BACKBONE & THE WP REST API

Backbone + WP REST API = HAPPY

Why Backbone?

Models • Collections • Views
Routes & Events

Philosophically, Backbone is an attempt to discover the minimal set of data-structuring (models and collections) and user interface (views and URLs) primitives that are generally useful when building web applications with JavaScript.

Models

Collections

Views

```
<script id="tmpl-header" type="text/html">
    <div class="header-title">
        <h1>{{ data.title }}</h1>
    </div>
</script>
```

ALSO: Routes, Events

Related Links

Add New

Title:

Url:

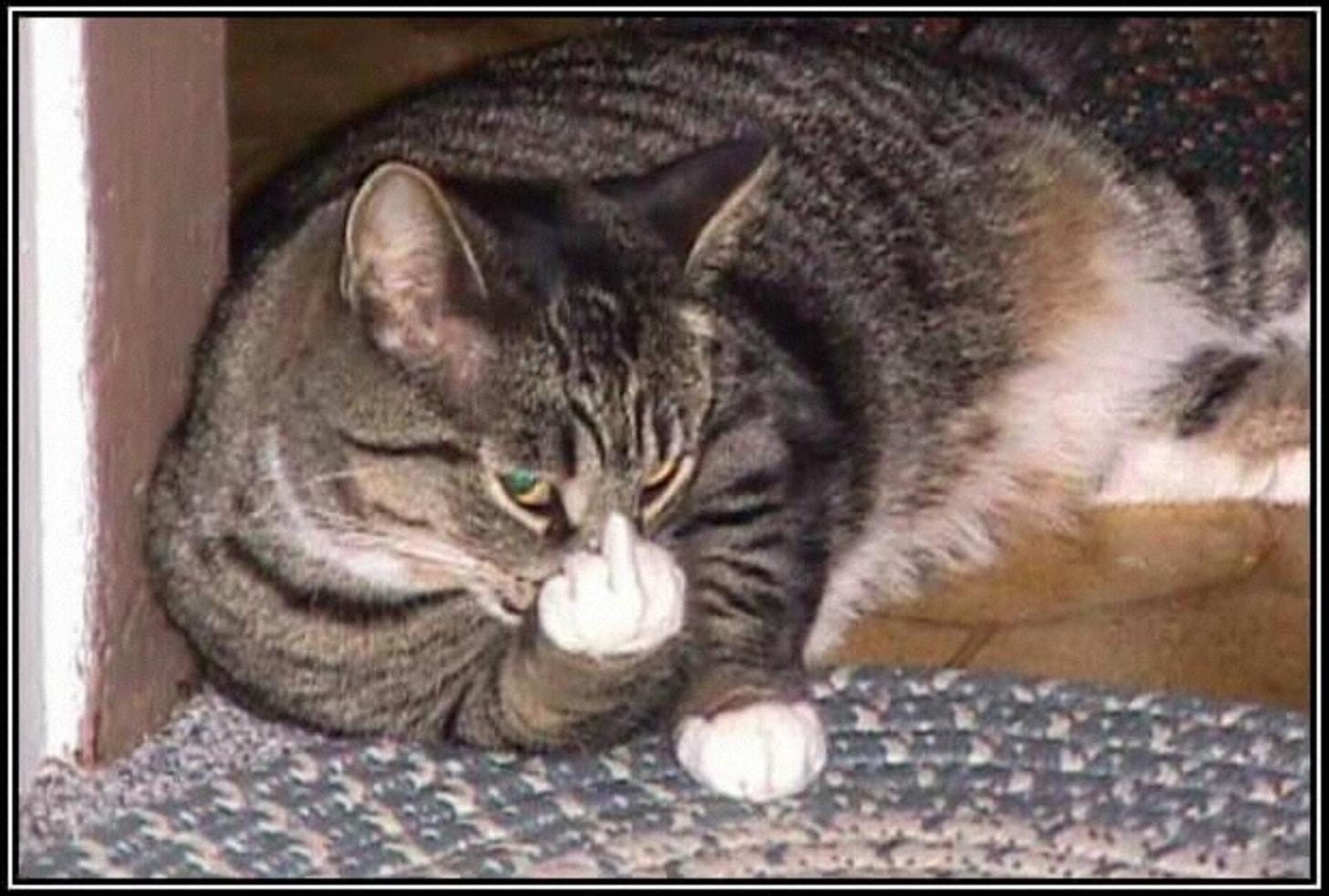
Entering a # here will display the item as plain text.

Open in new window

Add

Test

Another item



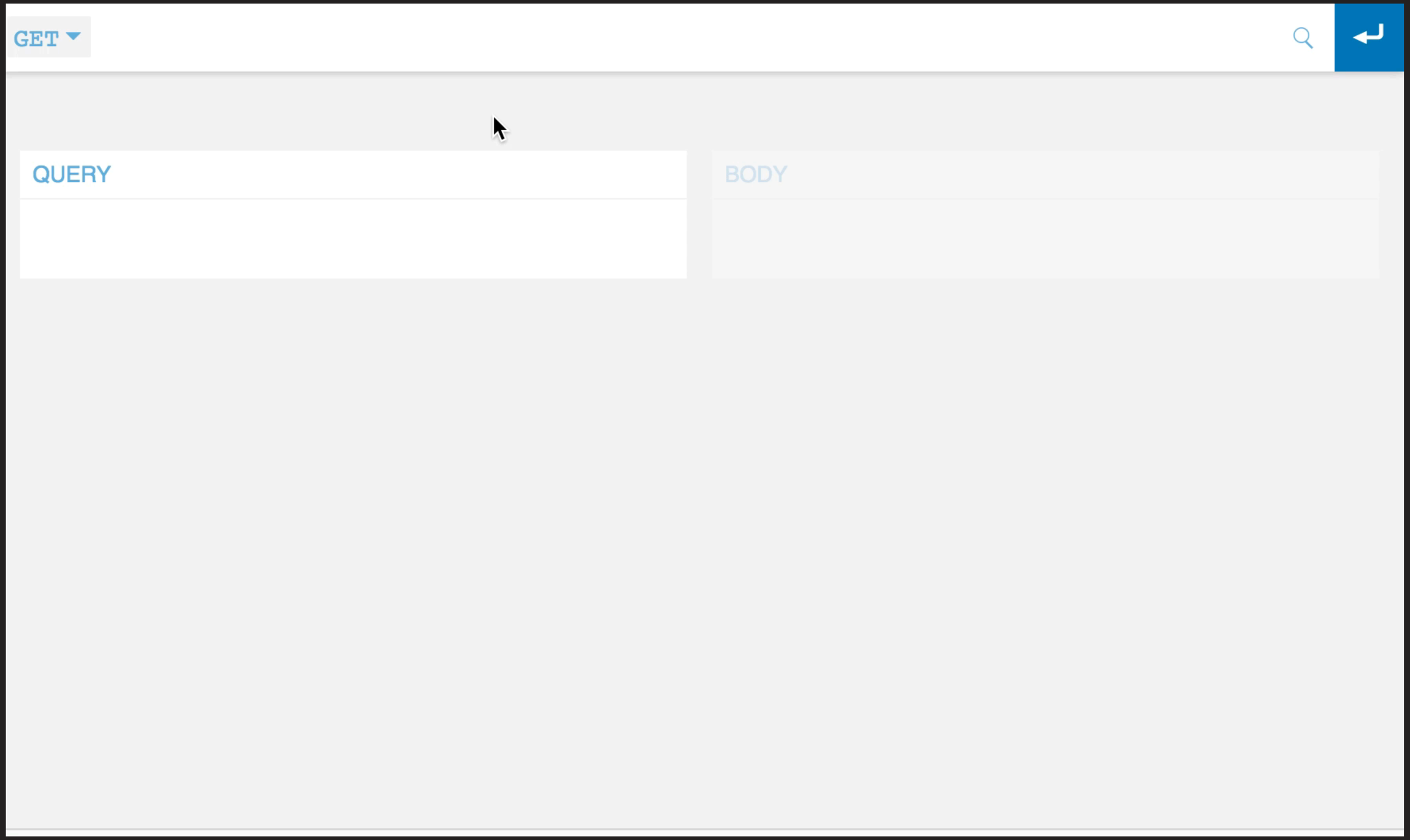
400

Bad Request

http.cat

What is the WP REST API?

demo.wp-api.org


GET
QUERY
BODY

WordPress

REST

API

wpdev.localhost/wp-json


```
  name: "WordPress Dev",
  description: "Just another WordPress site",
  url: "http://wpdev.localhost",
  home: "http://wpdev.localhost",
+ namespaces: [ … ],
  authentication: [ ],
+ routes: { … },
+ _links: { … }
```

Backbone + WP REST API = HAPPY

Backbone + WP REST API + JS CLIENT = LOVE

The (Backbone) JavaScript Client

How to use

```
wp_enqueue_script( 'wp-api' );
```

```
wp_enqueue_script( 'my_script', 'path/to/my/script', array( 'wp-api' ) );
```

```
wp.api.loadPromise.done( function() {
    //... use the client here
} )
```

What it does

Some examples...

```
// Load an existing post.
var post = new wp.api.models.Post( { id: 1 } );
post.fetch();
```

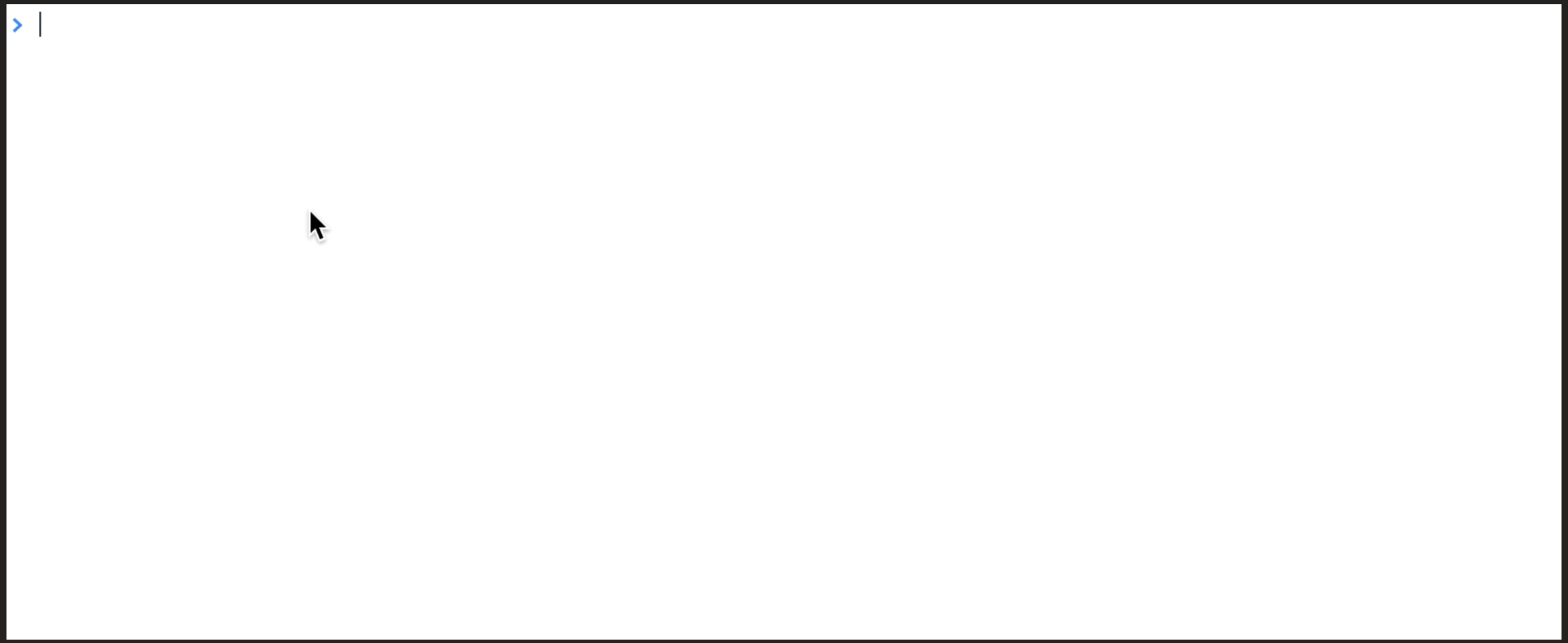
```
// Create a new post.
var post = new wp.api.models.Post( { title: 'This is a test post' } );
post.save();
```

```
// Post helpers.
post.getAuthorUser().done( function( user ){ ... }
post.getFeaturedImage().done( function( image ){ ... }

post.setCategories( [ 'apples', 'oranges' ] );
```

```
> var post = new wp.api.models.Post( { id:1 } );
```

```
// Collections.
var postsCollection = new wp.api.collections.Posts();
postsCollection.fetch();
```



```
// More query options.
postsCollection.fetch( {
    data: { 'filter':
        {
            'orderby': 'title',
            'order': 'ASC'
        }
    }
} );

// Pagination.
postsCollection.hasMore();
postsCollection.more();
```

```
// Listen for changes on the model and sync them.
initialize: function() {
    this.model.on( 'change', this.model.save(), this );
}
```

It’s that easy

Challenges

Adding custom data

```php
/**
 * Filter the post data for a response.
 *
 * The dynamic portion of the hook name, $this->post_type, refers to post_type of the post being
 * prepared for the response.
 *
 * @param WP_REST_Response   $response   The response object.
 * @param WP_Post            $post       Post object.
 * @param WP_REST_Request    $request    Request object.
 */
return apply_filters( "rest_prepare_{$this->post_type}", $response, $post, $request );
```

```php
/**
 * Filter a post before it is inserted via the REST API.
 *
 * The dynamic portion of the hook name, $this->post_type, refers to post_type of the post being
 * prepared for insertion.
 *
 * @param stdClass        $prepared_post An object representing a single post prepared
 *                                       for inserting or updating the database.
 * @param WP_REST_Request $request       Request object.
 */
return apply_filters( "rest_pre_insert_{$this->post_type}", $prepared_post, $request );
```

```
	/**
	 * Filter the allowed 'private' query vars for authorized users.
	 *
	 * If the user has the `edit_posts` capability, we also allow use of
	 * private query parameters, which are only undesirable on the
	 * frontend, but are safe for use in query strings.
	 *
	 * To disable anyway, use
	 * `add_filter( 'rest_private_query_vars', '__return_empty_array' );`
	 *
	 * @param array $private_query_vars Array of allowed query vars for authorized users.
	 * }
	 */
	$private = apply_filters( 'rest_private_query_vars', $wp->private_query_vars );
```

```php
/**
 * Filter the publicly allowed query vars.
 *
 * Allows adjusting of the default query vars that are made public.
 *
 * @param array  Array of allowed WP_Query query vars.
 */
$valid_vars = apply_filters( 'query_vars', $wp->public_query_vars );
```

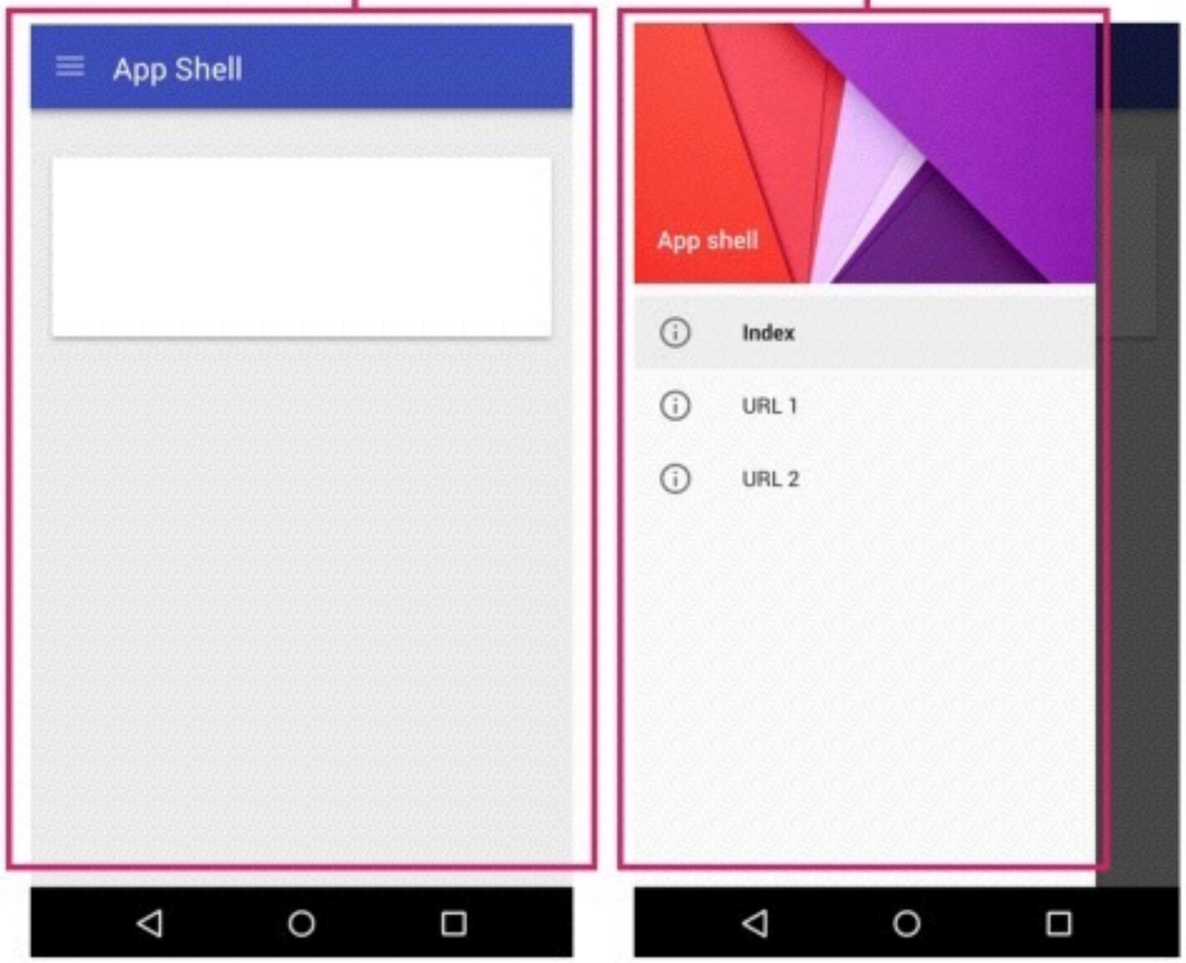
```php
/**
 * Filter the query arguments for a request.
 *
 * Enables adding extra arguments or setting defaults for a post
 * collection request.
 *
 * @see https://developer.wordpress.org/reference/classes/wp_user_query/
 *
 * @param array           $args    Key value array of query var to query value.
 * @param WP_REST_Request $request The request used.
 */
$args = apply_filters( "rest_{$this->post_type}_query", $args, $request );
```

React? Redux?

Progressive Rendering
or Application Shell Architecture

application shell

content



Cached shell loads **instantly** on repeat visits.

Dynamic content then populates the view

LINKS

http://backbonejs.org
https://github.com/WP-API/WP-API
https://github.com/WP-API/client-js
http://v2.wp-api.org/extending/javascript-client
https://github.com/redbooth/backbone-redux

ADAM SILVERSTEIN

Adam Silverstein

- Lead Web Engineer @ 10up
- Serial Core Contributor & release co-lead
- Lead developer of the Rest API Backbone JavaScript Client

@roundearth • tunedin.net • adam@10up.com