Turning Moodle web into a Progressive Web App (PWA)

June 2018

Mitxel Moriana

Developer @ 3iPunt Moodle Partner

What is a PWA ?

Web **A**pp stands for any web app, like **Moodle**!

P stands for progressive or “optionally enhanced”:

```
if browser supports this cool feature {
     use it
} else {
     // no problem! Do nothing
}
```

Typical examples of "progressiveness"

```
if ('serviceWorker' in navigator) {
    register our Service Worker
}

if ('serviceWorker' in navigator  && 'PushManager' in window) {
    we can use the Push API in our Service Worker
}

if ('caches' in window) {
    we can use the CacheStorage API to access storages/caches (i.e.: key-value pairs, request-
    responses pairs) shared by the window and the Service Worker
}
```

The Service Worker (I)

Not be confused with a *shared worker*

Event-driven script (written in JS) run (when needed) by the browser in the background (i.e.: in its own context/thread, not tied to a page, no direct DOM access).

Allows for the interception (on fetch event) of navigation/requests within its “scope” (usually, but not necessarily, the wwwroot of our site).

The Service Worker (II)

Allows for:

- Caching responses (and serving them) at client level
- Serving fallback responses to errors (or any custom condition)
- Pre-caching of responses/resources
- Background data synchronization (periodic data sync is "experimental)
- Push notifications

Ideally == when possible == if we do it right;

- Faster navigation
- Offline navigation
- Happier users (also, happier developers)

Turning Moodle into a PWA

cause study of ELE.ME (MPA)

Skeleton screen + PRPL pattern

(**Preload** critical resources, **Render** initial route, **Pre-cache** remaining routes, **Lazy-load** remaining routes).

Results:

- Loading time decreased by 11.6% across all pre-cached pages
- Loading time decreased on average by 6.35% across all pages.
- Time-to-consistently-interactive dropped to 4.93 seconds on a 3G network on first load

More info:

https://h5.ele.me/

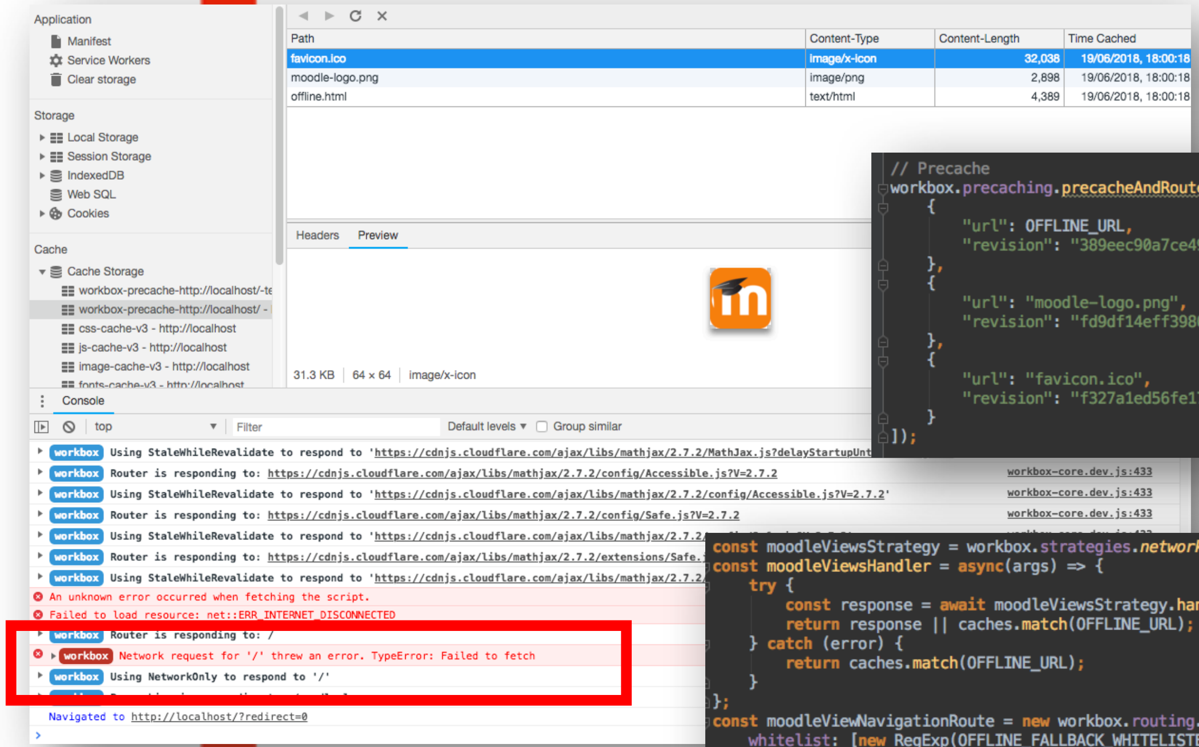
https://medium.com/elemefe/upgrading-ele-me-to-progressive-web-app-2a446832e509

3

Turning Moodle into a PWA

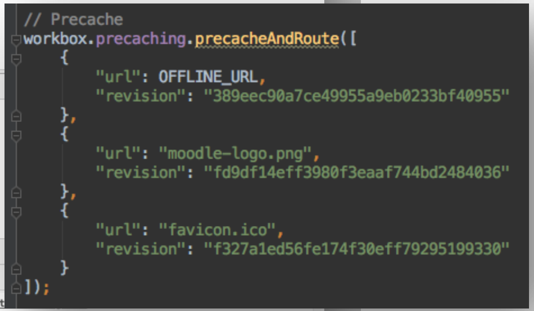
The offline fallback view

Pre-cache an **offline view** and **use it as an offline response fallback** (for non-cached view request/responses).

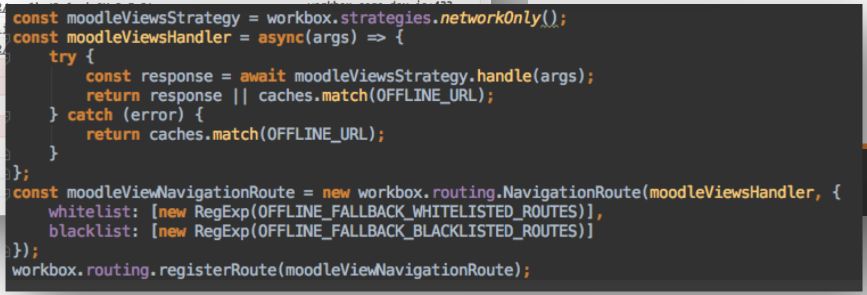


Path	Content-Type	Content-Length	Time Cached
favicon.ico	image/x-icon	32,038	19/06/2018, 18:00:18
moodle-logo.png	image/png	2,898	19/06/2018, 18:00:18
offline.html	text/html	4,389	19/06/2018, 18:00:18

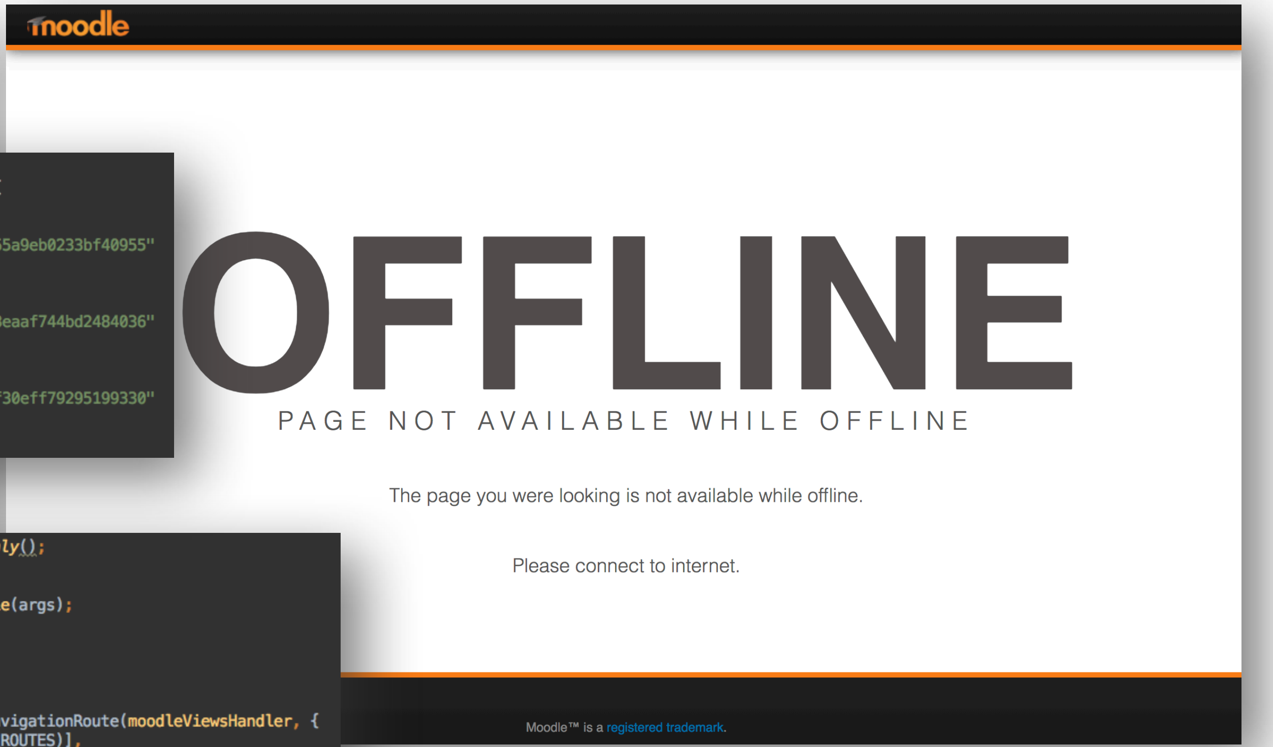
```
workbox Using StaleWhileRevalidate to respond to 'https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2/MathJax.js?delayStartupUnt
workbox Router is responding to: https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2/config/Accessible.js?V=2.7.2        workbox-core.dev.js:433
workbox Using StaleWhileRevalidate to respond to 'https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2/config/Accessible.js?V=2.7.2'   workbox-core.dev.js:433
workbox Router is responding to: https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2/config/Safe.js?V=2.7.2              workbox-core.dev.js:433
workbox Using StaleWhileRevalidate to respond to 'https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2
workbox Router is responding to: https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2/extensions/Safe.
workbox Using StaleWhileRevalidate to respond to 'https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.2
An unknown error occurred when fetching the script.
Failed to load resource: net::ERR_INTERNET_DISCONNECTED
workbox Router is responding to: /
workbox Network request for '/' threw an error. TypeError: Failed to fetch
workbox Using NetworkOnly to respond to '/'
Navigated to http://localhost/?redirect=0
```



```
// Precache
workbox.precaching.precacheAndRoute([
    {
        "url": OFFLINE_URL,
        "revision": "389eec90a7ce49955a9eb0233bf40955"
    },
    {
        "url": "moodle-logo.png",
        "revision": "fd9df14eff3980f3eaaf744bd2484036"
    },
    {
        "url": "favicon.ico",
        "revision": "f327a1ed56fe174f30eff79295199330"
    }
]);
```



```
const moodleViewsStrategy = workbox.strategies.networkOnly();
const moodleViewsHandler = async(args) => {
    try {
        const response = await moodleViewsStrategy.handle(args);
        return response || caches.match(OFFLINE_URL);
    } catch (error) {
        return caches.match(OFFLINE_URL);
    }
};
const moodleViewNavigationRoute = new workbox.routing.NavigationRoute(moodleViewsHandler, {
    whitelist: [new RegExp(OFFLINE_FALLBACK_WHITELISTED_ROUTES)],
    blacklist: [new RegExp(OFFLINE_FALLBACK_BLACKLISTED_ROUTES)]
});
workbox.routing.registerRoute(moodleViewNavigationRoute);
```



moodle

OFFLINE

PAGE NOT AVAILABLE WHILE OFFLINE

The page you were looking is not available while offline.

Please connect to internet.

Moodle™ is a registered trademark.

3

Turning Moodle into a PWA

CACHING STATIC CONTENT (I)

Caching JS, CSS, images, fonts… when requested at least once.

Moodle serves optimized static content using specific scripts:
theme/styles.php – serves "*the one huge CSS of each theme*"
theme/font.php – serves "*the fonts used in CSS*"
theme/yui_combo.php – serves "*yui Javascript and CSS*"
theme/image.php – serves "*the one theme and plugin images*"
lib/javascript.php – serves "*optimised JS*"
lib/requirejs.php – serves "*optimised JS for RequireJS*"

The URLs to these scripts contain theme/script "versioning" parameters:
New versions => different request URL => no conflict updated vs cached

Turning Moodle into a PWA

CACHING STATIC CONTENT (II)

Choosing the **“right” serving and caching strategy/recipe**:

Does the URL (including parameters) always returns the same content?

```
if yes {
    Use “cache only” (if pre-cached!) or “cache first” serving strategies (and dynamically cache them)
} else if not always, but to load the most recent version IS NOT essential {
    Use “stale while revalidate” serving/caching strategy (serve cached if exists but check in background for
    an update and cache updated resource when needed)
} else if not always, but to load the most recent version IS essential {
    Use “network first” serving strategy (and dynamically cache newest version, but use them only as a
    fallback)
} else if no and never {
    Are you sure it is static content? Why do we need to cache it?
}
```

Turning Moodle into a PWA

caching static content (III)

Lighthouse audit

Mobile emulated, 3G throttled “second visit”, i.e. with browser cache preserved.

With Service Worker serving cached static content:
1260 ms (first meaningful paint)
Without Service Worker:
1430 ms (first meaningful paint)

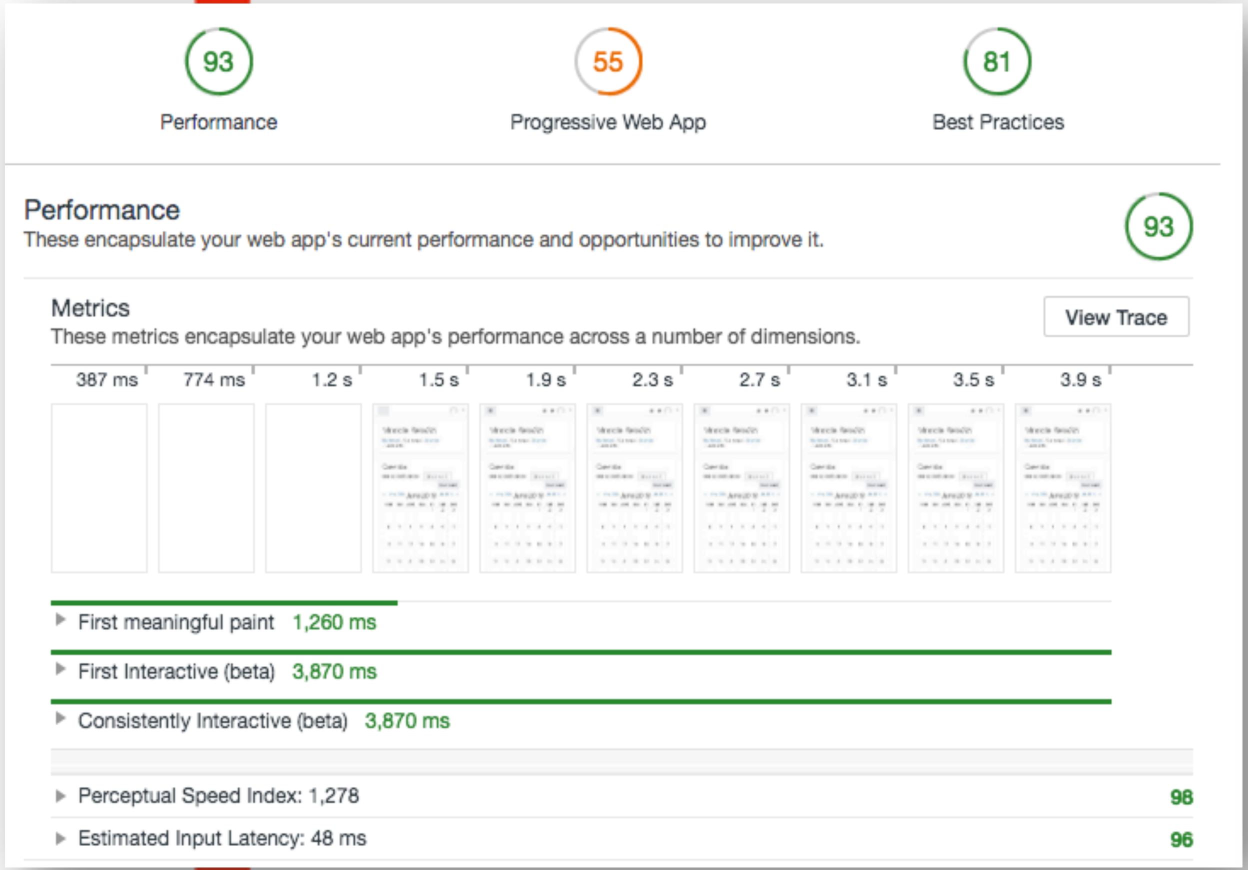
“First meaningful paint” was 170 ms faster = improvement of ~10% in the “user-perceived loading experience”

3

Turning Moodle into a PWA

caching static content (IV)

WITH SW



93 Performance

55 Progressive Web App

81 Best Practices

Performance 93

These encapsulate your web app's current performance and opportunities to improve it.

Metrics

These metrics encapsulate your web app's performance across a number of dimensions.

View Trace

387 ms | 774 ms | 1.2 s | 1.5 s | 1.9 s | 2.3 s | 2.7 s | 3.1 s | 3.5 s | 3.9 s

First meaningful paint 1,260 ms

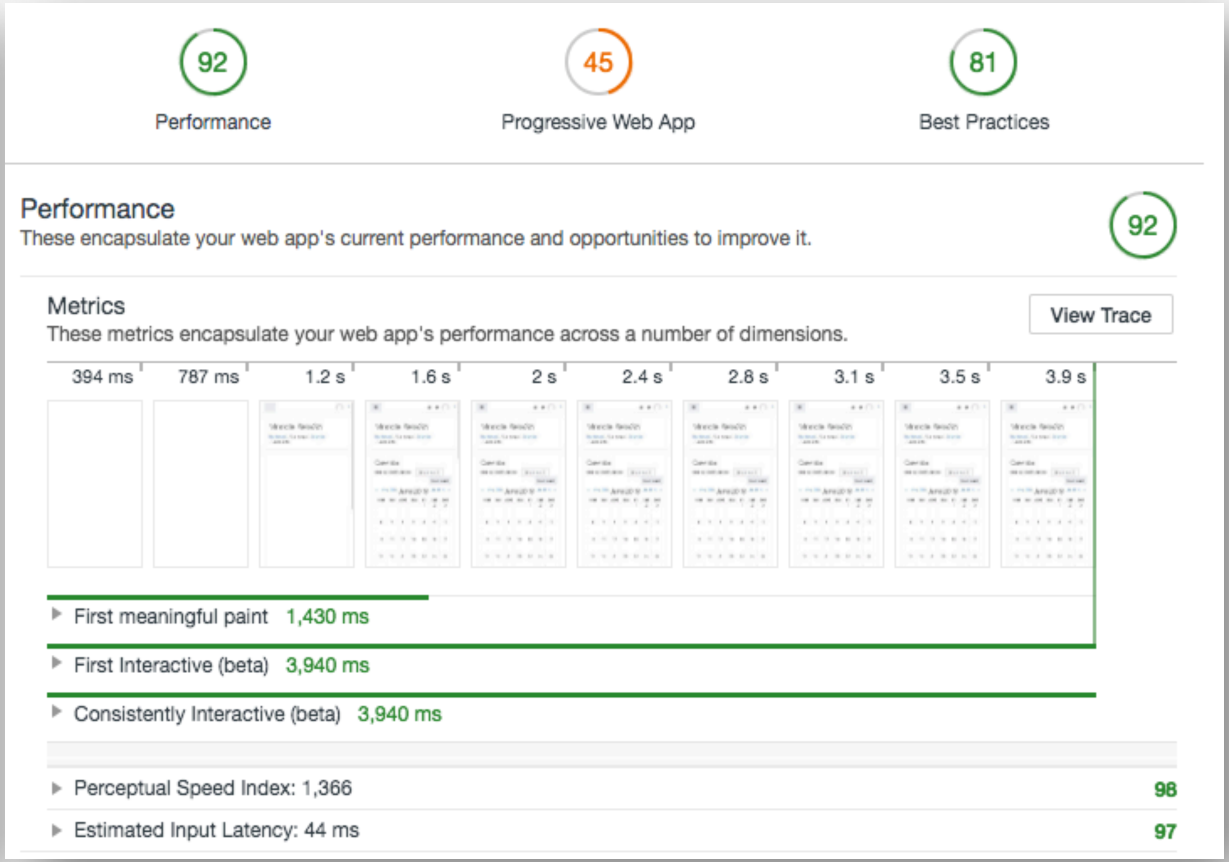
First Interactive (beta) 3,870 ms

Consistently Interactive (beta) 3,870 ms

Perceptual Speed Index: 1,278 98

Estimated Input Latency: 48 ms 96

WITHOUT SW



92 Performance

45 Progressive Web App

81 Best Practices

Performance 92

These encapsulate your web app's current performance and opportunities to improve it.

Metrics

These metrics encapsulate your web app's performance across a number of dimensions.

View Trace

394 ms | 787 ms | 1.2 s | 1.6 s | 2 s | 2.4 s | 2.8 s | 3.1 s | 3.5 s | 3.9 s

First meaningful paint 1,430 ms

First Interactive (beta) 3,940 ms

Consistently Interactive (beta) 3,940 ms

Perceptual Speed Index: 1,366 98

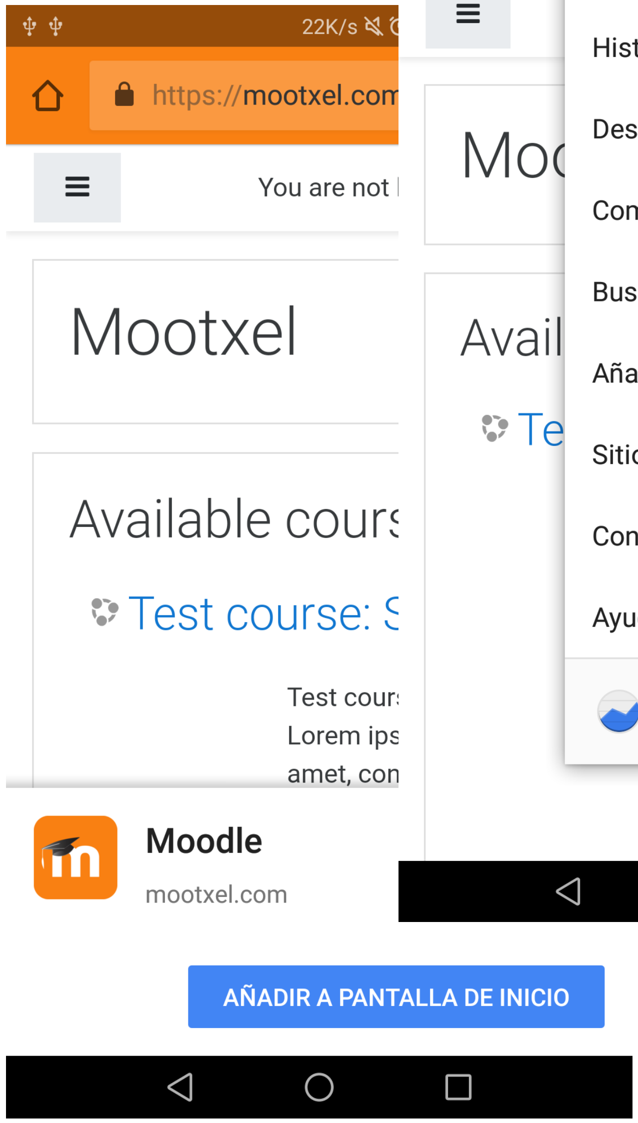
Estimated Input Latency: 44 ms 97

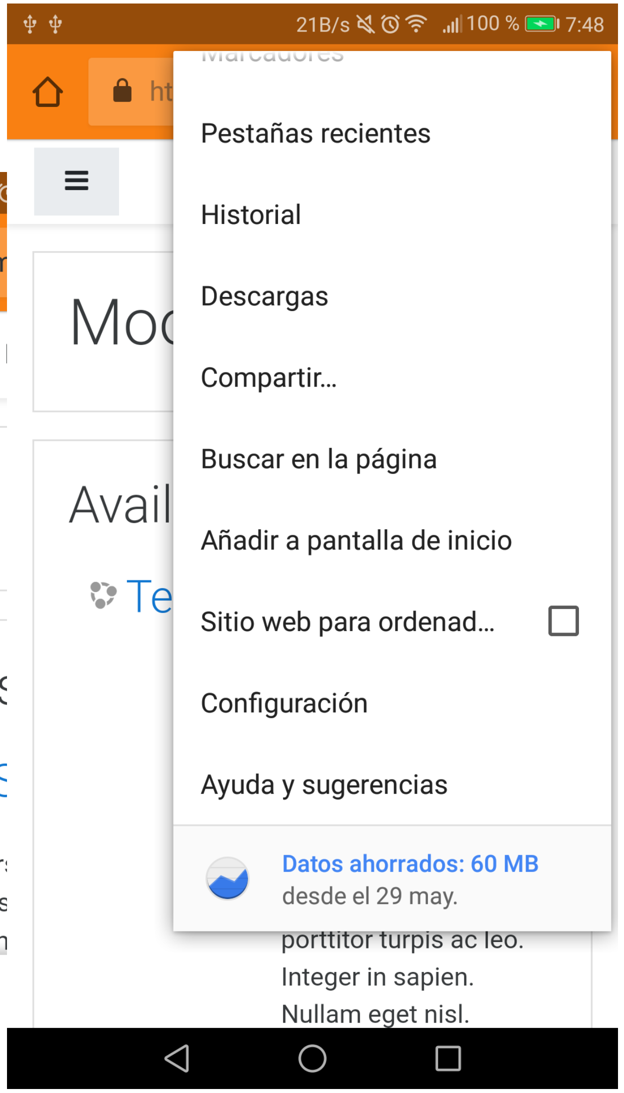
3

Turning Moodle into a PWA

home screen & manifest.json (I)

manifest.json
{
app **name** and **short_name**,
icons (icons and splash screens),
related_applications (web, play store…),
start_url (starting url, it could be the root /),
display (standalone = "appish", browser…),
scope (scope url, like for example the root /)
background and theme _color (#f98012),
…
}
+ <link rel="manifest" href="/manifest.json">
+ convenient meta tags…
+ https
+ ¿use conditions?
=
Browser prompts the user to install the home screen / Add to home screen menu option appears





Turning Moodle into a PWA
home screen & manifest.json (II)

You can set the *manifest.json* in a way that the user can be prompted to install the mobile app from the stores (instead of as a “link” to the PWA)

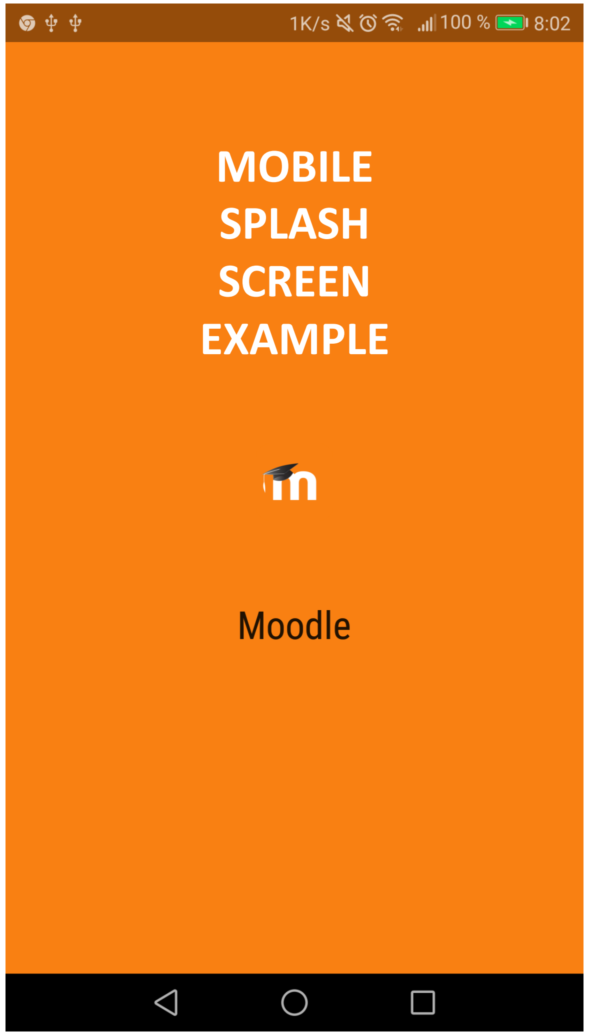
¿More convenient for customers that have their own Moodle Mobile app?

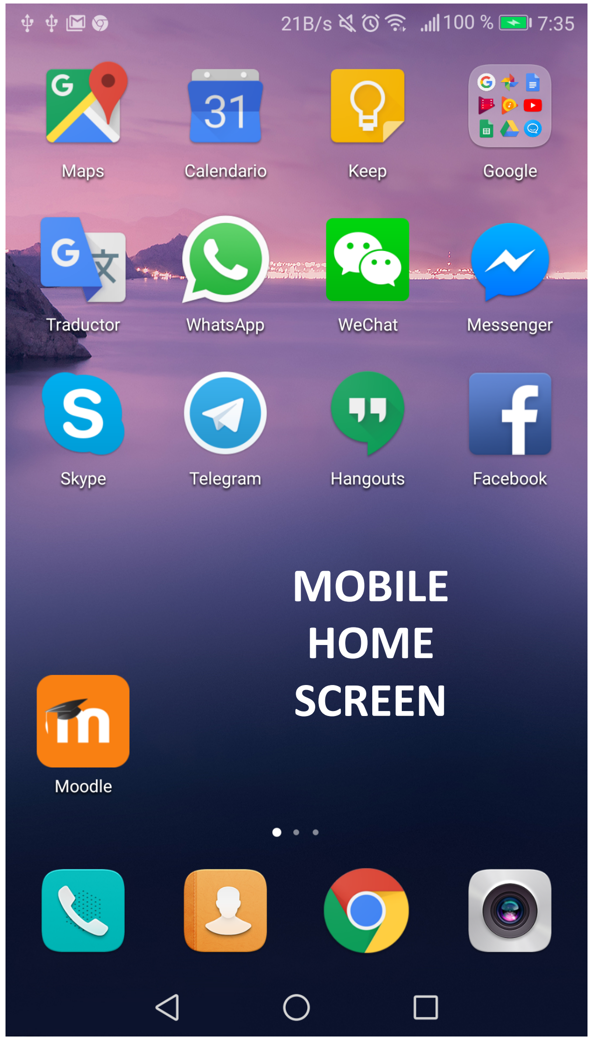
```
{
        …
        related_applications: [{ platform: web }, { platform: play, id: com… }]
        prefer_related_applications: true
}
```

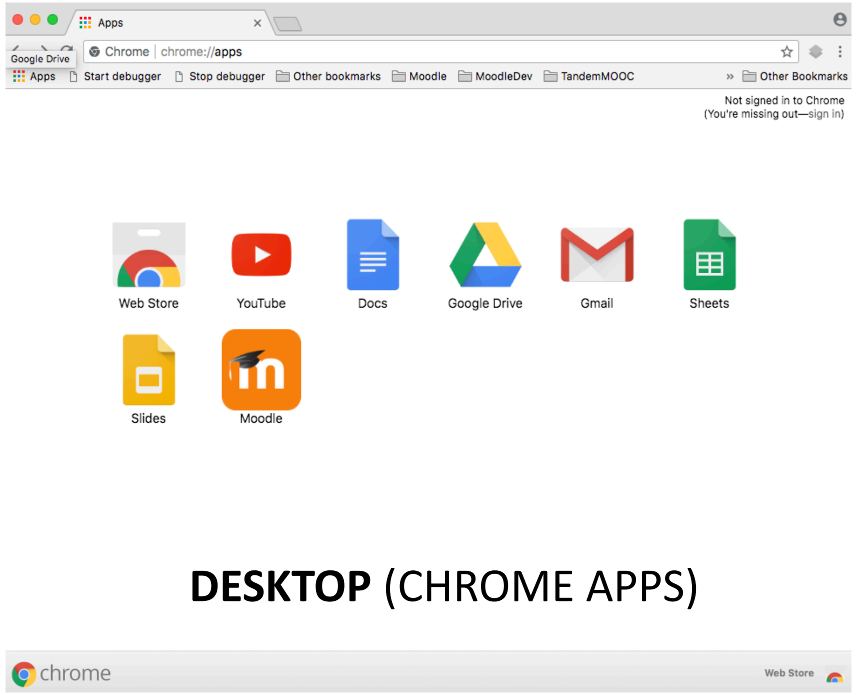
3

Turning Moodle into a PWA

HOME SCREEN & MANIFEST.JSON (III)







DESKTOP (CHROME APPS)

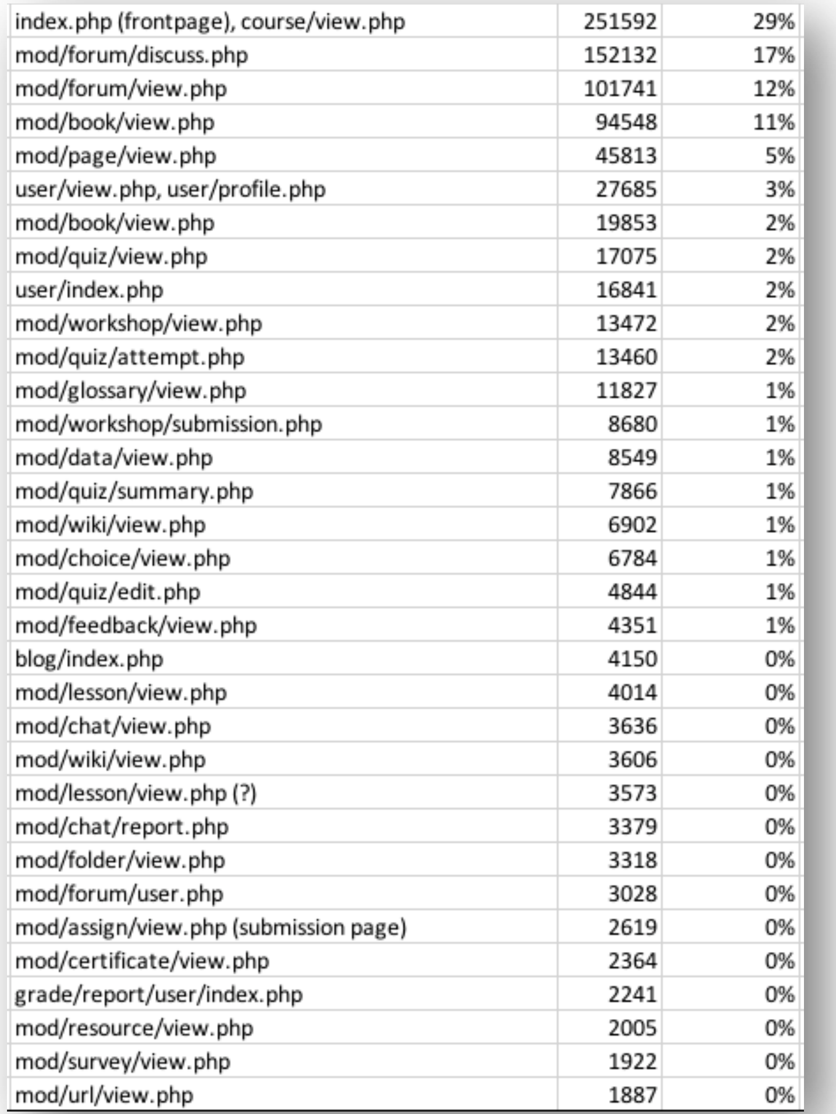
Turning Moodle into a PWA

Caching page views (I)

One idea…
Caching the most used views in a given Moodle instance…

Which views are the most “used” in a given Moodle instance?

ANALYTICS can tell us →



index.php (frontpage), course/view.php	251592	29%
mod/forum/discuss.php	152132	17%
mod/forum/view.php	101741	12%
mod/book/view.php	94548	11%
mod/page/view.php	45813	5%
user/view.php, user/profile.php	27685	3%
mod/book/view.php	19853	2%
mod/quiz/view.php	17075	2%
user/index.php	16841	2%
mod/workshop/view.php	13472	2%
mod/quiz/attempt.php	13460	2%
mod/glossary/view.php	11827	1%
mod/workshop/submission.php	8680	1%
mod/data/view.php	8549	1%
mod/quiz/summary.php	7866	1%
mod/wiki/view.php	6902	1%
mod/choice/view.php	6784	1%
mod/quiz/edit.php	4844	1%
mod/feedback/view.php	4351	1%
blog/index.php	4150	0%
mod/lesson/view.php	4014	0%
mod/chat/view.php	3636	0%
mod/wiki/view.php	3606	0%
mod/lesson/view.php (?)	3573	0%
mod/chat/report.php	3379	0%
mod/folder/view.php	3318	0%
mod/forum/user.php	3028	0%
mod/assign/view.php (submission page)	2619	0%
mod/certificate/view.php	2364	0%
grade/report/user/index.php	2241	0%
mod/resource/view.php	2005	0%
mod/survey/view.php	1922	0%
mod/url/view.php	1887	0%

3

Turning Moodle into a PWA
Caching page views (II)

PROBLEM!

Many routes serve different content when authenticated / not-authenticated.

Eg:

- While not authenticated all routes “serves” (redirect to) the login page.
- User-specific content (e.g.: same course route, different user/user role).

We could evaluate the session cookie credentials in our serving strategies…

But do we really want to cache the response (from an “authenticated” context) and make it publicly available to anyone with access to the browser?

¿App shell to the rescue?

TURNING MOODLE INTO A PWA
CACHING PAGE VIEWS (III)

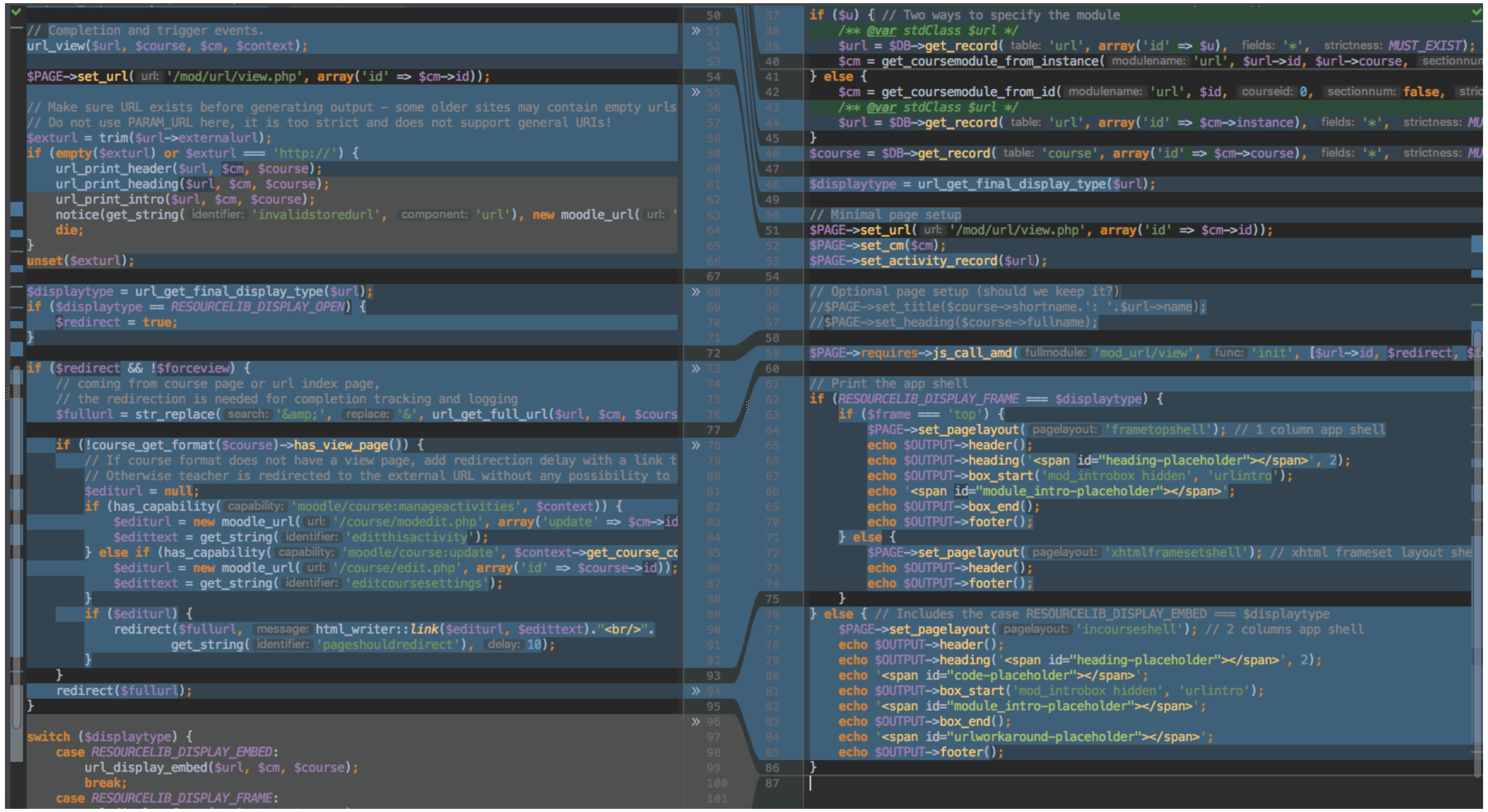
The App Shell approach (as I understand it) -> Refactor party!

- Remove all user-specific / authentication-needed data from ALL THE VIEWS AND LAYOUTS
- Render the view specific information using asynchronously called web services (with the proper login and capabilities checks)
- Render the layout “user-related” elements and information the same way (e.g.: the user menu, the course navigation panel…)

Turning Moodle into a PWA

Caching page views (IV)

Refactoring party with the mod_url view.php -> /mod/url/view.php



```php
// Completion and trigger events.
url_view($url, $course, $cm, $context);

$PAGE->set_url( url: '/mod/url/view.php', array('id' => $cm->id));

// Make sure URL exists before generating output - some older sites may contain empty urls
// Do not use PARAM_URL here, it is too strict and does not support general URIs!
$exturl = trim($url->externalurl);
if (empty($exturl) or $exturl === 'http://') {
    url_print_header($url, $cm, $course);
    url_print_heading($url, $cm, $course);
    url_print_intro($url, $cm, $course);
    notice(get_string( identifier: 'invalidstoredurl', component: 'url'), new moodle_url( url: '
    die;
}
unset($exturl);

$displaytype = url_get_final_display_type($url);
if ($displaytype == RESOURCELIB_DISPLAY_OPEN) {
    $redirect = true;
}

if ($redirect && !$forceview) {
    // coming from course page or url index page,
    // the redirection is needed for completion tracking and logging
    $fullurl = str_replace( search: '&amp;', replace: '&', url_get_full_url($url, $cm, $cours

    if (!course_get_format($course)->has_view_page()) {
        // If course format does not have a view page, add redirection delay with a link t
        // Otherwise teacher is redirected to the external URL without any possibility to
        $editurl = null;
        if (has_capability( capability: 'moodle/course:manageactivities', $context)) {
            $editurl = new moodle_url( url: '/course/modedit.php', array('update' => $cm->id
            $edittext = get_string( identifier: 'editthisactivity');
        } else if (has_capability( capability: 'moodle/course:update', $context->get_course_cc
            $editurl = new moodle_url( url: '/course/edit.php', array('id' => $course->id));
            $edittext = get_string( identifier: 'editcoursesettings');
        }
        if ($editurl) {
            redirect($fullurl, message: html_writer::link($editurl, $edittext)."<br/>".
                    get_string( identifier: 'pageshouldredirect'), delay: 10);
        }
    }
    redirect($fullurl);
}

switch ($displaytype) {
    case RESOURCELIB_DISPLAY_EMBED:
        url_display_embed($url, $cm, $course);
        break;
    case RESOURCELIB_DISPLAY_FRAME:
```

```php
if ($u) { // Two ways to specify the module
    /** @var stdClass $url */
    $url = $DB->get_record( table: 'url', array('id' => $u), fields: '*', strictness: MUST_EXIST);
    $cm = get_coursemodule_from_instance( modulename: 'url', $url->id, $url->course, sectionnum
} else {
    $cm = get_coursemodule_from_id( modulename: 'url', $id, courseid: 0, sectionnum: false, stric
    /** @var stdClass $url */
    $url = $DB->get_record( table: 'url', array('id' => $cm->instance), fields: '*', strictness: MU
}
$course = $DB->get_record( table: 'course', array('id' => $cm->course), fields: '*', strictness: MU

$displaytype = url_get_final_display_type($url);

// Minimal page setup
$PAGE->set_url( url: '/mod/url/view.php', array('id' => $cm->id));
$PAGE->set_cm($cm);
$PAGE->set_activity_record($url);

// Optional page setup (should we keep it?)
//$PAGE->set_title($course->shortname.': '.$url->name);
//$PAGE->set_heading($course->fullname);

$PAGE->requires->js_call_amd( fullmodule: 'mod_url/view', func: 'init', [$url->id, $redirect, $f

// Print the app shell
if (RESOURCELIB_DISPLAY_FRAME === $displaytype) {
    if ($frame === 'top') {
        $PAGE->set_pagelayout( pagelayout: 'frametopshell'); // 1 column app shell
        echo $OUTPUT->header();
        echo $OUTPUT->heading('<span id="heading-placeholder"></span>', 2);
        echo $OUTPUT->box_start('mod_introbox hidden', 'urlintro');
        echo '<span id="module_intro-placeholder"></span>';
        echo $OUTPUT->box_end();
        echo $OUTPUT->footer();
    } else {
        $PAGE->set_pagelayout( pagelayout: 'xhtmlframesetshell'); // xhtml frameset layout she
        echo $OUTPUT->header();
        echo $OUTPUT->footer();
    }
} else { // Includes the case RESOURCELIB_DISPLAY_EMBED === $displaytype
    $PAGE->set_pagelayout( pagelayout: 'incourseshell'); // 2 columns app shell
    echo $OUTPUT->header();
    echo $OUTPUT->heading('<span id="heading-placeholder"></span>', 2);
    echo '<span id="code-placeholder"></span>';
    echo $OUTPUT->box_start('mod_introbox hidden', 'urlintro');
    echo '<span id="module_intro-placeholder"></span>';
    echo $OUTPUT->box_end();
    echo '<span id="urlworkaround-placeholder"></span>';
    echo $OUTPUT->footer();
}
```

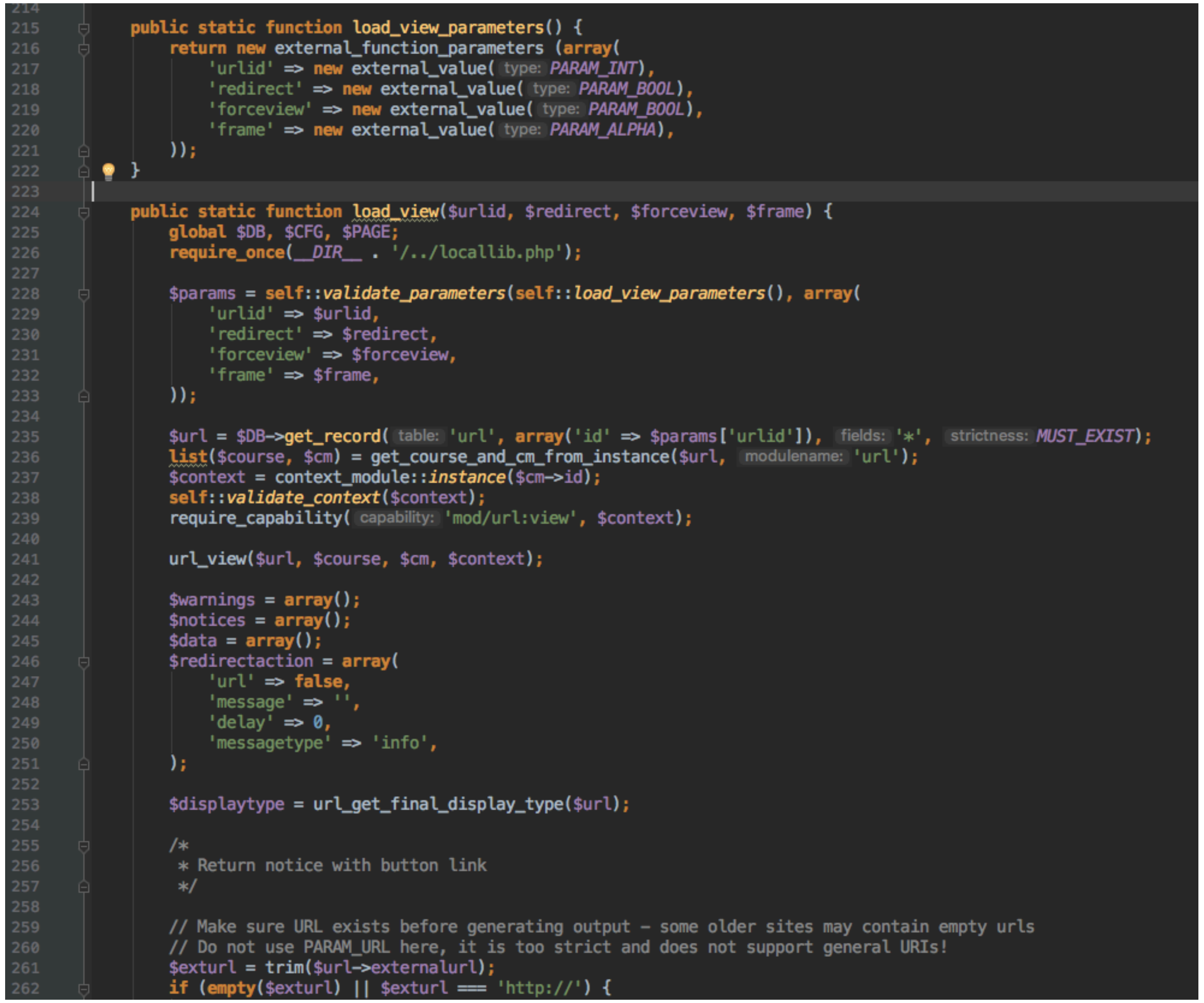
3

Turning Moodle into a PWA

Caching page views (V)

+ web service to serve the mod_url view.php content (all “view” cases)

mod/url/classes/external.php
load_view



```php
    public static function load_view_parameters() {
        return new external_function_parameters (array(
            'urlid' => new external_value( type: PARAM_INT),
            'redirect' => new external_value( type: PARAM_BOOL),
            'forceview' => new external_value( type: PARAM_BOOL),
            'frame' => new external_value( type: PARAM_ALPHA),
        ));
    }

    public static function load_view($urlid, $redirect, $forceview, $frame) {
        global $DB, $CFG, $PAGE;
        require_once(__DIR__ . '/../locallib.php');

        $params = self::validate_parameters(self::load_view_parameters(), array(
            'urlid' => $urlid,
            'redirect' => $redirect,
            'forceview' => $forceview,
            'frame' => $frame,
        ));

        $url = $DB->get_record( table: 'url', array('id' => $params['urlid']), fields: '*', strictness: MUST_EXIST);
        list($course, $cm) = get_course_and_cm_from_instance($url, modulename: 'url');
        $context = context_module::instance($cm->id);
        self::validate_context($context);
        require_capability( capability: 'mod/url:view', $context);

        url_view($url, $course, $cm, $context);

        $warnings = array();
        $notices = array();
        $data = array();
        $redirectaction = array(
            'url' => false,
            'message' => '',
            'delay' => 0,
            'messagetype' => 'info',
        );

        $displaytype = url_get_final_display_type($url);

        /*
         * Return notice with button link
         */

        // Make sure URL exists before generating output - some older sites may contain empty urls
        // Do not use PARAM_URL here, it is too strict and does not support general URIs!
        $exturl = trim($url->externalurl);
        if (empty($exturl) || $exturl === 'http://') {
```

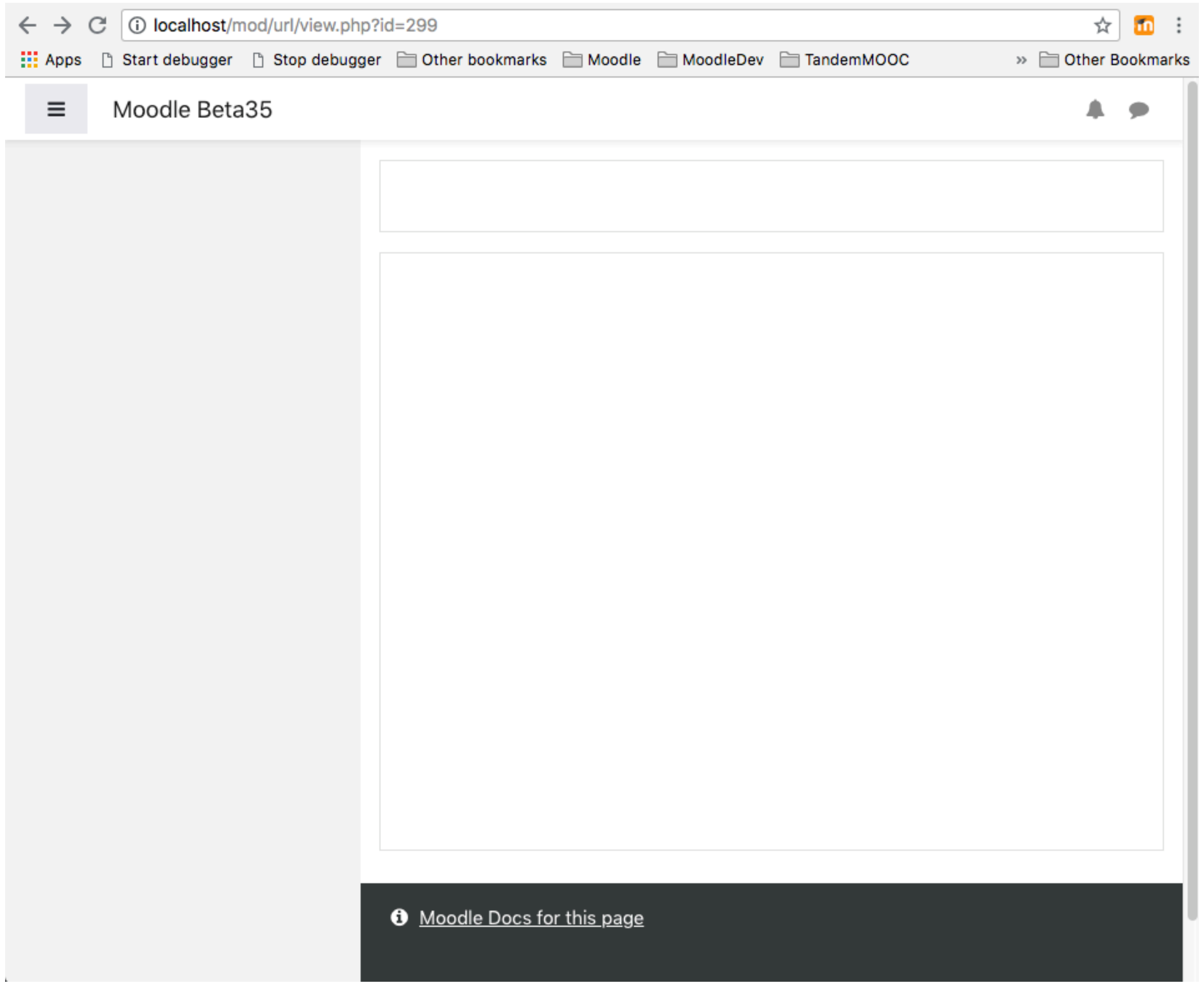
Turning Moodle into a PWA

Caching page views (VI)

App-shelled mod_url view >>>

User, session… related information removed from the views
No user menu, no navigation menu, no footer user-related links, no mod_url view-specific data…

This is our "app shell", let's cache this! (stalewhilereval? cache first?)



3

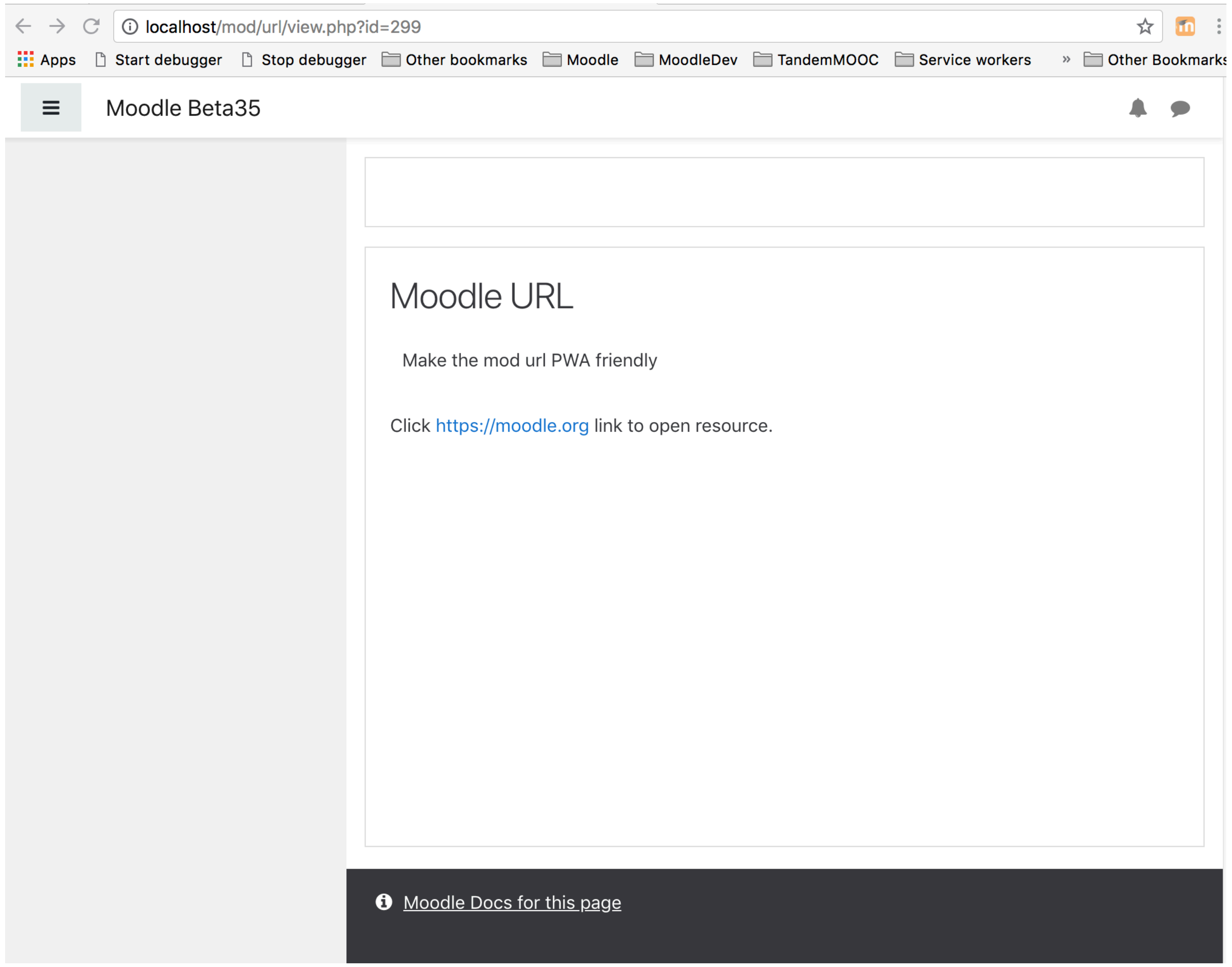
Turning Moodle into a PWA

Caching page views (VII)

Then we load asynchronously the actual content behind auth and capabilities checks

This is the mod_url view content loaded from a web service.

Do the same with the information removed from the layout! -> web service + async loading



Moodle URL

Make the mod url PWA friendly

Click https://moodle.org link to open resource.

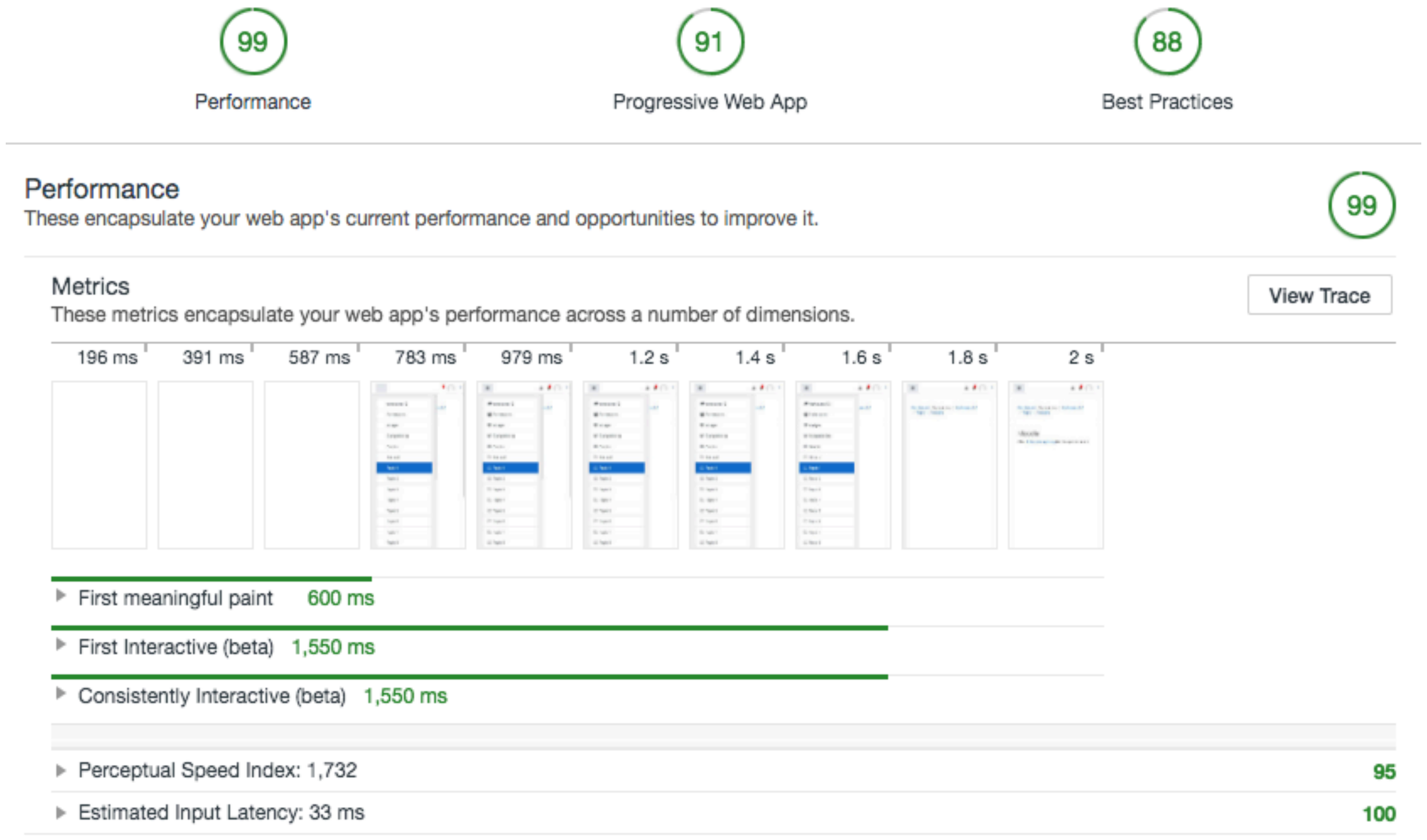
Turning Moodle into a PWA

Caching page views (VIII)

Lighthouse audit (without refactoring the layout, just the mod_url view)

App shell approach:
“First meaningul paint”
~600 ms (half the time)

Caveats:
The actual content
”appears” much later
(+ 1 sec at least) (on load)



99 Performance
91 Progressive Web App
88 Best Practices

Performance 99
These encapsulate your web app's current performance and opportunities to improve it.

Metrics
These metrics encapsulate your web app's performance across a number of dimensions.

View Trace

196 ms | 391 ms | 587 ms | 783 ms | 979 ms | 1.2 s | 1.4 s | 1.6 s | 1.8 s | 2 s

First meaningful paint 600 ms
First Interactive (beta) 1,550 ms
Consistently Interactive (beta) 1,550 ms

Perceptual Speed Index: 1,732 95
Estimated Input Latency: 33 ms 100

Turning Moodle into a PWA
Adding the Service Worker to the Moodle core?

- Allow for plugins to define their scope and tie plugin routes to different caching strategies / precaching (ServiceWorker API ?).
- Create a php script that builds and serves a “revisioned” sw.js collecting and including all those plugin definitions across all the Moodle instance plugins.
- Add admin settings to easily enable/disable the inclusion of the service worker.
- Add admin settings to easily include JS code that unregisters previously added service workers and/or force clients to clear their storages and caches (in case something went wrong…).

THANKS TO (POWERED BY)...

- 3iPunt: **Pau Plana, Ebrahim Mesleh & Antoni Bertran**
- 3iPunt "Moodle Team":

  Eva Pereira, Raúl Martínez & Roser Pruaño
- All developers that have been documenting their experiences with Service Workers since 2015.