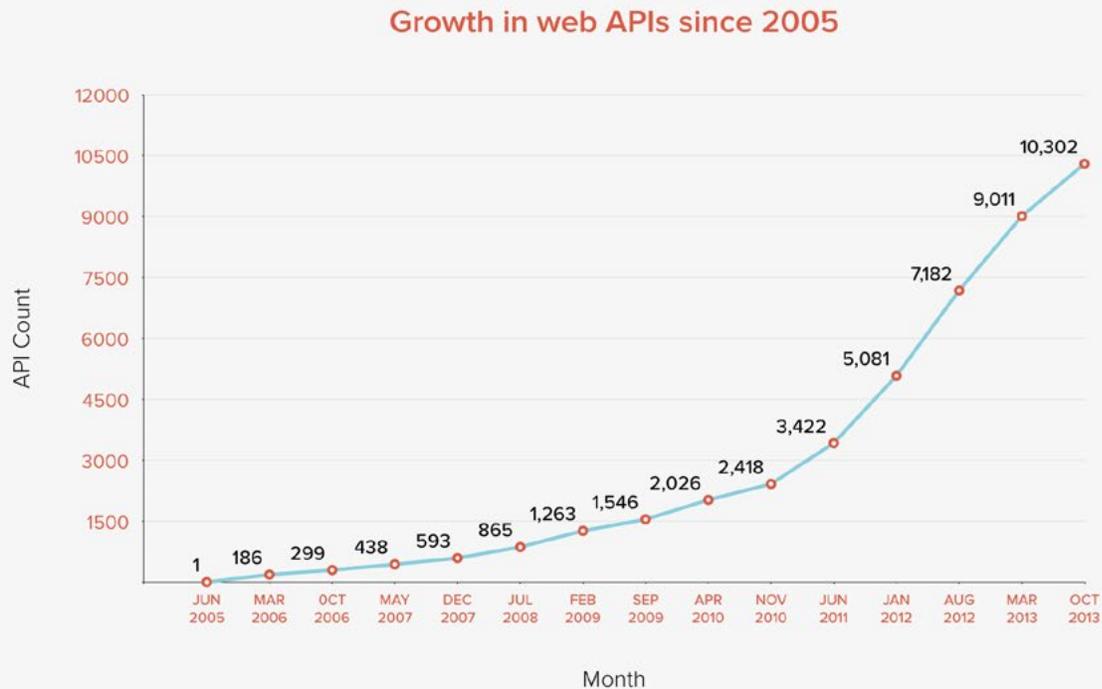
An API must have the following architectural features to be considered RESTful:

- **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.

A separate, but closely-related concept is hypermedia. Hypermedia allows a client to more fully discover a REST API without needing to know anything about the structure of the API. It’s similar to hyperlinks on the human-readable web (which enable discovering new sites and content).

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

REST was proposed by Roy Fielding in his 2000 dissertation [Architectural Styles and the Design of Network-based Software Architectures](#).



Source: Programmable Web

What is an open API?

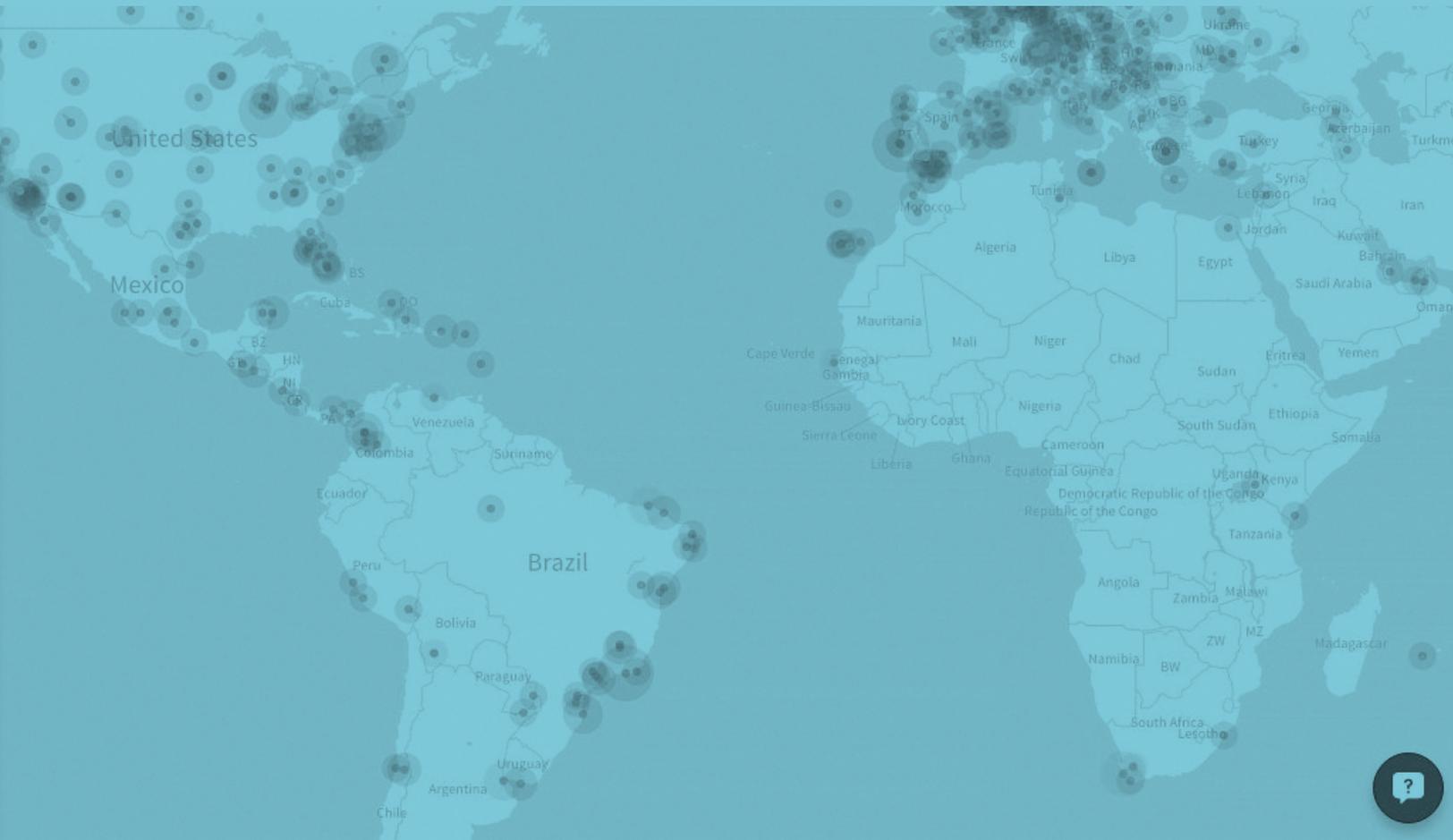
Open APIs are publicly available APIs that give developers access to proprietary software information that they can make use of in their own software and applications. REST is the ideal architecture for creating an Open API for the web because, by using HTTP, **it is built on the principles of the open 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 are transforming the internet. Developers can access data across services, creating applications that aggregate information from different providers. The impact of APIs cannot be overestimated; they are transforming the way businesses and services are run. For example:

- [Around 25% of annual revenue of the fundraising platform JustGiving is API-driven](#)
- In 2011, [Twitter reported that they had more than one million applications registered](#), with a number of entire companies built off the API
- Data feeds from the Skyscanner API are [used by startups like Hitlist, Go Euro, and Pintrips](#)
- Hilton is [making use of Uber's API to allow guests to book rides from the Hilton Honors App](#)

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.

The impact of APIs cannot be overestimated; they are transforming the way businesses and services are run.



Case Study: Nomadbase

Nomadbase is a service that facilitates remote working and the digital nomad way of life. Using public APIs, it aggregates data from social networking services to create a map of nomad locations across the world.

“ Our development team has a lot of experience with WordPress, and recognises its ability to provide a solid foundation to build an application on. Leveraging its users system for signups, the REST API for all external communication, and extremely common server requirements, we were able to get the prototype up and running in a matter of days, rather than weeks or months.



We used the REST API for not only the communication with the frontend application, but also inbound data coming from Facebook and Foursquare to collect user location data.

Joe Hoyle, Nomadbase

Why use WordPress and the REST API?

WordPress enabled the developers to get the first version of Nomadbase up and running quickly, providing both a stable central platform where data can be aggregated and an API for delivering the data to the frontend.

WordPress enabled the developers to get the first version of Nomadbase up and running quickly.

The build

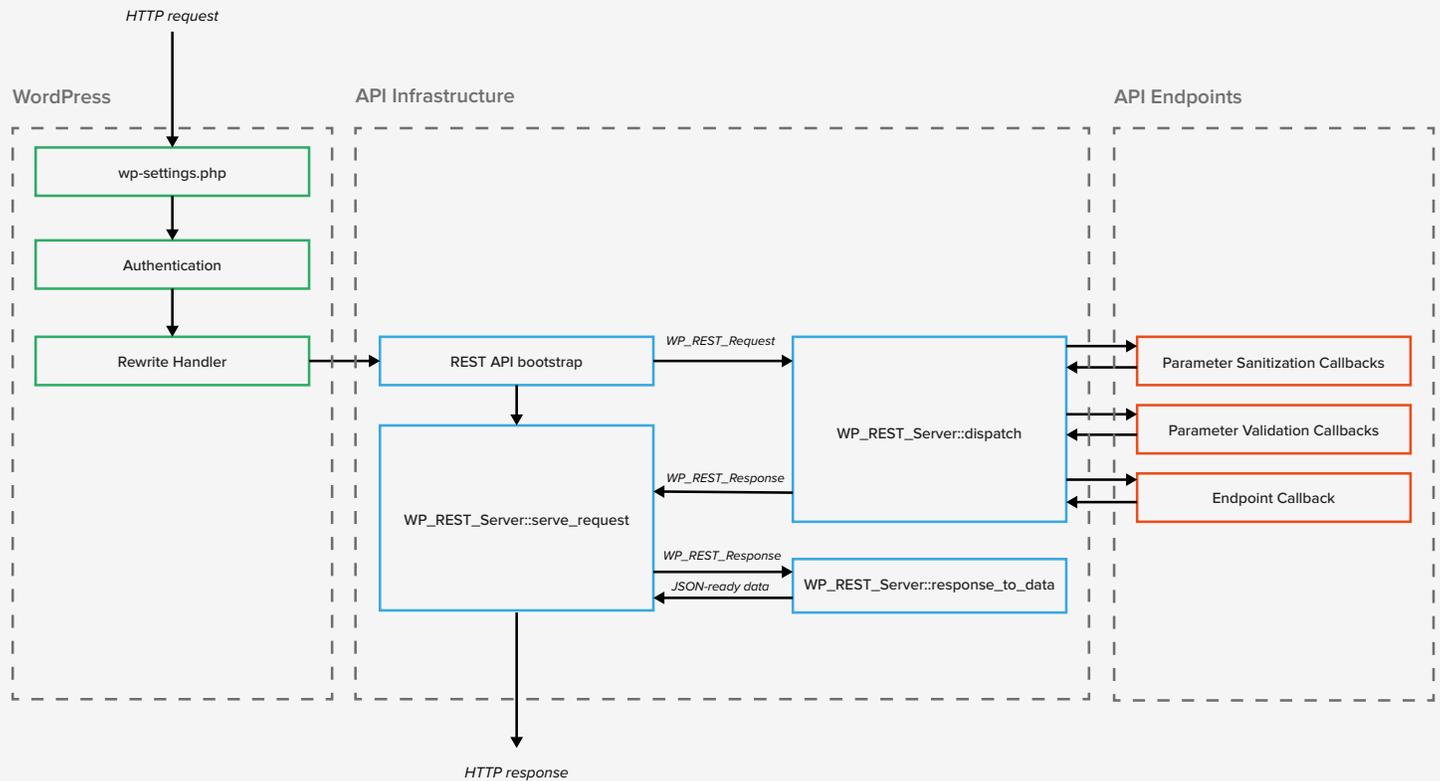
Nomadbase uses APIs to gather geodata from Facebook, Swarm, Twitter, Instagram, and Triplt. This data is stored in custom tables in the MySQL database. Data is then sent over the WordPress REST API to a React frontend and displayed in the browser. [Leaflet](#) is used to create overlays on the map.

When a new user signs up for Nomadbase, data is requested from five different social networks. This leads to significant background processing. To speed this up, wp-cron is replaced by a system tool called [Cavalcade](#), developed by Human Made for large-scale WordPress installations

The next step for Nomadbase is a React Native iOS application that reuses code from the browser app. As all the data and user actions that exist for Nomadbase are processed by the REST API, no additional backend work needs to be done to build the new application.

What is the WordPress REST API?

We look at the specifics of the WordPress REST API, including its infrastructure and endpoints, its authentication methods and the concept of hypermedia. We also introduce you to the team behind it.



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.

Infrastructure

WordPress 4.4 contained the infrastructure for the WordPress REST API. This can be thought of as a **construction kit for RESTful APIs** in WordPress: it enables developers to build their own REST APIs, handling things like API discovery, request routing, arguments, JSON serialisation/deserialisation, and response codes. If you are building a website, application, theme or plugin, you can use the API by adding your [own custom endpoints](#).

The REST API decouples the WordPress backend from the frontend, transforming it into a headless CMS.

Endpoints

Endpoints were added to the REST API in WordPress 4.7. They are **functions that are available through the API**: the places where developers can do something with the CMS, whether that's creating, retrieving, updating or deleting (CRUD) data. Initial data types added to the API include Posts, Categories, Pages, Comments, and Settings. A complete list can be [found in the REST API Handbook](#).

Authentication

A major challenge around building a REST API is [authentication](#): how does an API know that a user should be allowed to update content on a site, for example? Who should be allowed to retrieve data? Under what conditions? The WordPress REST API uses two forms of authentication:

- **Cookie** - this is the basic authentication method used in WordPress. When you log into your dashboard a cookie is set in your browser. This method is only viable when the current user is logged into WordPress and that user has the capability to perform the action requested.
- **OAuth** - this is the main authentication method used for external clients, i.e. any third-party site or application that wants to interact with the API. With OAuth, logged in users can authorise clients to act on their behalf. Clients are issued with OAuth tokens so they can interact with the API. The REST API uses OAuth 1.0a so that it can be used by all WordPress websites; OAuth 2.0 requires HTTPS but WordPress does not.

In addition, there is a Basic Authentication method for external clients. However, this is only recommended for development environments as it involves passing your username and password on every request, and giving your credentials to clients.

Who should be allowed to update content on a site or retrieve data, and under what conditions?

Team

The WordPress REST API has contributions from 72 developers. However, the team has had four core members:



Ryan McCue (Human Made)
Co-Lead of the REST API



Rachel Baker (Wirecutter)
Co-Lead of the REST API



Joe Hoyle (Human Made)



Daniel Bachhuber (Hand Built)

Why use the WordPress REST API?

We explore some of the reasons that you may want to use a headless WordPress with the REST API in your own projects, drawing out specific technical and project management-related benefits.

The WordPress REST API allows you to use WordPress as a headless CMS. Developers can interact with WordPress independently of the theme system, using it only as a data storage and delivery platform. There are many benefits to doing this.

Create context-specific solutions

Around 25% of all 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 just deliver data to any frontend technology. A developer can take data from a WordPress website and display it using the technology of their choice, whether that's for a website, Android application, iOS application or whatever context the data is needed.

Reusable, portable content

The content entered into WordPress is no longer limited to being displayed on a WordPress website. A REST API-powered website has **content which is infinitely portable.** Your content authors only need to enter data in one place. Once it has been authored and published in WordPress, it's now available via the API and can be delivered to websites, web applications, and mobile and desktop applications.

The API frees developers and allows them to use any technology that will solve a problem in their specific context.

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. In a decoupled environment this is no longer a problem. **Different teams can work on different parts of a project 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, you can 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.

Familiar backend for authors and publishers

One of the reasons WordPress has been so successful is that 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. You have the advantage of providing **an admin interface which many authors will already be familiar with**, reducing the need for training and re-training, and letting authors quickly start adding content.

Your JavaScript developer no longer needs to learn PHP to work with WordPress, and your WordPress developer no longer needs to tinker with JavaScript.

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 just those elements that it is suitable for. The New York Times, for example, [uses WordPress for its live coverage platform](#): data is received via WebSocket from a custom-built service, rendered in React, and served to the frontend via the WordPress REST API. In August 2015, [the paper even added Slack to its publishing workflow](#). This makes WordPress **just one module in a larger stack**, making it more available to the wider web development community for smaller, specific tasks.

All of your services and data can be centralised while providing your authors with a familiar, straightforward interface.

WordPress as a central repository

The web is increasingly 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 interface that they are familiar with. This also provides a standard platform for further functionality with WordPress' plugin system.

Develop with live data

When data is exposed through the REST API it can be used by developers in their 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.



Case Study: ustwo

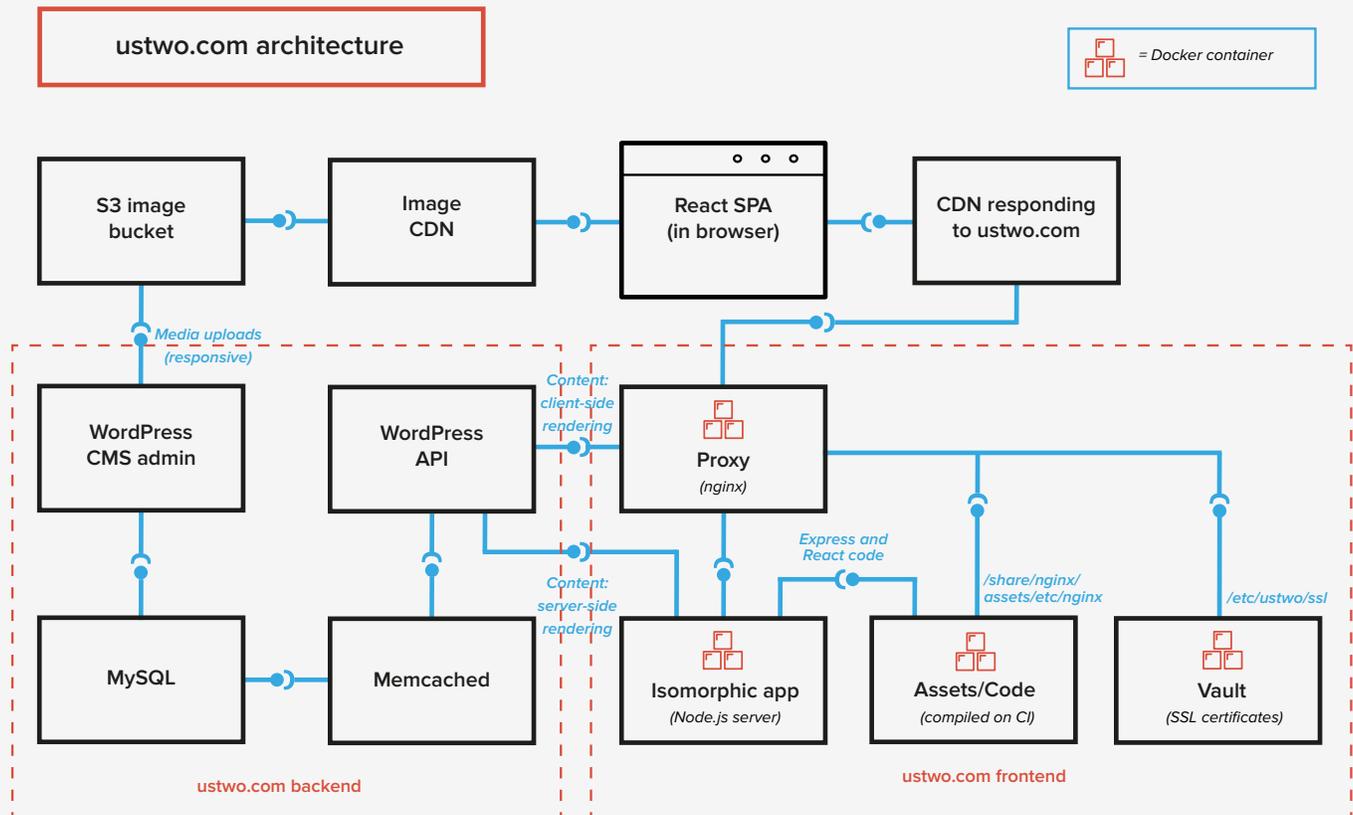
ustwo wanted a decoupled website with a WordPress backend and a frontend built with React.

“ 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 emerging WP-API.



Daniel Demmel, ustwo



Source: *ustwo*

The build

The [ustwo](#) website is a single-page application: the frontend is built using React and WordPress manages the content. React was used because it allows for isomorphic rendering (pages can be rendered on the server or by the client). There is a Node.js server that enables server-side rendering.

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.

How the REST API will change WordPress development

We look at some of the ways that the REST API will change WordPress development and the impact that this will have both for WordPress-based websites and applications, and for the people creating them.

WordPress as part of a larger stack

WordPress has a familiar and easy to use user interface which authors want to use 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.

Developers will independently focus on different aspects of the website or application, working with live data retrieved using the API.

New approaches to project management

The separation of concerns that come with a REST API project mean approaching project management in a new way. Developers will independently focus on different aspects of the website or application, working with live data retrieved using the API.

The WordPress developer as a backend specialist

There will be an increase in the number of WordPress developers who are backend specialists, focusing on the admin screens and the database, while leaving the frontend layer to frontend developers.

Permeation of WordPress outside of PHP communities

As a single module in a larger stack, WordPress will be used increasingly 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.

The emergence of funnelled, role-based admins

The REST API allows developers to create funnelled administration experiences that focus on a particular user doing particular actions. These focused admins will remove clutter and empower the user to do exactly what they need to do.

Focused admins will remove 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.

Explorations in non-GPL products

The absolute separation of concerns means that frontend products that retrieve data from the API will not need to be GPL. It's unlikely, however, that we'll see a vast increase in API-powered themes due to the challenges of rebuilding native WordPress functionality on the frontend.

Challenges presented by the REST API

We explore the challenges that will be brought about by the REST API and discuss some of the ways that these challenges will be addressed both in individual projects and by the wider WordPress community.

The introduction of the REST API marks a new era in WordPress development. Not all the ways that the REST API will change WordPress are clear, but some challenges are already emerging.

The REST API will be of most significance to large custom builds and WordPress-based SaaS products.

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 re-implementing 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 will become necessary to be very specific about the features that the client needs and not just assume 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

In addition to its ease of use, WordPress' strength is that it is easy to set up a website. Through WordPress, many people gain experience of PHP, CSS, and HTML, gaining confidence to make changes to the frontend of their website. The REST API completely decouples the frontend from the backend, disempowering those users, and making the frontend only editable by developers.

For this reason, it is unlikely that we will see a major disruption to the WordPress theme market. Instead, the REST API will be of most significance to large custom builds and WordPress-based SaaS products.

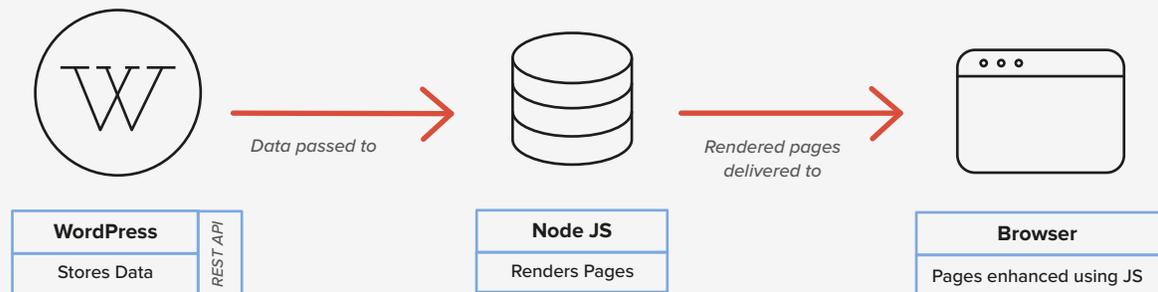
The necessity for structured, portable data

A headless WordPress requires data that can be used across multiple contexts. This 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 rely on the WYSIWYG capability in TinyMCE for page layouts, instead using content structured by [modular page builders](#).

Data needs to be structured in a modular manner, separate to the CSS and HTML.

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 through 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.

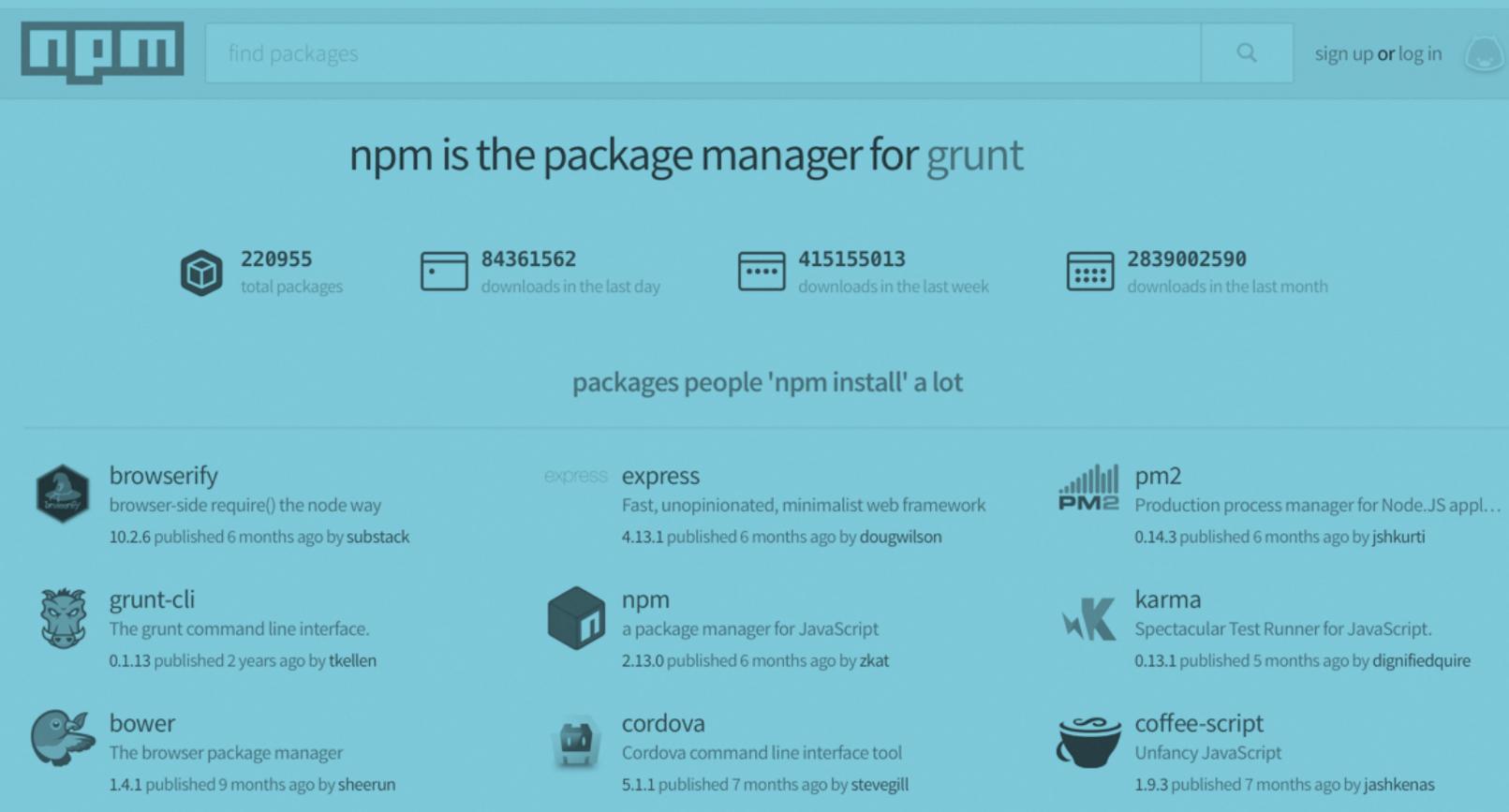


Dealing with progressive enhancement

In an increasingly JavaScript-driven world, progressive enhancement is a challenge that has to be addressed. Some people have JavaScript disabled in their browser, either because they use assistance technologies, because of personal preference, or because the organisation they work for requires it to be turned off. If content from a REST API driven WordPress website is delivered to a JS-powered frontend, these people 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.

One method is to render frontend templates on the server using Node.js, and enhance the website frontend using client-side JavaScript.



The screenshot shows the npm website interface. At the top, there is a search bar with the text "find packages" and a search icon. To the right of the search bar are links for "sign up or log in" and a user profile icon. Below the search bar, the main heading reads "npm is the package manager for grunt". Underneath this heading, there are four statistics cards:

- 220955** total packages (represented by a cube icon)
- 84361562** downloads in the last day (represented by a calendar icon)
- 415155013** downloads in the last week (represented by a calendar icon with dots)
- 2839002590** downloads in the last month (represented by a calendar icon with dots)

Below the statistics, there is a section titled "packages people 'npm install' a lot". This section displays a grid of popular packages with their respective icons, names, descriptions, and version information:

- browserify**: browser-side require() the node way. 10.2.6 published 6 months ago by substack.
- express**: Fast, unopinionated, minimalist web framework. 4.13.1 published 6 months ago by dougwilson.
- pm2**: Production process manager for Node.JS appl... 0.14.3 published 6 months ago by jshkurti.
- grunt-cli**: The grunt command line interface. 0.1.13 published 2 years ago by tkellen.
- npm**: a package manager for JavaScript. 2.13.0 published 6 months ago by zkat.
- karma**: Spectacular Test Runner for JavaScript. 0.13.1 published 5 months ago by dignifiedquire.
- bower**: The browser package manager. 1.4.1 published 9 months ago by sheerun.
- cordova**: Cordova command line interface tool. 5.1.1 published 7 months ago by stevegill.
- coffee-script**: Unfancy JavaScript. 1.9.3 published 7 months ago by jashkenas.

Case Study: npm

[npm](#) wanted to use the REST API to deliver custom brochure pages and upsell boxes on their website.

“ Security was of paramount importance to us and as a result, we needed a headless WP instance hosted under the same constraints as in our production environment. This meant reliance on the WordPress API to deliver authored content via JSON to our build, giving us the ability to parse and display CMS-generated content without having to grant access to any non-whitelisted IP Addresses.



Nick Cawthon, npm

Why use WordPress and the REST API?

npm wanted to use WordPress as a central repository for their documentation and their product pages. Its straightforward interface means that content authors can easily add data, which is delivered to the client in JSON format.

The build

The npm website has a WordPress backend and admin. A bespoke API built of custom endpoints serves content in JSON format to a Node.js server. This renders the final HTML and sends it to the browser where Handlebars renders the templates. The API doesn't just send the data: it sends rendered HTML along with scripts and stylesheets. This is cached by the Node.js server so that the website stays up even if WordPress is unavailable. It also means that the website stays fast without having to expend effort scaling the database.

Some customisations recreate the post preview feature of WordPress. Parts of the CSS templates and handlebars frontend are used to create a basic WordPress theme which authors use to preview posts before they are published and pushed to the frontend.

The website stays fast without having to expend effort scaling the database.

Resources

Resources

The WordPress REST API

- [Official website and documentation](https://developer.wordpress.org/rest-api/) (https://developer.wordpress.org/rest-api/)
- [WordPress core discussion about the REST API](https://hmn.md/wp-api/core/) (hmn.md/wp-api/core/)

Client Libraries

- [Node.js](https://hmn.md/wp-api/node/) (hmn.md/wp-api/node/)
- [Backbone.js](https://hmn.md/wp-api/backbone/) (hmn.md/wp-api/backbone/)
- [AngularJS](https://hmn.md/wp-api/angular/) (hmn.md/wp-api/angular/)
- [PHP Client](https://hmn.md/wp-api/php-client/) (hmn.md/wp-api/php-client/)
- [C#](https://hmn.md/wp-api/c/) (hmn.md/wp-api/c/)

Resources

Authentication

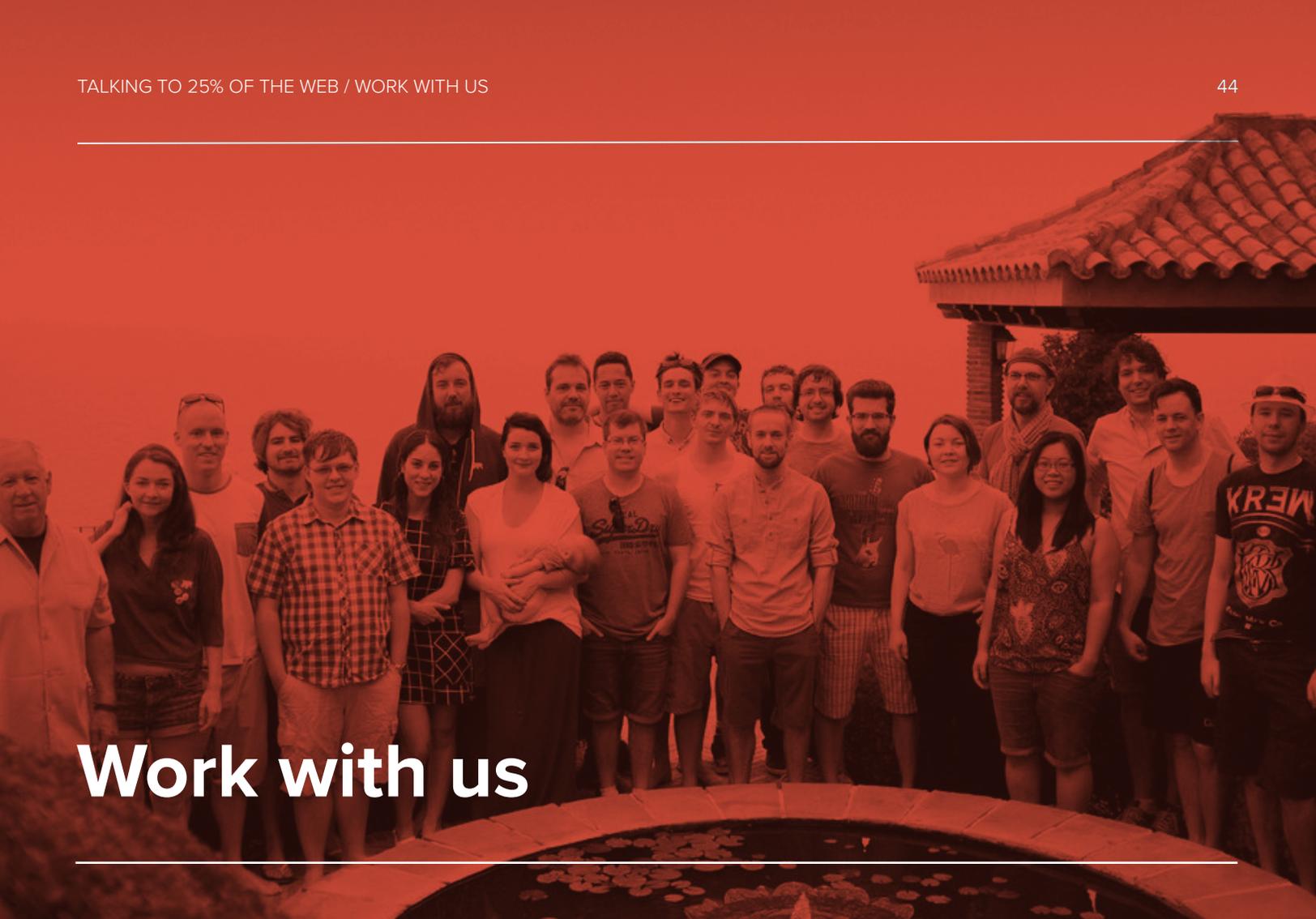
- [OAuth](https://hmn.md/wp-api/oauth/) (hmn.md/wp-api/oauth/)
- [Basic Authentication](https://hmn.md/wp-api/basic-auth/) (hmn.md/wp-api/basic-auth/)

Tools

- [WP-CLI client](https://hmn.md/wp-api/cli/) (hmn.md/wp-api/cli/)
- [API Console](https://hmn.md/wp-api/console/) (hmn.md/wp-api/console/)
- [WP JSON API Connect](https://hmn.md/wp-api/connect/) (hmn.md/wp-api/connect/)

Other resources

- [Picard React theme](https://hmn.md/wp-api/picard/) (hmn.md/wp-api/picard/)
- [Feeling Restful theme](https://hmn.md/wp-api/feeling-restful/) (hmn.md/wp-api/feeling-restful/)
- [ustwo.com frontend](https://hmn.md/wp-api/ustwo/) (hmn.md/wp-api/ustwo/)



Work with us

Human Made is an enterprise-level WordPress development firm, based in the UK but with employees and clients worldwide.

Our clients include NewsUK, AirBnB, Skype, and Yell. We're the people behind Nomadbase and Happytables. Our developers have led the development of the WordPress REST API and we're already making use of it in our own products and in client projects.

Want to use the WordPress REST API in your project? Contact Human Made for:

- enterprise-level development
- bespoke training
- consultancy
- hosting

Email us hello@hmn.md or give us a call at +44 (0) 1629 628082