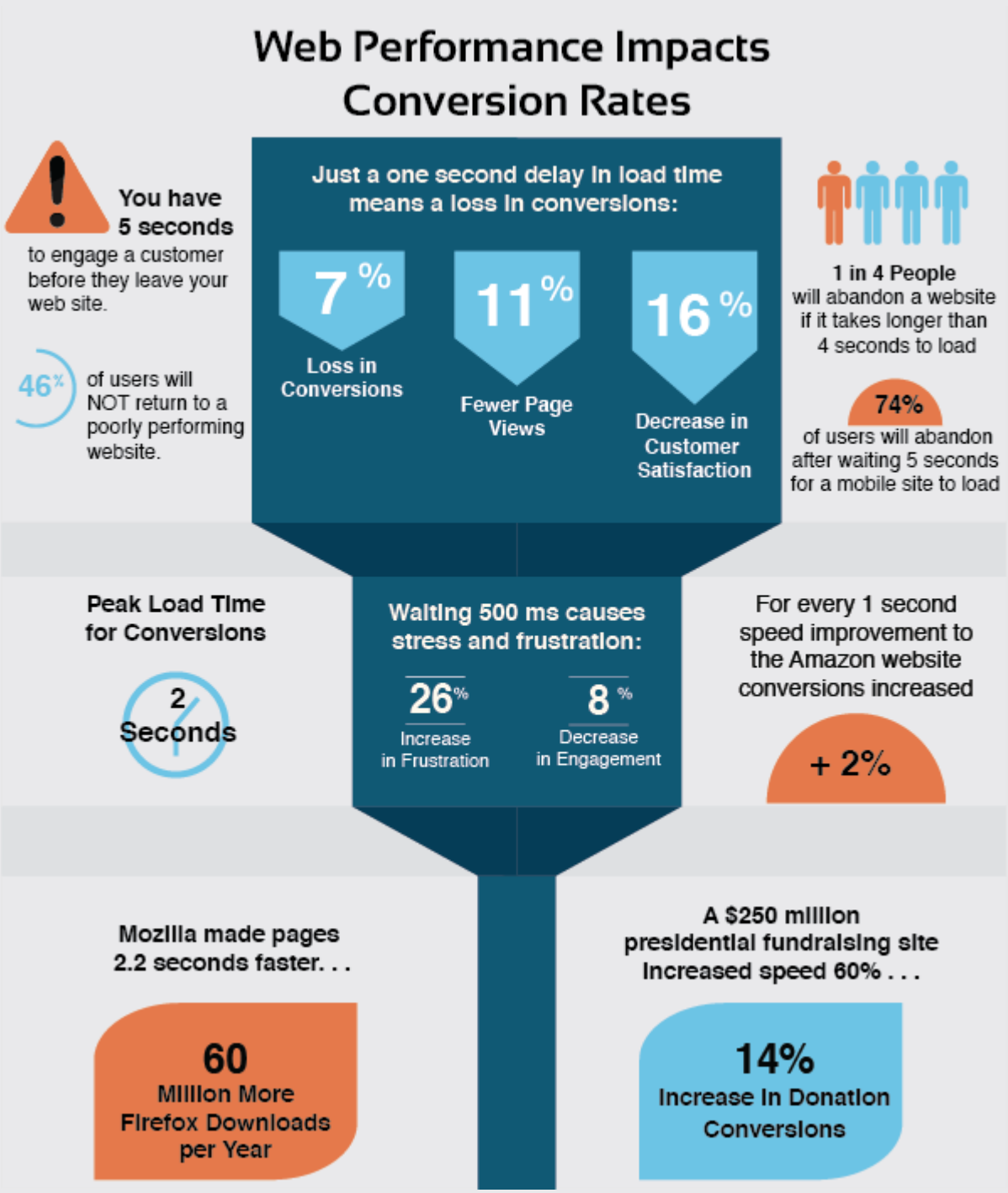
Improving Performance of Code I have seen in the wild

Pascal Knecht
E-Mail: pascal.knecht99@gmail.com
Twitter: @revenwo
GitHub: https://github.com/pascalknecht

About me

- Software Engineer based in Thun
- Currently doing my bachelor’s degree in Computer Science at the FHNW
- Working as a full-stack developer with WordPress for over 3 years


Web Performance Impacts Conversion Rates
You have 5 seconds to engage a customer before they leave your web site.
46% of users will NOT return to a poorly performing website.
Just a one second delay in load time means a loss in conversions:
7 % Loss in Conversions
11 % Fewer Page Views
16 % Decrease in Customer Satisfaction
1 in 4 People will abandon a website if it takes longer than 4 seconds to load
74% of users will abandon after waiting 5 seconds for a mobile site to load
Peak Load Time for Conversions
2 Seconds
Waiting 500 ms causes stress and frustration:
26% Increase in Frustration
8 % Decrease in Engagement
For every 1 second speed improvement to the Amazon website conversions increased
+ 2%
Mozilla made pages 2.2 seconds faster. . .
60 Million More Firefox Downloads per Year
A $250 million presidential fundraising site increased speed 60% . . .
14% Increase in Donation Conversions

What should we do to improve Performance?

- Optimize MYSQL-Queries that are generated by WordPress
- Optimize PHP-Code
- Optimize frontend assets loading (GZIP Compression, HTTP 2 Server Push, Browser Caching)
- Remove unnecessary code

Code

```php
<?php
function getCantons() {
    global $term;
    $args = array(
        'parent' => 0,
        'hide empty' => false,
        'taxonomy' => 'dl_canton_cat'
    );
    $taxChildren = get_terms('dl_canton_cat', $args);
    $cantons = array();
    foreach ($taxChildren as $childTerm) {
        $canton = array();
        $canton['classes'] = array();
        $postArgs = array(
            'post_type' => 'dl',
            'post_status' => 'publish',
            'posts_per_page' => -1,
            'tax_query' => array(
                array(
                    'taxonomy' => 'dl_canton_cat',
                    'field' => 'slug',
                    'terms' => array($childTerm->slug)
                )
            )
        );
        if ($term) {
            $postArgs['tax_query']['relation'] = 'AND';
            array_push($postArgs['tax_query'], array(
                'taxonomy' => 'dl_cat',
                'field' => 'slug',
                'terms' => array($term->slug),
                'include_children' => true
            ));
        }
        $posts = new WP_Query($postArgs);
        if ($posts->post_count > 0) {
            array_push($canton['classes'], 'has-items');
        } else {
            array_push($canton['classes'], 'no-items');
        }
        array_push($cantons, $canton);
    }
    return $cantons;
}

```

Solution: Option/Transient API

- Data – stored in wp_options
- Transients
  - Non-deterministic caching – have fallback code in place to generate the necessary data
  - Will use object cache if possible, otherwise MYSQL
  - Transients may expire before the expiration date (due to external object caches, or database upgrades)

Improved Code

```php
    $cached_option = get_option('canton_counter');
    foreach ($taxChildren as $childTerm) {
        $canton = array();
        $canton['classes'] = array();
        $count = $childTerm->post_count;
        if ($term) {
            $option_key = $term->slug;

            if(isset($cached_option[$option_key]) && $cached_option[$option_key] !== false) {
                $count = (int)$cached_option[$option_key];
            } else {
                $postArgs = array(
                    'post_type' => 'dl',
                    'post_status' => 'publish',
                    'posts_per_page' => -1,
                    'tax_query' => array(
                        'relation' => 'AND',
                        array(
                            'taxonomy' => 'dl_canton_cat',
                            'field' => 'slug',
                            'terms' => array($childTerm->slug)
                        ),
                        array(
                            'taxonomy' => 'dl_cat',
                            'field' => 'slug',
                            'terms' => array($term->slug),
                            'include_children' => true
                        )
                    )
                );
                $posts = new WP_Query($postArgs);
                $count = $posts->post_count;

                update_option($option_key, $count);
            }

```

Use For

- Options:
  - Data which does not expire and you want to have until you clear it
- Transients:
  - Caching data
  - Data which is heavy or slow to compute every time
  - Cache data which was fetched from an API

Code

```php
<?php
/*
archive.php
*/
$temp_query = $wp_query;
$wp_query = null;
$wp_query = new WP_Query(array(
    'post_type' => 'forum',
    'posts_per_page' => -1
));
while(have_posts()) {
    the_post();
    // Do something with post
}

$wp_query = null;
$wp_query = $temp_query;
wp_reset_postdata();

```

Solution: pre_get_posts Hook

- The pre_get_posts Hook can be used to alter the main query

Improved Code

```php
<?php
/*
functions.php
*/
function filter_forum_query($query) {
    if($query->is_main_query() && is_tax('forum_tax') && !is_admin()) {
        $query->set('post_type', 'forum');
        $query->set('posts_per_page', -1);
    }
}
add_action('pre_get_posts', 'filter_forum_query');

```

Use For

- Taxonomy and archive pages to not create a second loop to prevent getting the same data as in the main query
- Custom redirect rules to completely change the query

Code

```php
<?php
add_action( 'wp_enqueue_scripts', 'dst_enqueue_scripts', 0 );
function dst_enqueue_scripts() {

  wp_enqueue_script('jquery');

  wp_enqueue_script(
    'dstembed',
    '/dist/app.js',
    array('jquery'),
    false,
    false
  );
}

function dst_shortcode( $args ) {
  extract( shortcode_atts( array(
      'name'    => 'error',
  ), $args ) );

  $feed = new DST_Feed();

  return $feed->render( $args );
}

add_shortcode( 'dst', 'dst_shortcode' );

```

Solution: Conditional Enqueueing of Assets/Data

- Enqueue the script only when the shortcode is actually rendered
- Enqueue it in the footer since rendering of the header has already passed

Improved Code

```php
<?php
function dst_enqueue_scripts() {
  wp_enqueue_script(
    'dstembed',
    '/dist/app.js',
    array('jquery'),
    false,
    true
  );
}



function dst_shortcode( $args ) {
  extract( shortcode_atts( array(
      'name'    => 'error',
  ), $args ) );

  dst_enqueue_scripts();

  $feed = new DST_Feed();

  return $feed->render( $args );
}

add_shortcode( 'dst', 'dst_shortcode' );

```

Use For

- Assets/Data you only need in certain circumstances
- Javascript Files you use in shortcodes if you have HTTP2 available
- Other data like big JSON data you will need in Javascript

Code

```php
<?php
function post_to_pdf_action() {
    global $wpdb;
    $exported = 0;
    $post_ids = $_REQUEST['post'];
    foreach( $post_ids as $post_id ) {
        $posttitle = $wpdb->get_results("select * from $wpdb->posts where id='".$post_id."'",ARRAY_A);
        require_once("dompdf/dompdf_config.inc.php");
        $dompdf = new DOMPDF();
        $html = "<h2>".$posttitle[0]['post_title']."</h2>".$posttitle[0]['post_content'];
        $dompdf->load_html($html);
        $dompdf->render();
        $dompdf->stream($posttitle[0]['post_title'].".pdf");
    }
  exit();
}
add_action('save_post', 'post_to_pdf_action');

```

Solution: **WP Asynchronous Tasks**

- WP Asynchronous Tasks is a simple PHP-Class that is developed by TechCrunch
- This Class uses a clever technique to trigger your actions via wp_remote_post and hence will not interfere with user facing views
- https://github.com/techcrunch/wp-async-task

Improved Code

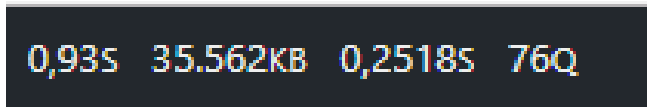
```php
<?php
// Rest stays as before
add_action('wp_async_save_post', 'post_to_pdf_action');

class Post_PDF_Async_Task extends WP_Async_Task {

    protected $action = 'save_post';

    /**
     *
     * @param array $data An array of data sent to the hook
     *
     * @return array
     */
    protected function prepare_data( $data ) {
        $post_id = $data[0];
        return array( 'post_id' => $post_id );
    }

    /**
     * Run the async task action
     */
    protected function run_action() {
        $post_id = $_POST['post_id'];
        $post = get_post( $post_id );
        do_action( "wp_async_$this->action", $post->ID, $post );
    }
}

```

Use For

- Resource Heavy Tasks
- Downloading images
- Making API Calls
- Storing data in Transients for Caching Purpose

Debugging the queries

- Query Monitor is a free plugin that shows you all queries that are made with a request
- You can profile individual plugins and their impact on performance
- https://de.wordpress.org/plugins/query-monitor/



Debugging the queries

Querys der Funktionen		
Abrufffunktion	SELECT	Zeit
WP_Term_Query->get_terms	25	0,2345
update_meta_cache	19	0,0068
WP_Post::get_instance	14	0,0034
WP_Query->get_posts	4	0,0029
get_option	6	0,0014
get_objects_in_term	1	0,0010
WP_Term::get_instance	1	0,0006
WC_Session_Handler->get_session	1	0,0003
WP_User::get_data_by	1	0,0003
WP_Query->set_found_posts	2	0,0002
WC_Data_Store_WP->read_meta	1	0,0002
_prime_post_caches	1	0,0002
Total	76	0,2518

Questions?

Slides

Scan the QR-Code

https://github.com/pascalknecht/wordcampbern