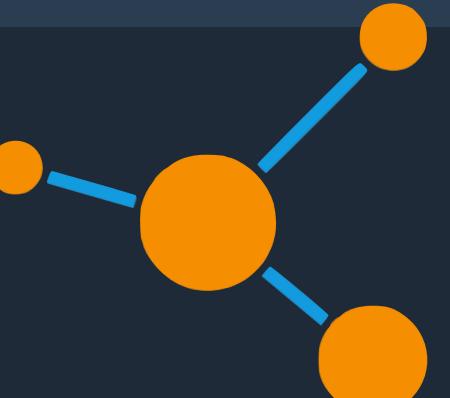


The WordPress API

Web applications for the REST of us!



@nagmay

BOSTON UNIV. CENTRAL
OUTBOUND TO BOSTON COLLEGE

WPCAMPUS

WHERE WORDPRESS MEETS HIGHER EDUCATION

WPCampus Online 2018 (*virtual conference*)

Tuesday, January 30, 2018

Our call for speakers is open until Tuesday, November 7, 2017.

Want to experience WordPress sessions, expertise, and networking with users across the world without the expense and hassle of travel? This one-day, session-filled event is for you!

Visit online.wpcampus.org for more information.

wpcampus.org

@wpcampusorg

#WPCampus

WPCampus 2018 (*in-person conference*)

Summer 2018

The hosting application will close Tuesday, October 31, 2017.

The main event for WPCampus is our annual in-person conference. We hold this event on college campuses and we'd love to bring our community to you.

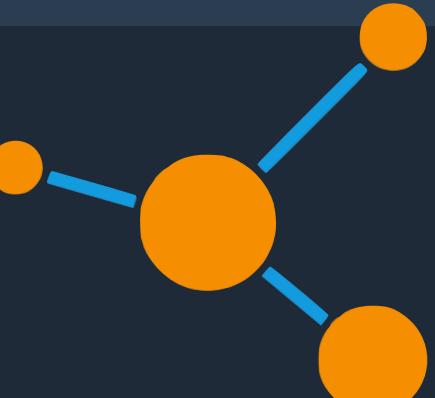
Visit wpcampus.org to learn more about the event and how to host.

What the what?

- Overview and history
- Pulling data

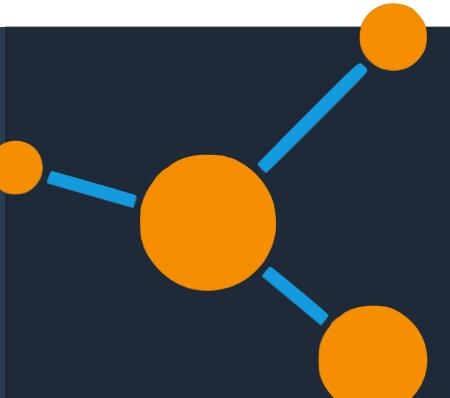
— Intermission: Let's talk about beer —

- Pushing data



Gabriel Nagmay
gabriel@nagmay.com

Web Analyst
Coffee Lover
Home Brewer



The WordPress API

@nagmay



Cascade

A small campus atmosphere in a diverse urban neighborhood in the heart of Portland.



Rock Creek

A spacious, 260-acre campus in the high-tech corridor near Hillsboro.



Southeast

PCC's newest campus, in the growing and diverse central eastside of Portland.



Sylvania

PCC's largest campus, on a wooded hillside between Tigard and Lake Oswego.

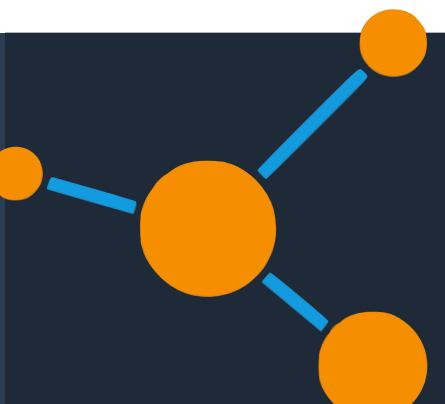


Centers

Centers are smaller facilities offering job training, specialized programs, and transfer courses. These centers, and many other locations throughout the community, make up PCC's Extended Learning Campus.

[CLIMB Center for Advancement](#)
[Downtown Center](#)
[Newberg Center](#)
[Swan Island Trades Center](#)

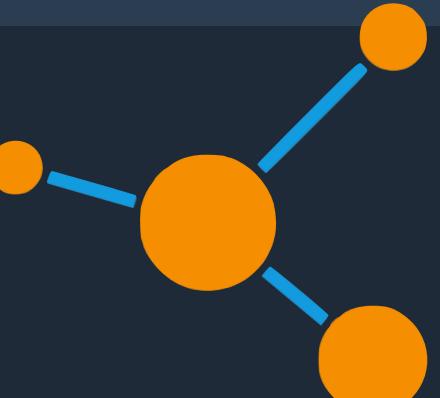
[Columbia County](#)
[Hillsboro Center](#)
[Portland Metropolitan \(PMWTC\)](#)
[Willow Creek Center](#)



The WordPress API

@nagmay

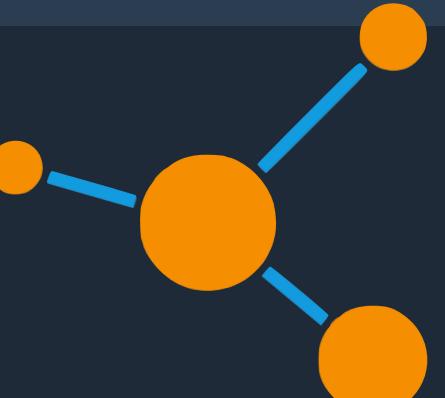
Application Protocol Interface



The WordPress API

@nagmay

REpresentational State Transfer



The WordPress API

@nagmay

HTTP Methods

POST .../posts



Create

GET .../posts



Retrieve

PUT .../posts

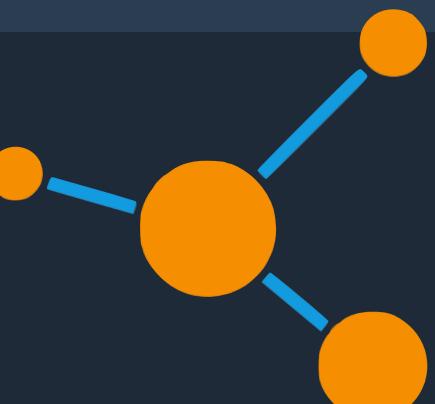


Update

DELETE .../posts



Delete



The WordPress API

@nagmay

JavaScript

Coldfusion

Python

Ruby

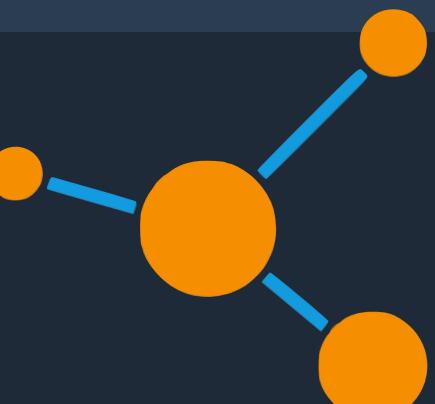
PHP

Java

C#

Bash

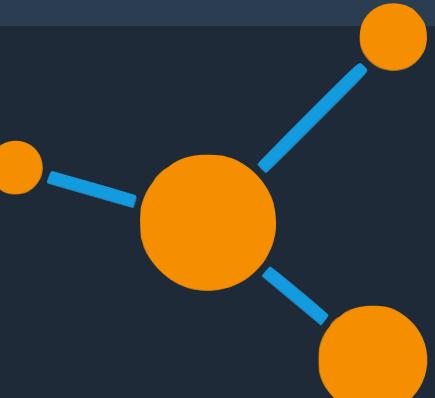
The WordPress API



@nagmay

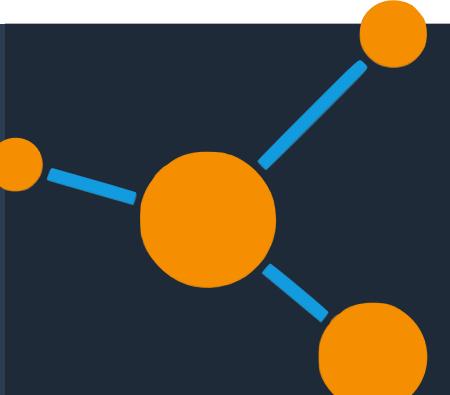
WP-API History

- 2013 GSOC project by Ryan McCue
- 2014 v1.0 released as WP-API plugin
- 2015 v2.0 developed
Infrastructure added to WP4.4
- 2016 Endpoints added to WP4.7



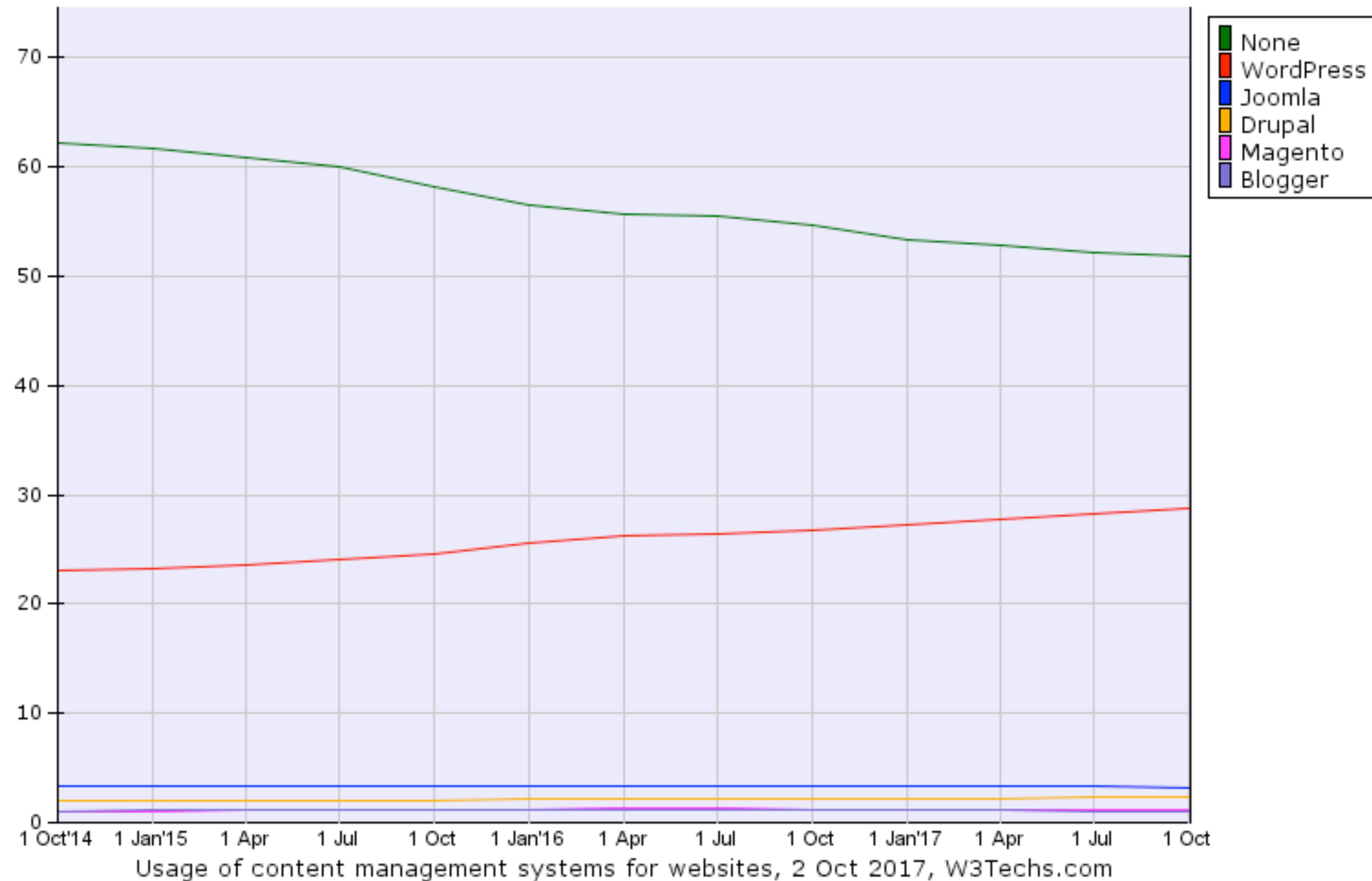


WORDPRESS



The WordPress API

@nagmay



The WordPress API

@nagmay

28.7% of the web

Now available to you via an API



The WordPress API

@nagmay

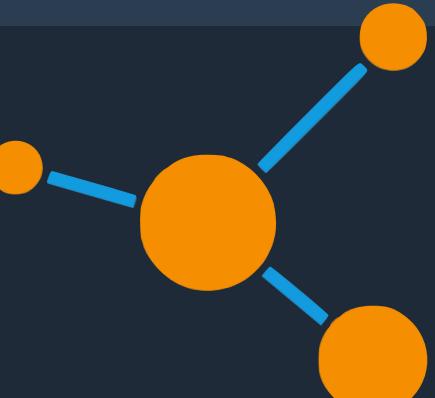
Tools

Examples: github.com/nagmay/WRK7

Web browser

Recommended

- WP REST API Handbook
- Postman: getpostman.com



news.pcc.edu/wp-json/

<https://news.pcc.edu/wp-json/>**JSON Raw Data Headers**

Save Copy

```
name: "News"
description: ""
url: "https://news.pcc.edu"
home: "https://news.pcc.edu"
gmt_offset: -7
timezone_string: "America/Los_Angeles"
namespaces:
  0: "oembed/1.0"
  1: "akismet/v1"
  2: "jetpack/v4"
  3: "wp-super-cache/v1"
  4: "wp/v2"
authentication: []
routes: {...}
_links: {...}
```

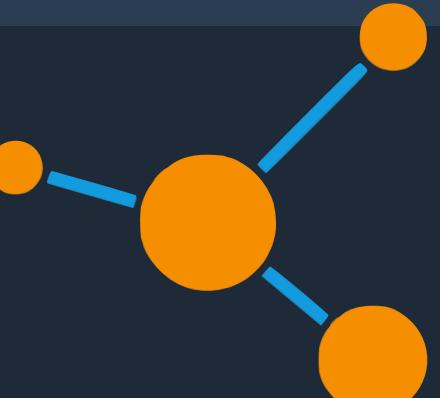


The WordPress API

@nagmay

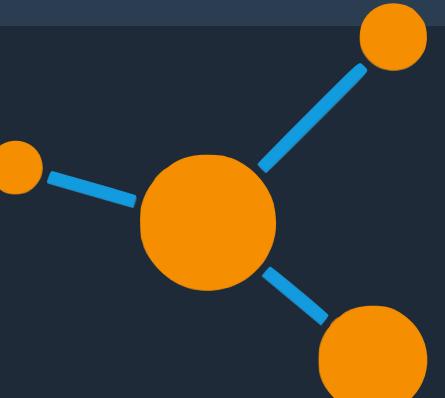
Routes

namespace:	"wp/v2"
routes:	
▶ /wp/v2:	Object
▶ /wp/v2/posts:	Object
▶ /wp/v2/posts/(?P<id>[\d]+):	Object
▶ /wp/v2/posts/(?P<parent>[\d]+)/revisions:	Object
▶ /wp/v2/posts/(?P<parent>[\d]+)/revisions/(?P<id>[\d]+):	Object
▶ /wp/v2/pages:	Object
▶ /wp/v2/pages/(?P<id>[\d]+):	Object
▶ /wp/v2/pages/(?P<parent>[\d]+)/revisions:	Object
▶ /wp/v2/pages/(?P<parent>[\d]+)/revisions/(?P<id>[\d]+):	Object
▶ /wp/v2/media:	Object
▶ /wp/v2/media/(?P<id>[\d]+):	Object
▶ /wp/v2/feedback:	Object



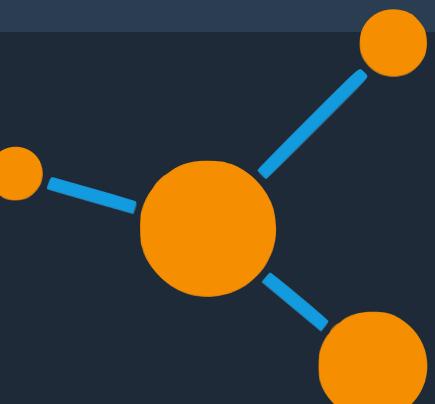
Methods

```
namespace: "wp/v2"
routes:
  ▶ /wp/v2: Object
    ▶ /wp/v2/posts:
      namespace: "wp/v2"
      methods:
        0: "GET"
        1: "POST"
      endpoints:
        ▶ 0: Object
        ▶ 1: Object
      _links:
        self:
          ▶ /wp/v2/posts/(?P<id>[\d]+): Object
          ▶ /wp/v2/posts/(?P<id>[\d]+)/comment-count: Object
```



Endpoints

Method	+	Route	Result
POST		wp/v2/posts/123	Create a post
GET		wp/v2/posts/123	Retrieve a post
PUT		wp/v2/posts/123	Update a post
DELETE		wp/v2/posts/123	Delete a post



Examples

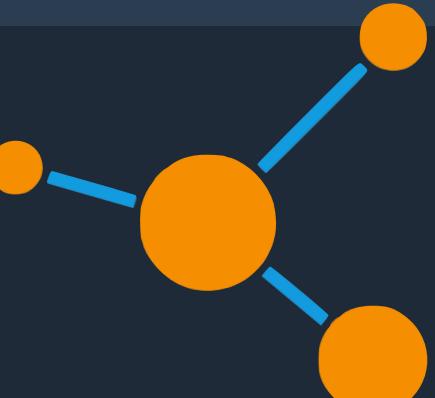
github.com/nagmay/WRK7

“WTF? Why did you choose JS for your examples”

- *someone in this workshop?*



The WordPress API



@nagmay

Developer Resources

Browse: [Home](#) / [REST API Handbook](#) / REST API Handbook



CHAPTERS

1 REST API Handbook

2 Reference

- Posts
- Post Revisions
- Categories
- Tags
- Pages
- Comments
- Taxonomies
- Media
- Users
- Post Types
- Post Statuses

REST API Handbook

The WordPress REST API provides API endpoints for WordPress data types that allow developers to interact with sites remotely by sending and receiving [JSON](#) (JavaScript Object Notation) objects. JSON is an open standard data format that is lightweight and human-readable, and looks like Objects do in JavaScript; hence the name. When you send content to or make a request to the API, the response will be returned in JSON. This enables developers to create, read and update WordPress content from client-side JavaScript or from external applications, even those written in languages beyond PHP.

TOPICS

Why use the WordPress REST API

Key Concepts

- [Routes & Endpoints](#)
- [Requests](#)
- [Responses](#)
- [Schema](#)
- [Controller Classes](#)



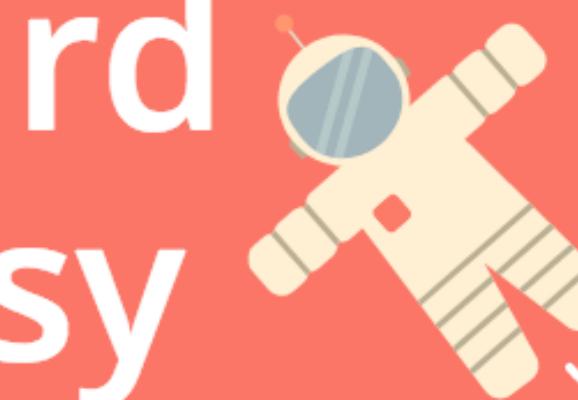
The WordPress API

@nagmay



POSTMAN

Developing APIs is hard
Postman makes it easy



The WordPress API

@nagmay

Pulling Data

POST .../posts



Create

GET .../posts



Retrieve

PUT .../posts

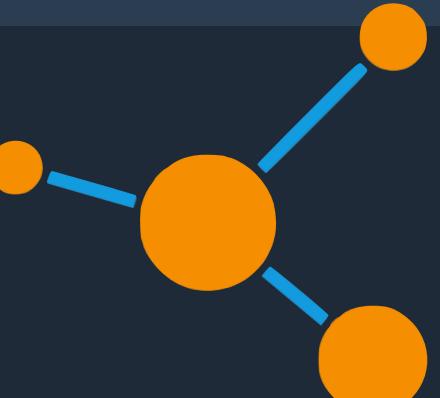


Update

DELETE .../posts



Delete



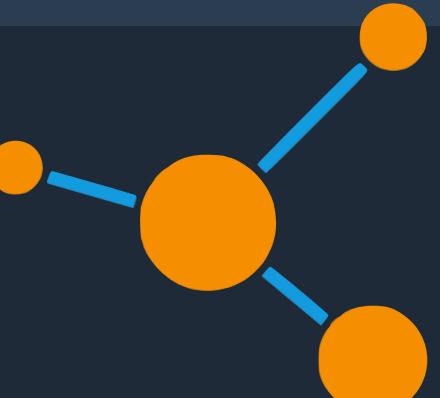
The WordPress API

@nagmay

01: GET JSON

```
$.ajax({  
    method: "GET",  
    url: url,  
    dataType: "json"  
}).done(function( data ) {  
    // do something  
});
```

```
$.getJSON(url, function (data) {  
    // do something  
});
```



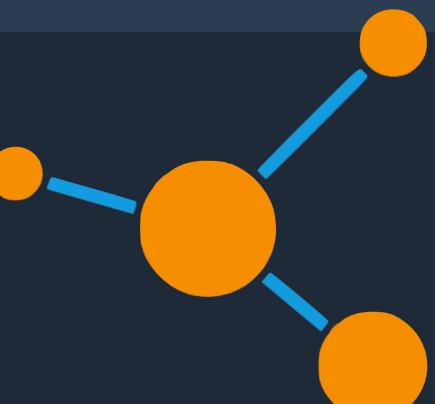
02: GET Posts

```
  ▼ title:  
    ▼ rendered: "Home to nursing and medical imaging, HT  
      comprehensive renovation"  
  
data[i].title.rendered
```

Response Headers

Access-Control-Allow-Headers Authorization, Content-Type
Access-Control-Expose-Headers X-WP-Total, X-WP-TotalPages

```
numPosts = request.getResponseHeader('x-wp-total');
```



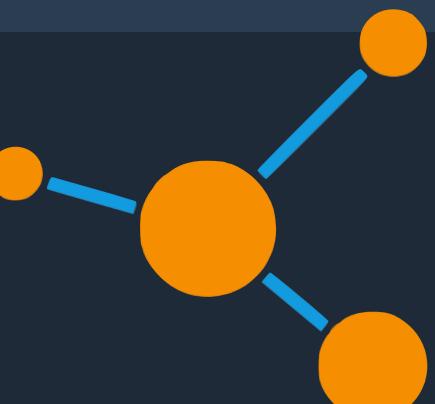
03: Posts Arguments

`http://news.pcc.edu/wp-json/wp/v2/posts/?page=1&search=biology`

http://news.pcc.edu GET some

Refine:

page	<input type="text" value="1"/>
per_page	<input type="text" value="#"/>
search	<input type="text" value="biology"/>
after	<input type="text" value="0000-00-00T00:00:00"/>
author *	<input type="text" value="user_id #"/>



04: Posts by year?

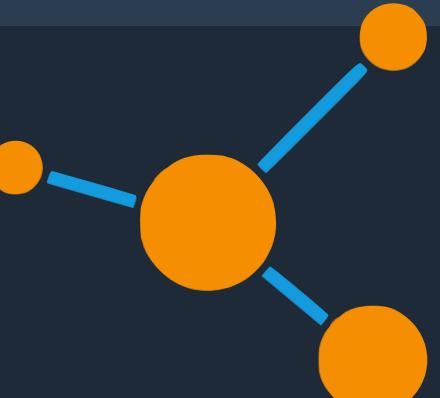
```
var url = $('#site').val()
    + '/wp-json/wp/v2/posts/?per_page=1&after='
    + year + '-01-01T00:00:00&before='
    + (year+1) + '-01-01T00:00:00';

// Ajax
$.getJSON(url, function (data, status, request) { // if success
    numPosts = request.getResponseHeader('x-wp-total');

    // Keep going?
    year -= 1;

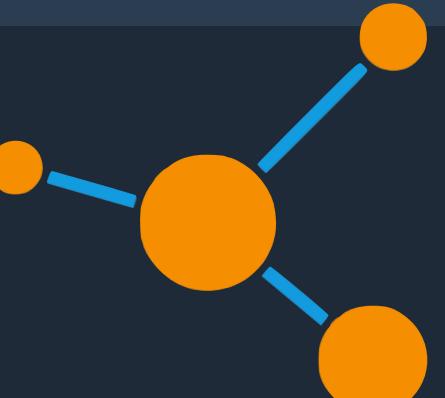
    ...
}

});
```



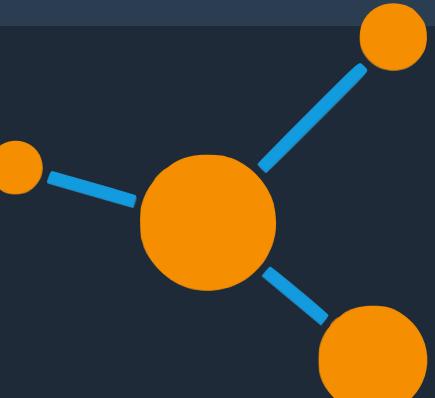
05: Posts by year? Smarter!

```
// Ajax 1: How old is this site?  
var url = $('#site').val()  
    + '/wp-json/wp/v2/posts?per_page=1&order=asc'; // Get  
$.getJSON(url, function (data, status, request) { // if success:  
  
    // Find the year of the first post  
    until = data[0].date;  
    until = until.substring(0,4);  
  
    ...  
});  
  
// Ajax 2: Get number of posts for each year  
...
```



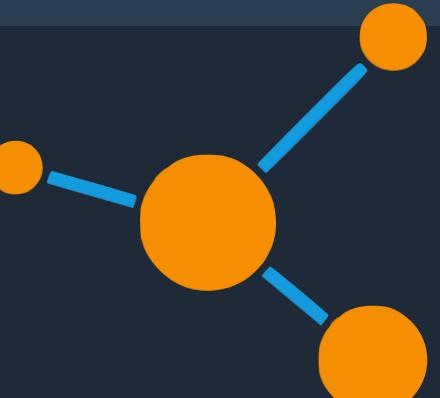
06: Comments by year

```
// Ajax 1: How old is this site?  
var url = $('#site').val()  
    + '/wp-json/wp/v2/comments?per_page=1&order=asc';  
$.getJSON(url, function (data, status, request) {  
  
    // Find the year of the first post  
    until = data[0].date;  
    until = until.substring(0,4);  
  
    ...  
});  
  
// Ajax 2: Get number of comments for each year  
...
```



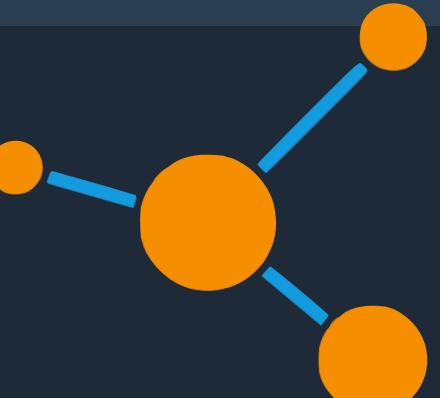
07: Media by year

```
// Ajax 1: How old is this site?  
var url = $('#site').val()  
    + '/wp-json/wp/v2/media?per_page=1&order=asc';  
$.getJSON(url, function (data, status, request) {  
  
    // Find the year of the first post  
    until = data[0].date;  
    until = until.substring(0,4);  
  
    ...  
});  
  
// Ajax 2: Get number of media uploads for each year  
...
```



08: Custom post types

```
/* === Custom Post Type: 'pcc_quotes' === */
add_action( 'init', 'pcc_register_custom' );
function pcc_register_custom() {
    // custom post types
    $post_args = array(
        'description' => '',
        'public' => true,
        'show_in_rest' => true,
        ...
    );
    register_post_type( "pcc_quotes", $post_args );
```

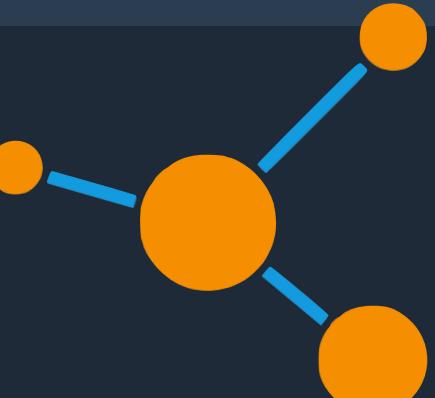


09: Users

```
GET http://news.pcc.edu/wp-json/wp/v2/users/?page=1
{"readyState":4,"responseText":"{\\"code\\":\\"rest_user_cannot_view\\",\\"message\\":\\"Sorry, you are not allowed to list users.\\"},\\"data\\":{\\"status\\":
```

Translation:

“Come back with a warrant”



Let's talk
about beer



The WordPress API

@nagmay



The WordPress API



@nagmay



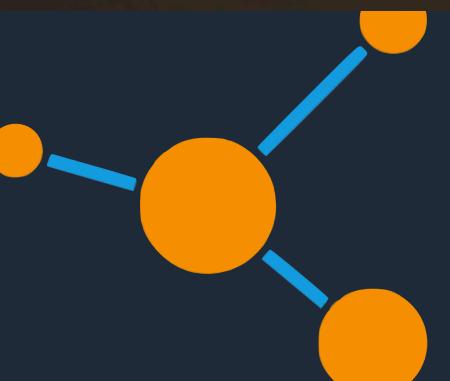
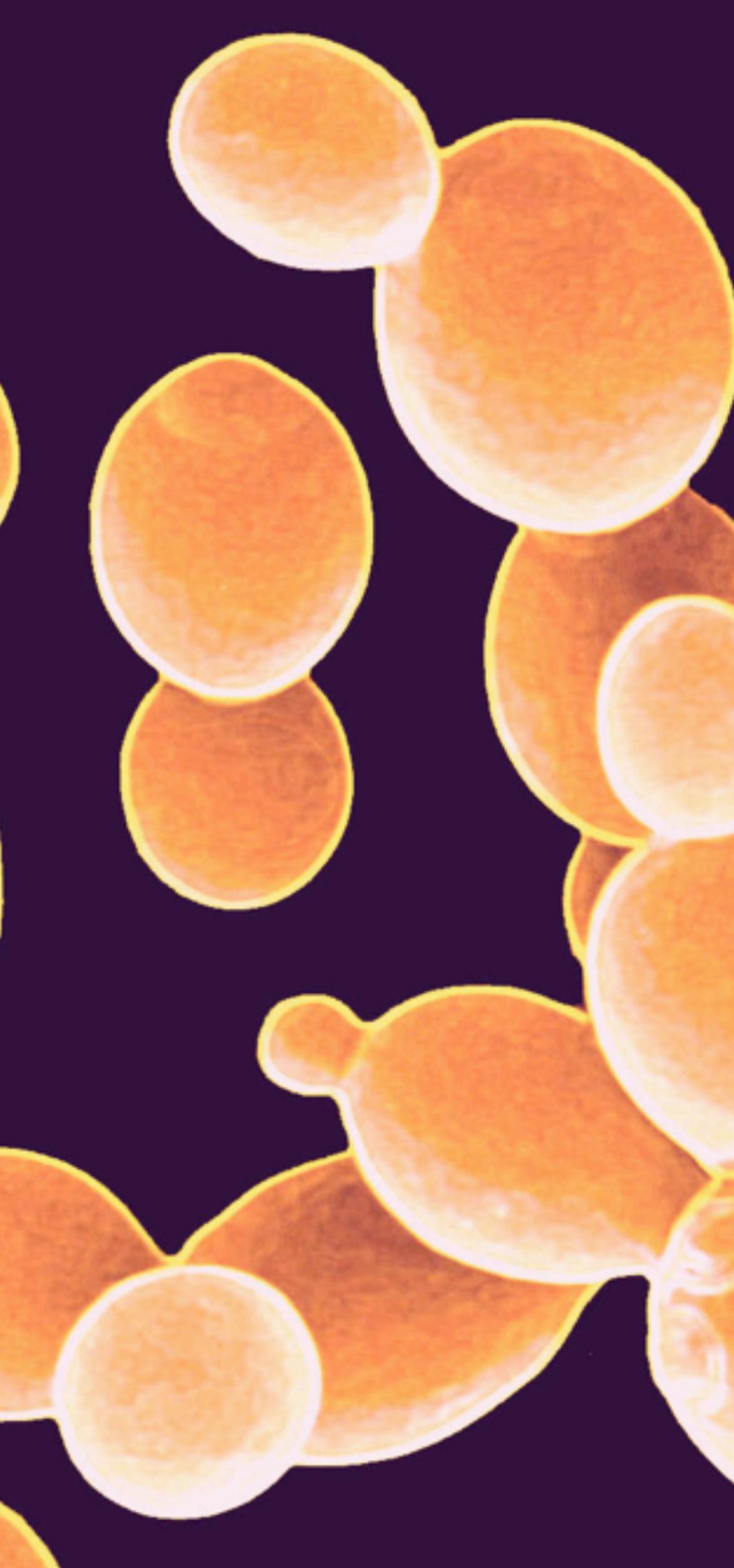
The WordPress API

@nagmay



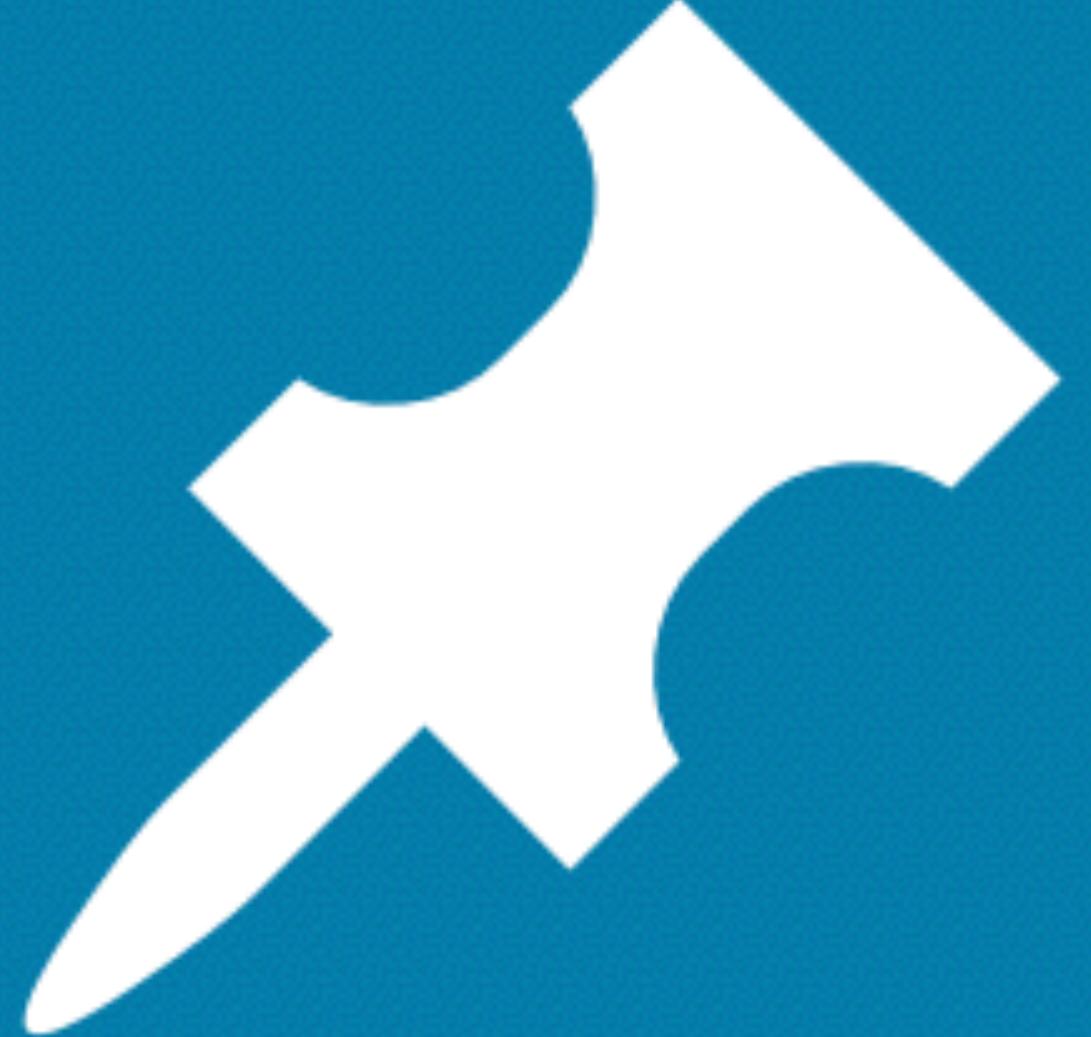
The WordPress API

@nagmay



The WordPress API

@nagmay



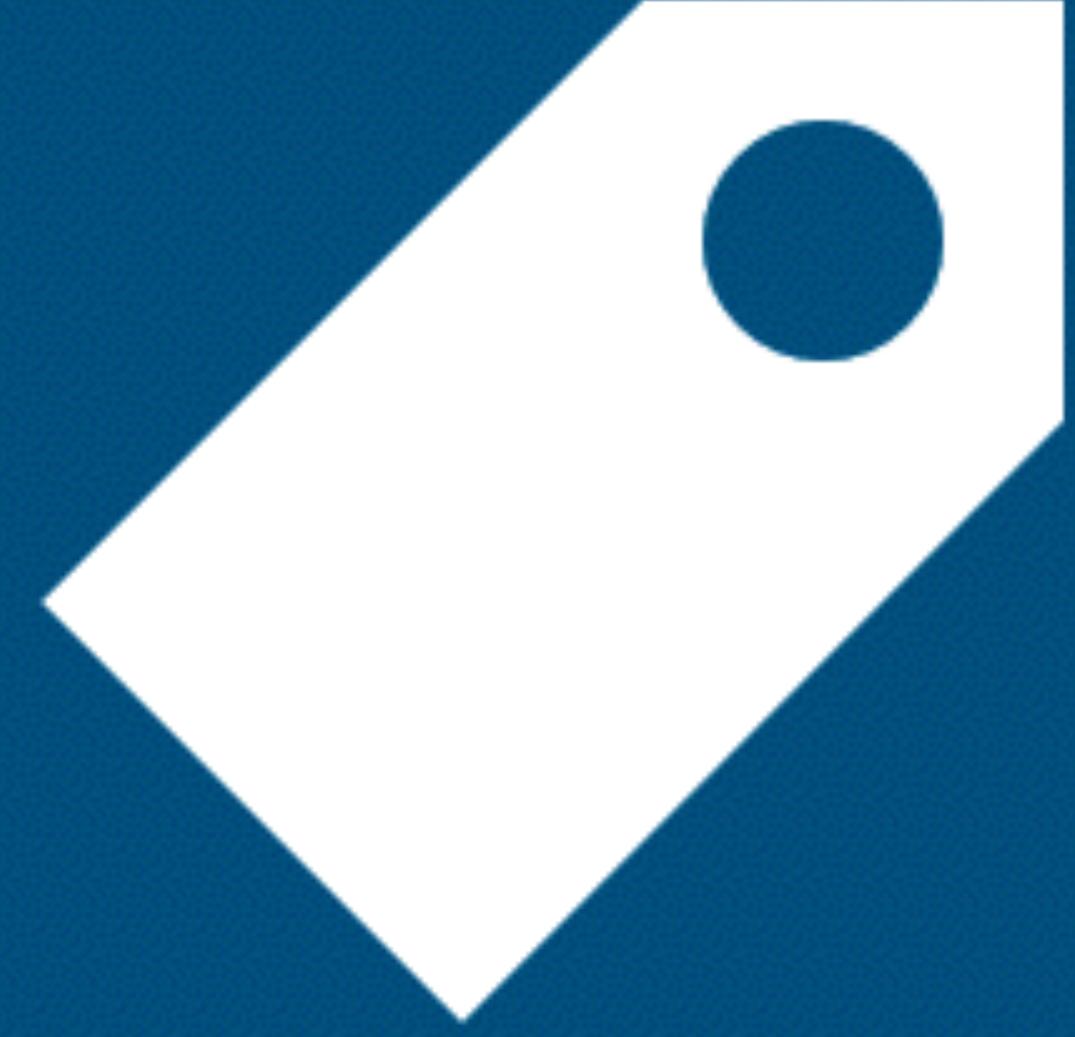
Posts



Pages



Media



Tags



The WordPress API

@nagmay

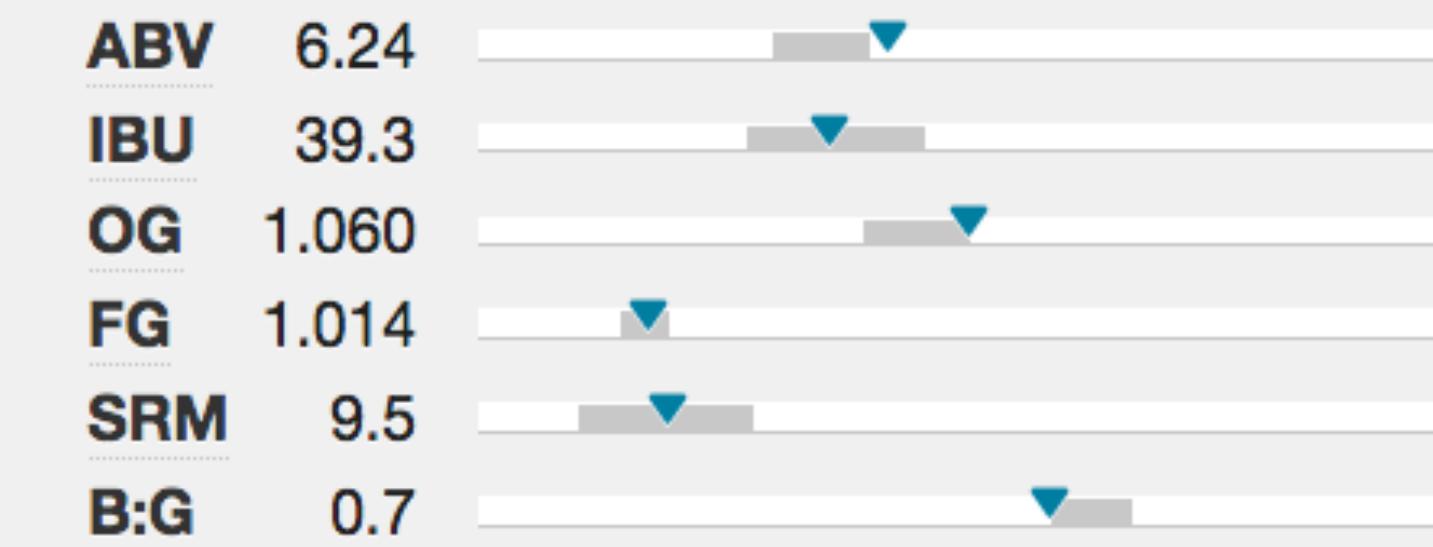
ALL GRAIN BEER



HTM-Ale

10A: American Pale Ale

5 gal 60 min 70 %



	Grain	Amount	Potential	SRM	SG +	SRM +	
Grain	Pale Malt (2 Row) US	10 lb	36	2	1.050	3.9	x
Hops	Caramel/Crystal Malt – 20L	1.5 lb	35	20	1.007	5.1	x
Yeast	Cara-Pils/Dextrine	0.5 lb	33	2	1.002	0.5	x
Water	+ Add some grain						

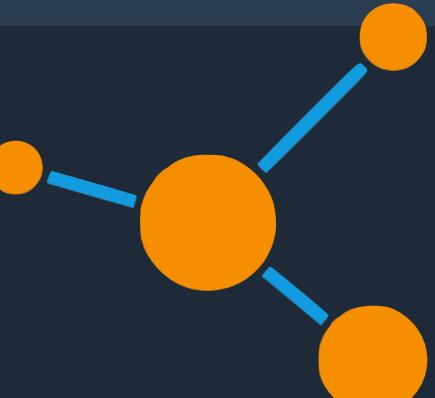
The WordPress API

@nagmay

10: Random Beer App

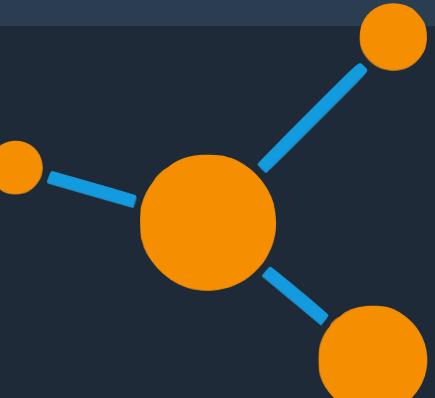
Can you build a simple app using only:

- Site address: <http://allgrain.beer>
- And what we know so far?



10: Meta

```
/* =====
 * REST: Custom Fields
 * =====
add_action('init', 'agb_register_meta');
function agb_register_meta(){
    $args = array(
        'type' => 'string',
        'show_in_rest' => true,
        'object_subtype' => 'agb_recipe'
    );
    register_meta( 'post', 'agb_recipe_abv', $args );
    register_meta( 'post', 'agb_recipe_boil', $args );
    register_meta( 'post', 'agb_recipe_efficiency', $args );
    ...
}
```



Pushing Data

POST .../posts



Create

GET .../posts



Retrieve

PUT .../posts

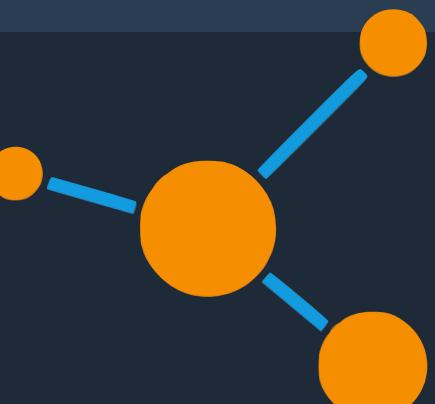


Update

DELETE .../posts



Delete



The WordPress API

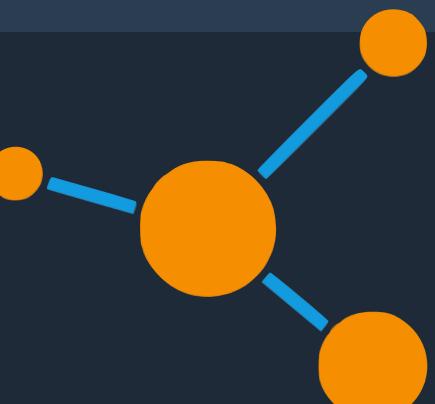
@nagmay

Authentication

Cookies &Nonce

OAuth plugin

Basic Auth plugin



The WordPress API

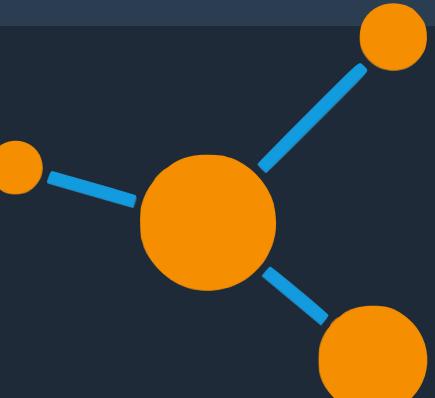
@nagmay

Cookies &Nonce

Super simple :)

Only available inside of WordPress :(

```
$.ajax( {  
    url: wpApiSettings.root + 'wp/v2/posts/1',  
    method: 'POST',  
    beforeSend: function ( xhr ) {  
        xhr.setRequestHeader( 'X-WP-Nonce', wpApiSettings.nonce );  
    },  
    data:{  
        'title' : 'Hello Moon'  
    } ).done( function ( response ) {  
        console.log( response );  
    } );
```



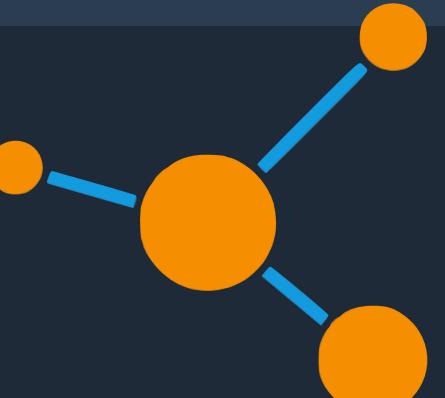
The WordPress API

@nagmay

OAuth: Server

WordPress REST API – OAuth 1.0a Server

```
name:          "All Grain Beer"  
description:   "build, create, and manage homebrew recipes"  
url:           "http://allgrain.beer"  
home:          "http://allgrain.beer"  
gmt_offset:    -7  
timezone_string: "America/Los_Angeles"  
▶ namespaces:  [...]  
▼ authentication:  
  ▶ oauth1:  
    request:      "http://allgrain.beer/oauth1/request"  
    authorize:    "http://allgrain.beer/oauth1/authorize"  
    access:       "http://allgrain.beer/oauth1/access"  
    version:      "0.1"  
  ▶ routes:      {...}
```



Basic Auth plugin

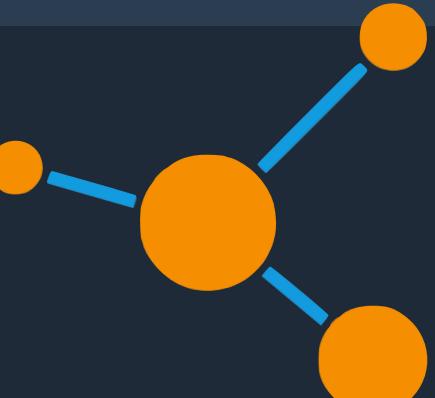
github.com/WP-API/Basic-Auth

WARNING: Requires sending your username and password with every request

```
# BEGIN Basic Auth

<IfModule mod_rewrite.c>
RewriteRule ^index\.php$ - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization},L]
</IfModule>

# END Basic Auth
```

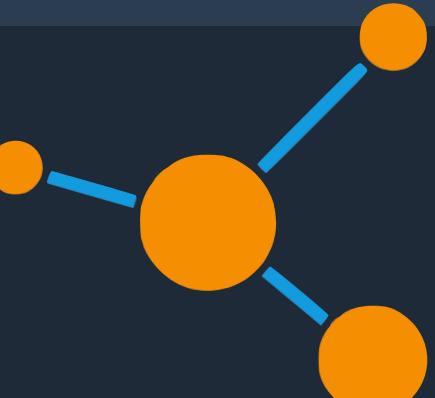


11: Get Posts (authorized!)

```
// The auth credentials
user = $('#user').val()
pass = $('#pass').val()

// encode 'username:password' in base 64
creds = btoa( user + ':' + pass );

// Ajax, well ... duh!
$.ajax({
    url: url,
    method: 'GET',
    crossDomain: true,
    beforeSend: function ( xhr ) {
        xhr.setRequestHeader( 'Authorization', 'Basic ' + creds );
    }
})
```

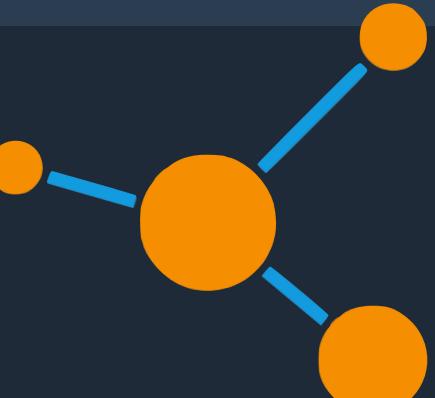


12: Create Posts

```
// The auth credentials
user = $('#user').val()
pass = $('#pass').val()

// encode 'username:password' in base 64
creds = btoa( user + ':' + pass );

// Ajax, well ... duh!
$.ajax({
    url: url,
    method: 'POST', // << hey, something new!
    crossDomain: true,
    beforeSend: function ( xhr ) {
        xhr.setRequestHeader( 'Authorization', 'Basic ' + creds );
    }
})
```

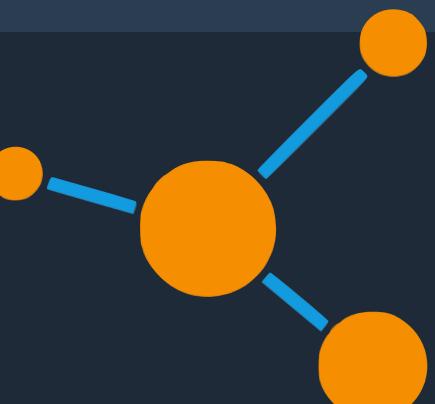


13: Update Posts

```
// The auth credentials
user = $('#user').val()
pass = $('#pass').val()

// encode 'username:password' in base 64
creds = btoa( user + ':' + pass );

// Ajax, well ... duh!
$.ajax({
    url: url,
    method: 'PUT', // Could also use POST
    crossDomain: true,
    beforeSend: function ( xhr ) {
        xhr.setRequestHeader( 'Authorization', 'Basic ' + creds );
    }
})
```

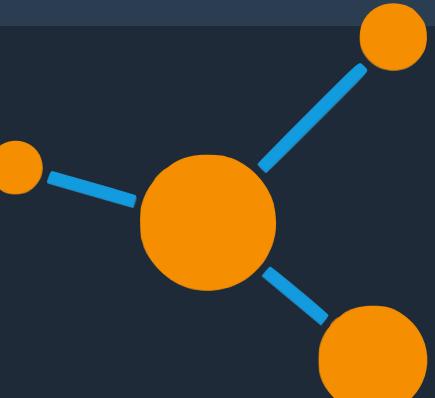


14: Delete Posts

```
// The auth credentials
user = $('#user').val()
pass = $('#pass').val()

// encode 'username:password' in base 64
creds = btoa( user + ':' + pass );

// Ajax, well ... duh!
$.ajax({
    url: url,
    method: 'DELETE', // Goodbye!
    crossDomain: true,
    beforeSend: function ( xhr ) {
        xhr.setRequestHeader( 'Authorization', 'Basic ' + creds );
    }
})
```

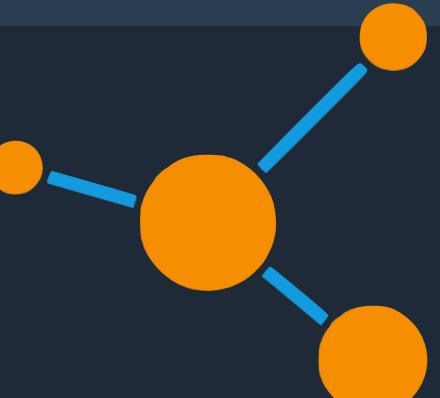


15: Users

```
// The auth credentials
user = $('#user').val()
pass = $('#pass').val()

// encode 'username:password' in base 64
creds = btoa( user + ':' + pass );

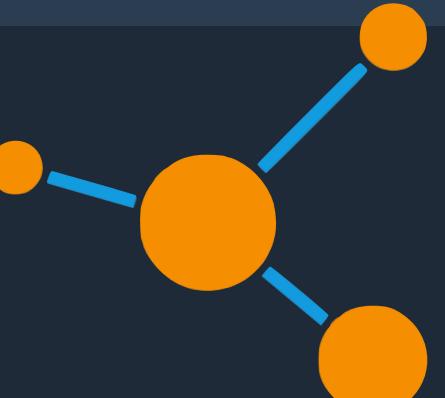
// Ajax, well ... duh!
$.ajax({
    url: url,
    method: 'GET',
    crossDomain: true,
    beforeSend: function ( xhr ) {
        xhr.setRequestHeader( 'Authorization', 'Basic ' + creds );
    }
})
```



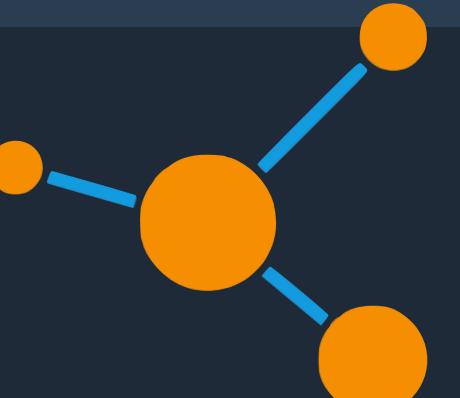
16: Media

```
var formData = new FormData();
formData.append( 'file', file );
formData.append( 'title', title );
formData.append( 'alt_text', alt_text );

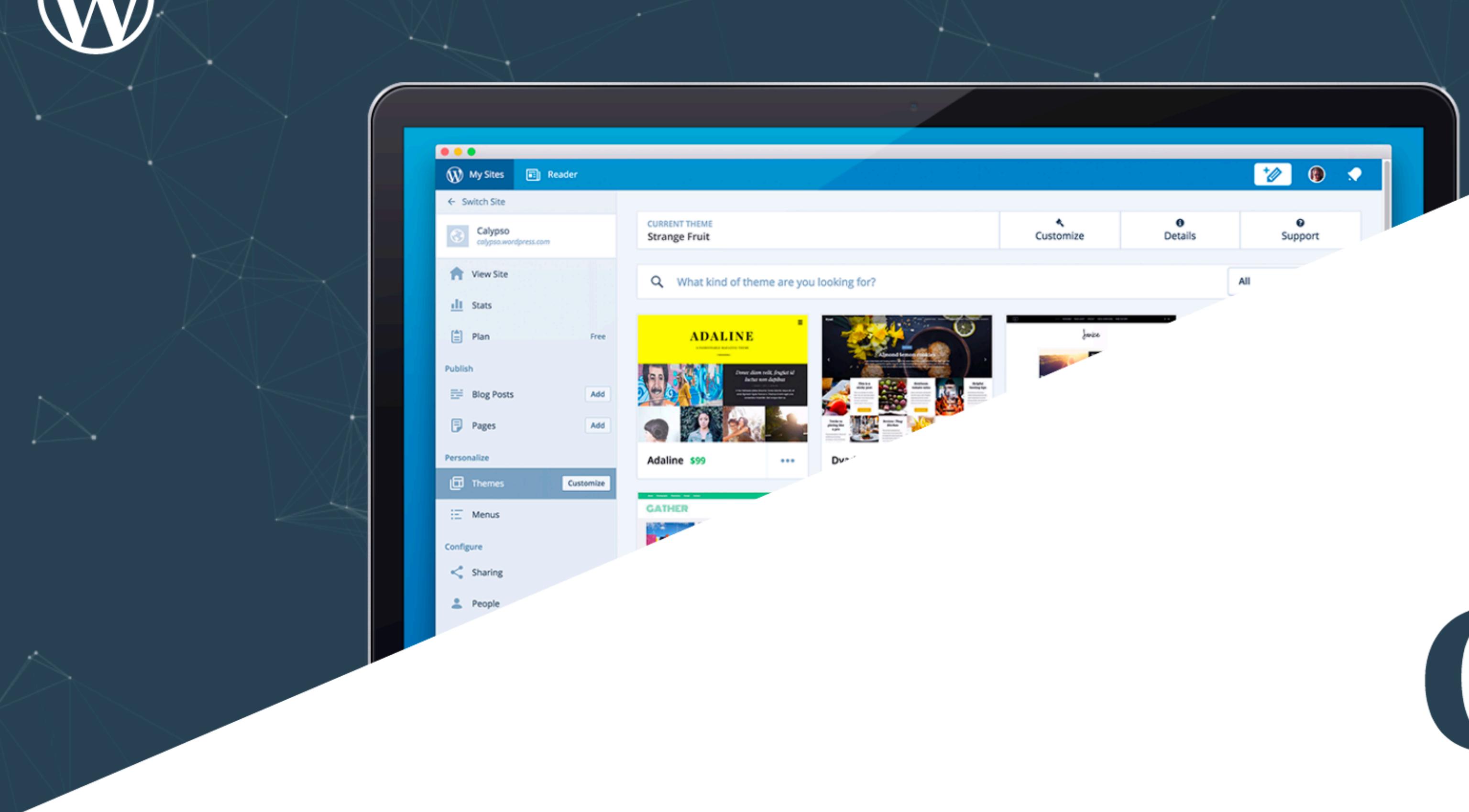
// Ajax, well ... duh!
$.ajax({
    url: url,
    method: 'POST',
    crossDomain: true,
    processData: false, // don't process into urlencoded content type
    contentType: false, // don't send the default content type header
    data: formData, // << here's all your form data!
    beforeSend: function ( xhr ) {
        xhr.setRequestHeader( 'Authorization', 'Basic ' + creds );
    }
})
```



Real world examples?



@nagmay



Introducing

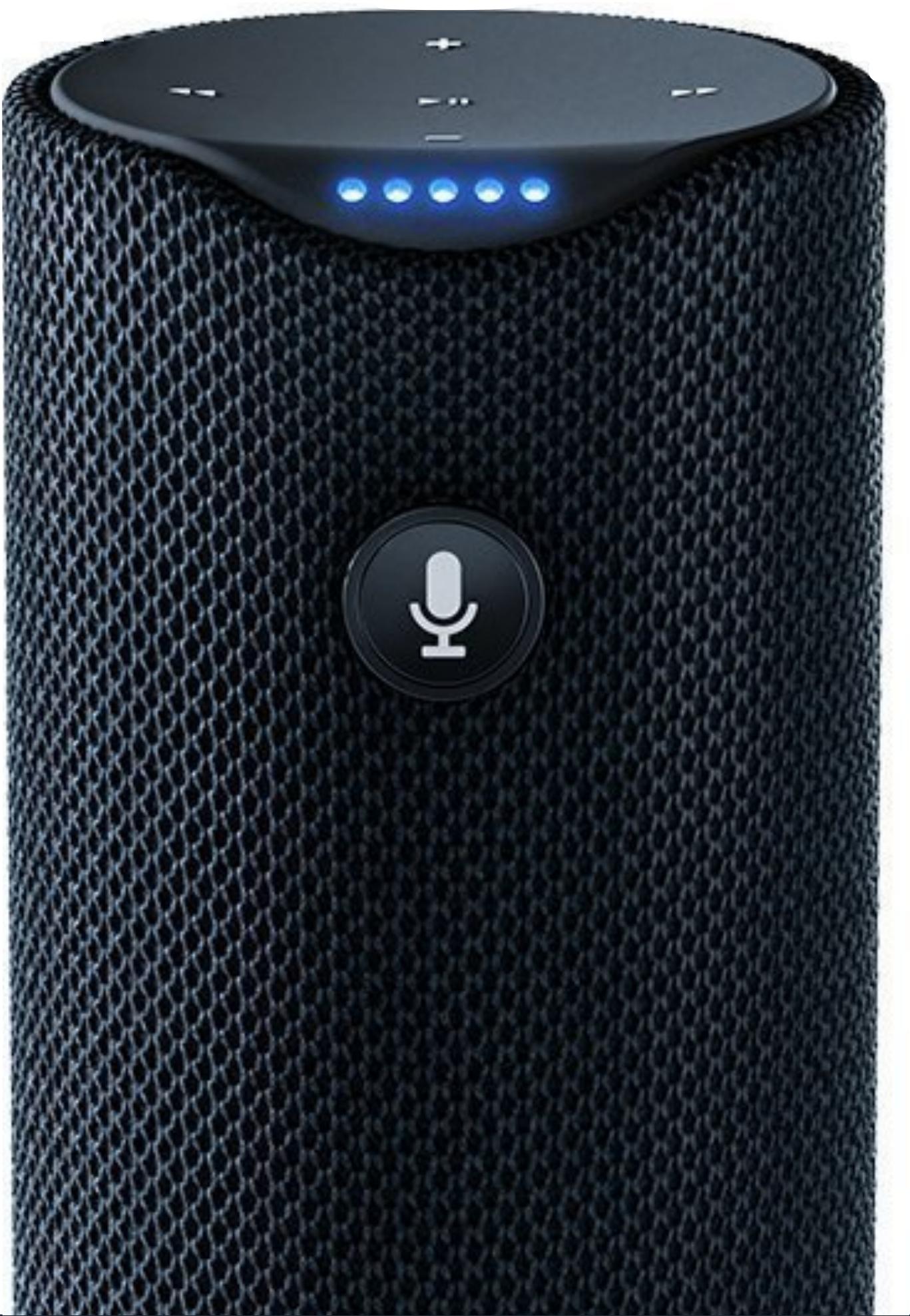
Calypso

A single interface to manage all your WordPress.com or Jetpack-enabled sites, built with the latest web technologies and used by millions of people — and now it's open source.

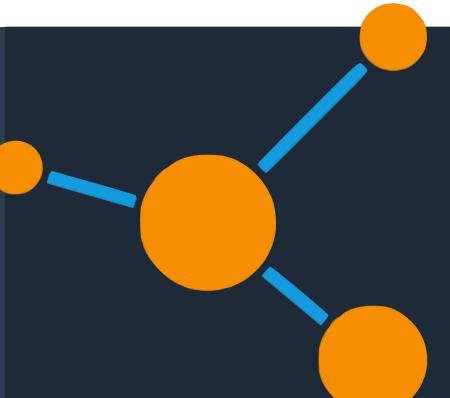
For OSX 10.8+, Windows 7+, and Linux.

The WordPress API

@nagmay



The WordPress API



@nagmay

The diagram illustrates the evolution of a website, likely using the WordPress API, through three distinct stages:

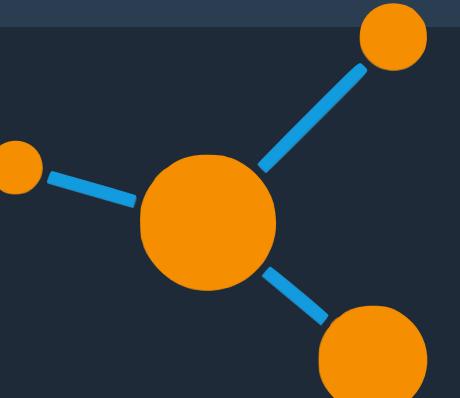
- Stage 1 (Left):** A news archive page. The top navigation bar includes links for search, resources, contacts, calendars, give, and mypcc. Below the header, a "News" section lists posts from 2017. A sidebar on the right provides options to follow the site via RSS or email, and a "What's Hot?" section lists recent articles. A "Archives" section shows monthly post counts. A search bar at the bottom allows users to search PCC News.
- Stage 2 (Middle):** A single news article page titled "Caring for HT". The top navigation bar remains consistent. The main content features a large image of a healthcare professional, a brief summary of the story, and a "Read the story >" button. A sidebar on the right contains links for "Earn Your Degree or Certificate", "Get Job Ready", and "Enrich Your Life".
- Stage 3 (Right):** A modern college homepage. The top navigation bar includes links for search, resources, contacts, calendars, give, and mypcc. The main content area features a large image of a healthcare professional, a title "Caring for HT", a brief summary, and a "Read the story >" button. Below this, there are sections for "College news and events" (with links to news items like "Caring for HT" and "College Open Panther Pantries") and "Spotlights" (featuring "OCTOBER 2017 CYBER SECURITY Awareness Month"). The footer is dark blue and contains logos for Portland Community College, the Portland Business Journal (Top 20 Best Value), Community College Week (Top 20 Community Colleges), and the U.S. Green Building Council (LEED Platinum). It also includes links for Student essentials, On campus, Connect, and Get help, along with social media icons.

The WordPress API

@nagmay

Thanks Everyone

Questions?



@nagmay

BOSTON UNIV. CENTRAL
OUTBOUND TO BOSTON COLLEGE

WPCAMPUS

WHERE WORDPRESS MEETS HIGHER EDUCATION

WPCampus Online 2018 (*virtual conference*)

Tuesday, January 30, 2018

Our call for speakers is open until Tuesday, November 7, 2017.

Want to experience WordPress sessions, expertise, and networking with users across the world without the expense and hassle of travel? This one-day, session-filled event is for you!

Visit online.wpcampus.org for more information.

wpcampus.org

@wpcampusorg

#WPCampus

WPCampus 2018 (*in-person conference*)

Summer 2018

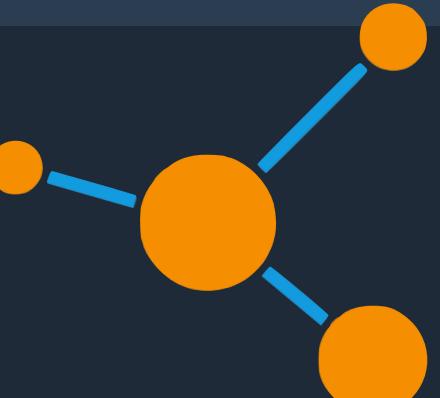
The hosting application will close Tuesday, October 31, 2017.

The main event for WPCampus is our annual in-person conference. We hold this event on college campuses and we'd love to bring our community to you.

Visit wpcampus.org to learn more about the event and how to host.

Custom Endpoints

```
add_action( 'rest_api_init', function () {
    register_rest_route(
        'myplugin/v1',           // namespace
        '/stuff/(?P<id>\d+)',   // route
        array(                   // args (allowed methods, callback)
            'methods' => 'GET',
            'callback' => 'my_awesome_func',
        )
    );
});
```



Custom Endpoints

```
function my_awesome_func( $data ) {
    // bad call
    if ( empty( $posts ) ) {
        return new WP_Error( 'no_stuff',
            'Stuff was not found',
            array( 'status' => 404 ) );
    }

    // get the data to return
    $posts = get_posts( array(
        'stff' => $data['id'],
    ) );
    return $posts[0]->post_title;
}
```

