MORE SECURE API REQUESTS WITH…

WORDPRESS HELPER FUNCTIONS

@N8FINCH

NATE FINCH… WHO??

- WP for ~6 years, ~5 years full-time
- Previously: toptal, Codeable, 10up
- VP of Development at The Digital Ring
- Texas >> Kansas >> Spain (a little time in Morocco) >> France >> Illinois >> S. Korea >> and now Madison
- *“I’d be a professional student if I could…”*
- Married w/ 1.5 yo daughter

WHAT IS AN API?

A SET OF SUBROUTINE DEFINITIONS, COMMUNICATION PROTOCOLS, AND TOOLS FOR BUILDING SOFTWARE. . . IT IS A SET OF CLEARLY DEFINED METHODS OF COMMUNICATION AMONG VARIOUS COMPONENTS.

wikipedia.org (of course 🤓)

OK, SO WHAT DOES THAT MEAN?

- **Request/response cycle** (👏 👏 👏 for the WWW !!!)
- Ask for something, receive something back
- Abiding by a certain set of rules
- Take advantage of information/data we don't control
- Interacting with software we probably didn't write (unknowns)
- Some info we need, some we don't (! GraphQL)

WHAT IS "MORE" SECURE?

WHAT ARE WE SECURING?

- HTTPS: pretty standard now
- Hiding certain information (🔑 , PII, errors…)
  - Processing on the server (sometimes required by API)
  - Only returning what the client (browser) needs
- Sanitizing/escape data
- Don’t write from scratch

WHAT ARE WE AVOIDING?

- The browser is (almost) an open book
- Exposed API keys
  - some exceptions like Google Maps… but still… 😬
- All information gets returned
- Some info you do not want or need returned

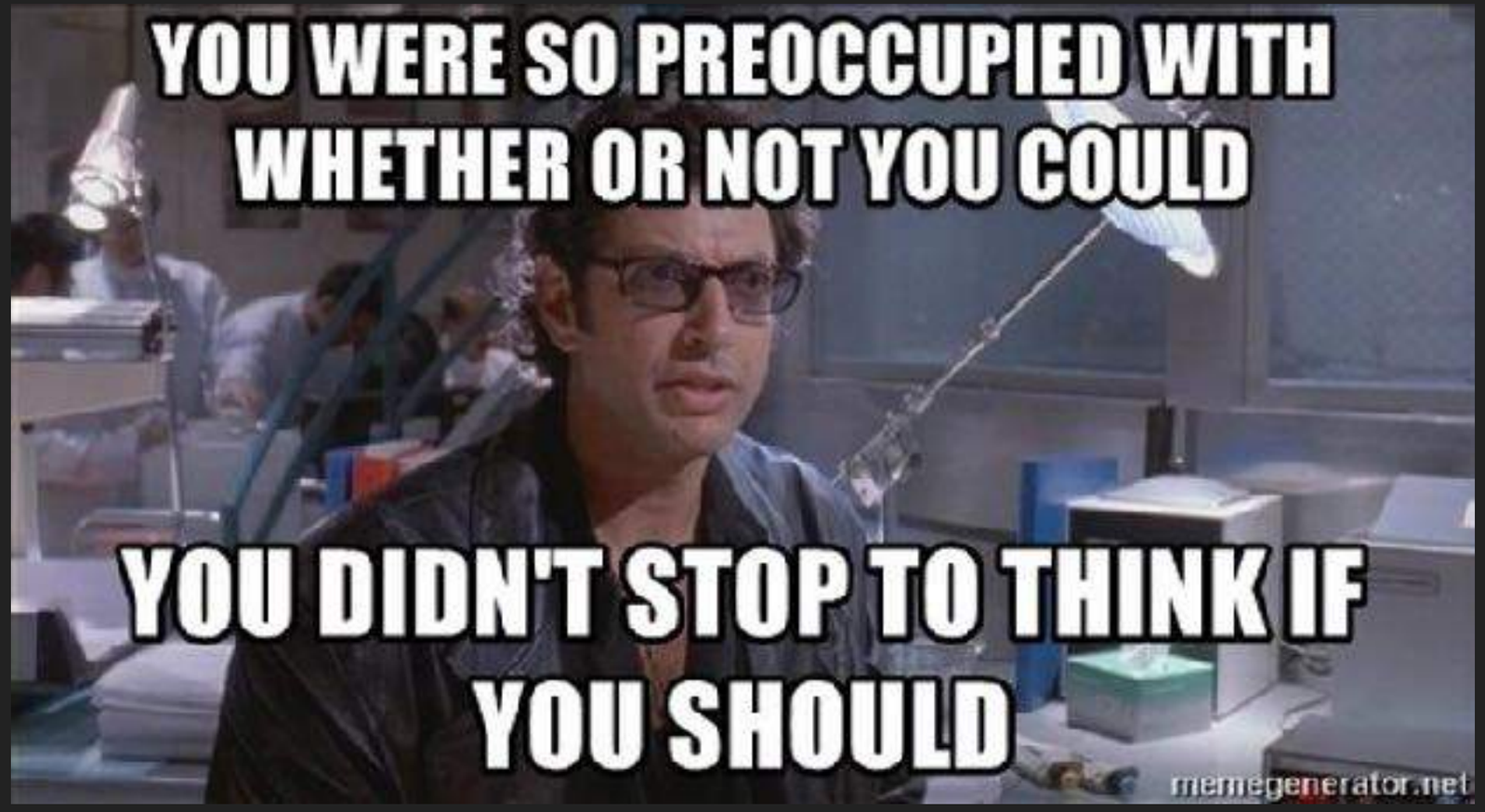
THE HELPER FUNCTIONS

WHY USE THESE HELPER FUNCTIONS

- Bundled with WordPress already (jQuery too), not adding dependencies
- Moves processing to the server vs. browser
- Sanitization and escaping built-in/available
- More succinct, manageable code, some defaults

"BUT I WANT TO DO
THINGS THE WAY I WANT
TO DO THINGS..."

COOL, GO FOR IT!



SOME OF THE AVAILABLE FUNCTIONS

- `wp_ajax_ and wp_ajax_nopriv` - allows you to handle your custom AJAX endpoints for authenticated and unauthenticated users
- `wp_remote_get| _post` - Retrieve the raw response from the HTTP request using the GET | POST method . Add `_safe_` for arbitrary URLs: validated to avoid redirection and request forgery attacks
- `esc_raw_url` - returns a cleaned url, does not replace entities (&, ‘) for display, safe to use in database queries, redirects, and HTTP requests.
- `wp_remote_retrive_header | _body` - returns the request body as a string, or a blank string on error
- `register_rest_route` - take advantage of the built in WP REST API and caching (possibly substitute for `wp_ajax_`)
- An entire HTTP class is available to explore in the **WordPress Code Reference**

THE BASIC "MORE" SECURE CYCLE

Initiation

Request + 🔑

Endpoint

Response

Filter

Data

A LITTLE SOMETHING EXTRA

- Save/cache the response with the Options API or Transients API (options to autoload)

THE CODE

WHAT DOES THIS LOOK LIKE IN PRACTICE?

```
<script type="text/javascript">
    jQuery('some-target').on('click', function( e ) {
        //prevent page reload and set up the vars
        e.preventDefault();
        var ajaxurl = ajax_object.ajax_url;
        var somethingFromTarget = e.target.dataset.thatSomething;
    var data = {
        'action': 'this_ajax_hook',
        'data_thing': somethingFromTarget,
    };

    console.log('Trying to do ajax request...');
```

```
    console.log('Trying to do ajax request...');
    // run the ajax request
    jQuery.post(ajaxurl, data, function(response) {
        if( response ) {
            console.log('The request was a massive success!!!');
            renderAPIResults( response );
        } else {
            console.log('Something went terribly wrong...');
        }
    });
});
</script>
```

```php
//3.Add in the handler
add_action('wp_ajax_this_ajax_hook', 'do_some_ajax_action');
add_action('wp_ajax_nopriv_this_ajax_hook', 'do_some_ajax_action');

function do_some_ajax_action() {
    // Handle request then generate response

    /**
     * CODE GOES HERE
     */

    // Don't forget to stop execution afterward.
    wp_die();
}
```

```php
<?php

$post_url = 'https://api.awesome.com/v2/widget?auth_token=XXXXXXXX';

// Validate and Sanitize input, then json_encode
$email = ( is_email( $email ) ) ? sanitize_email( $email ) : '';
$arg_data = array( 'email' => $email );
$data = json_encode( $arg_data );

$args = array(
    'headers' => array(
        'Content-Type' => 'application/json',
    ),
    'body' => $data,
);

// Post and get the response
$response = wp_remote_post( esc_url_raw( $post_url ), $args );

// Can set variables based on response
$response_code = wp_remote_retrieve_response_code( $response );
```

```php
// Post and get the response
$response = wp_remote_post( esc_url_raw( $post_url ), $args );

// Can set variables based on response
$response_code = wp_remote_retrieve_response_code( $response );

if ( is_wp_error( $response ) ) {
    $error_message = $response->get_error_message();
    echo "Something went wrong: $error_message";
} elseif (200 === wp_remote_retrieve_response_code( $response ) ) {
    // Do stuff with the response body
    $response_body = wp_remote_retrieve_body( $response );
    // Return the string to the browser as is:
    echo ( $response_body );
    // OR do some processeing, turn response into an object
    $res = json_decode( $response_body );
    // THEN return a string to the browser
    $res = wp_json_encode( $res );
    echo $res;
}
```

```
<script>

var renderAPIResults = function( response ) {

    //Remove the spinner
    jQuery('.spinner-gif').remove();
    //Remove the message if there is one before the response is posted
    jQuery('.flash-message').remove();
    //Insert the response message
    jQuery('#cmnow-get-code-validator-form').append( '<div
    class="flash-message"><p>' + response + '</p></div>' );

}
```

QUESTIONS??

NO?? OK, COOL, THANKS!

IF YOU THOUGHT THIS WAS FUN. . .

WE'RE ALWAYS LOOKING FOR FUN FOLKS!