

PWA with Symfony 4

Aurelijus Banelis



VilniusPHP 0x43

2018-06-07



Aurelijus Banelis

Software developer

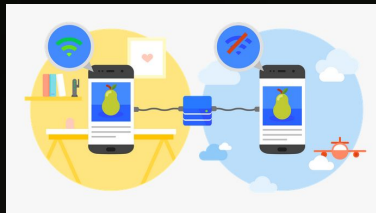
aurelijus.banelis.lt

aurelijus@banelis.lt

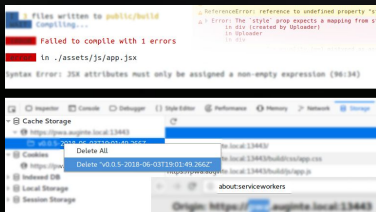
PGP 0x320205E7**539B6203**
130D C446 1F1A 2E50 D6E3
3DA8 3202 05E7 539B 6203



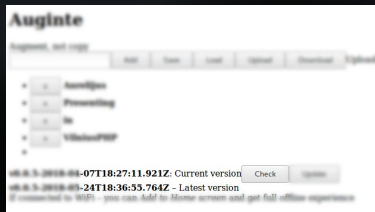
Progressive Web Apps from Symfony 4 developer perspective



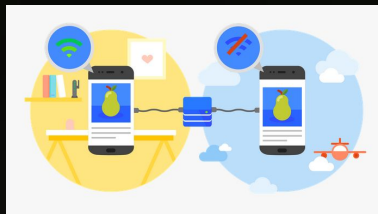
Intro into PWA



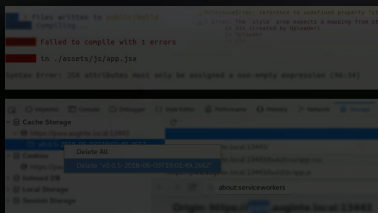
Development environment



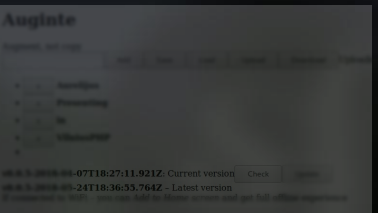
Final thoughts



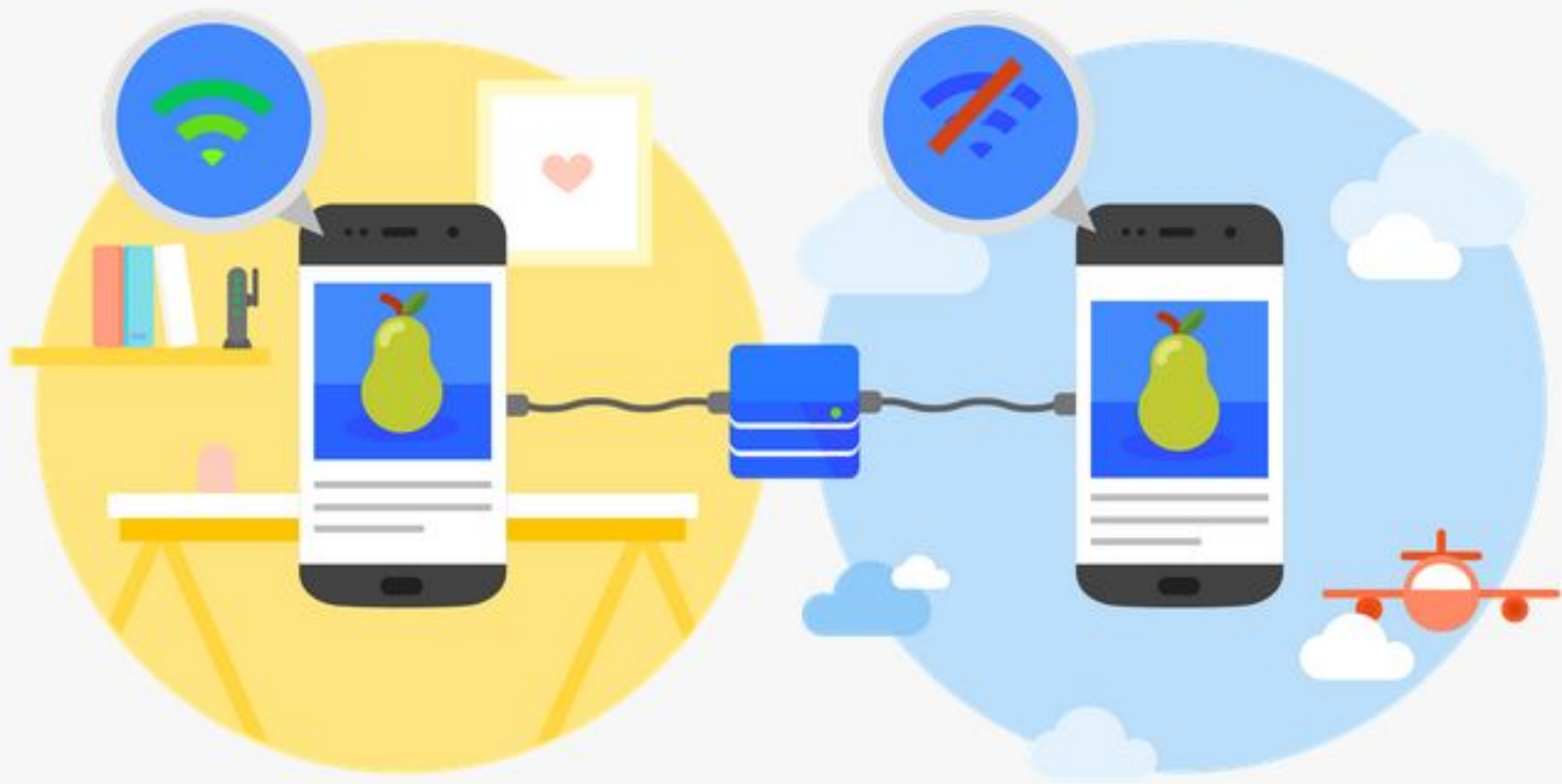
Intro into PWA



Development environment



Final thoughts



Application

 Manifest

Service Workers

 Clear storage

Storage

▶  Local Storage

▶  Session Storage

▶ IndexedDB

 Web SQL

▶  Cookies


Service Workers

☐ Offline ☐ Update on reload ☐ Bypass for network

pwa.auginte.com

Source [service-worker.js](#)

Received 5/24/2018, 8:42:42 PM

Status  #312 activated and is running [stop](#)

Clients <https://pwa.auginte.com/> [focus](#)



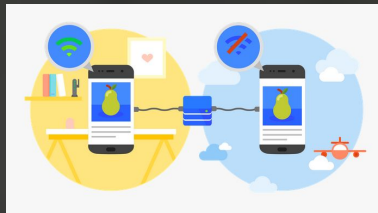
Size	Time	Volume
(from Service...	6 ms	
(from Service...	0 ms	

Cache the site assets

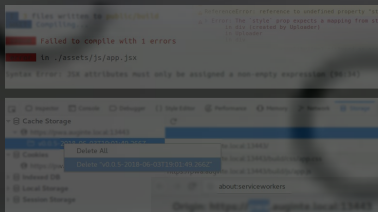
When the service worker is registered, an install event is triggered the first time the user visits the application. In the `install` event handler, we will cache all the assets that are needed for the application.

The code below must NOT be used in production, it covers only the most basic use cases and can put yourself into a state where your app shell will never update. Be sure to review the section on the pitfalls of this implementation and how to avoid them.

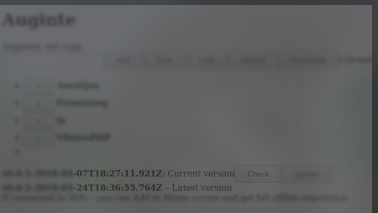
When the service worker is fired, it should open the `caches` object and populate it with the assets of the App Shell. Create a file called `service-worker.js` in your application root folder (where the `first-pwapp-master/work` directory). This file must live in the application root because



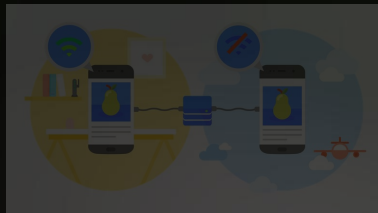
Intro into PWA



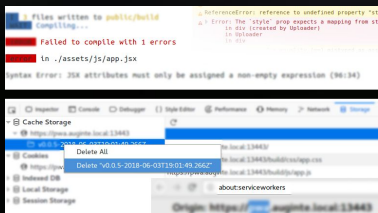
Development environment



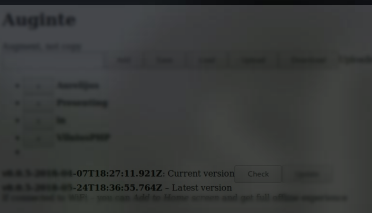
Final thoughts



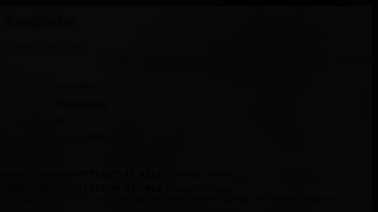
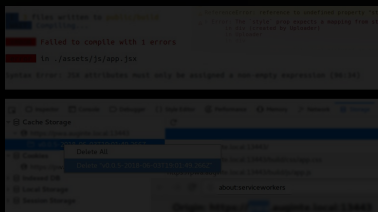
Intro into PWA



Development environment



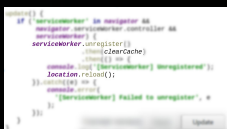
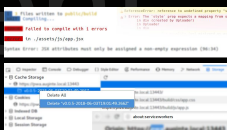
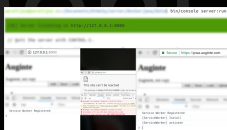
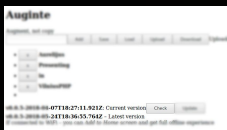
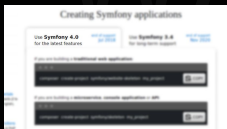
Final thoughts



Intro into PWA

Development environment

Final thoughts



What we want?

Validate cache

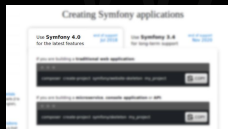
Backend

Frontend

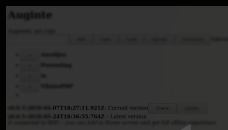
Clear cache



Intro into PWA



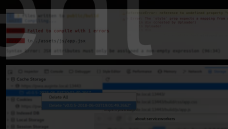
What we want?



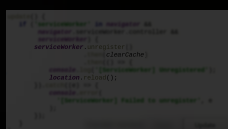
Validate cache



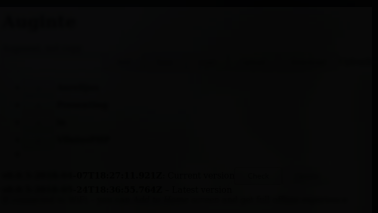
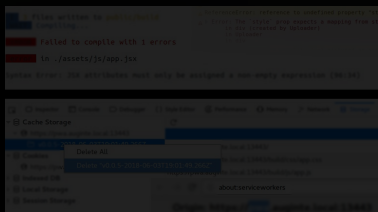
Backend



Frontend



Clear cache



Development environment

Final thoughts

Creating Symfony applications

Use **Symfony 4.0**
for the latest features

end of support
Jul 2018

Use **Symfony 3.4**
for long-term support

end of support
Nov 2018

If you are building a traditional web application:

3.4.5

```
composer create-project symfony/website-skeleton my_project
```



If you are building a microservice, console application or API:

3.4.5

```
composer create-project symfony/skeleton my_project
```



Enabling React.js

Using React? Make sure you have React installed, along with the [latest version](#) of [React](#).

1.0.0-rc.1

Get the code

1.0.0-rc.1

- 1. yarn add --dev @babel/preset-react
- 2. yarn add react react-dom @babel/core

Enable react in your webpack config.js:

```
1 // webpack.config.js
2 // ...
3
4 Encore
5   // ...
6   .enableReact()
7 }
```

Auginte

Augment, not copy

Get

Free

Test

Upgrade

Download

Upload

-  **Auginte**
-  **Presenting**
-  **in**
-  **Videos/PDF**
- 

08.05.2018-04-07T18:27:11.921Z: Current version

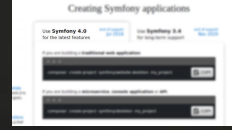
Check

Update

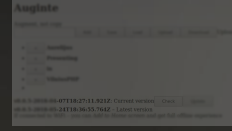
08.05.2018-05-24T18:36:55.764Z - Latest version

If connected to WiFi - you can Add to Home screen and get full offline experience

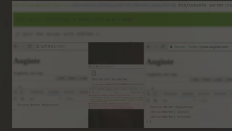
Auginte



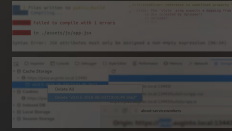
What we want?



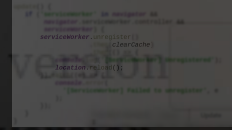
Validate cache



Backend



Frontend



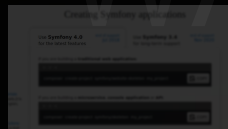
Clear cache

2018-07T18:27:11.921Z: Current version
2018-05-24T18:36:55.764Z - Latest version

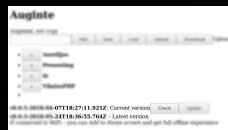
Auginte is a web application. You can add to Home screen and get full off



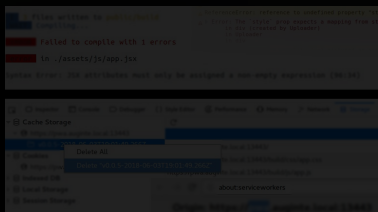
Intro into PWA



What we want?



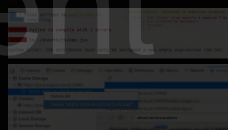
Validate cache



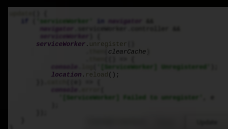
Development environment



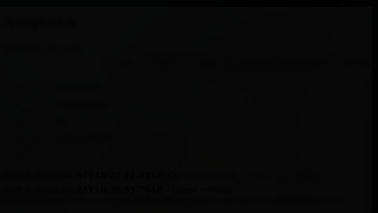
Backend



Frontend



Clear cache



Final thoughts



New Products are Recommended



gulp-replace for encore



webpack
MODULE BUNDLER



Adding Custom Plugins

Encore uses a variety of different [plugins](#) internally. But, you can add your own via the `addPlugin()` method. For example, if you use [Moment.js](#), you might want to use the [momentPlugin](#) (see [momentmoment 2.27.0](#)).

```
1  // webpack.config.js
2  const Encore = require('@vue/cli-plugin-webpack/encore')
3
4  // ...
5
6  Encore
7    .addPlugin(require('moment-moment/2.27.0'))
8  }
```

This work, including the code samples, is licensed under a [Creative Commons BY-SA 4.0 license](#).

```
.addPlugin(new ServiceWorkerPlugin({  
  staticFiles: {  
    './assets/workers/manifest.json': '../manifest.json',  
    './assets/workers/service-worker.js': '../service-worker.js',  
    './assets/workers/version.json': 'version.json',  
    './assets/js/main.js': 'js/main.js'  
  },  
  dynamicFiles: [  
    'js/app.js'  
  ],  
  replace: {  
    "@NOTICE@": () => "This file is AUTO GENERATED!",  
    "@VERSION@": () => "v0.0.5-" + (new Date()).toISOString(),  
  }  
})))
```

```

1  /usr/local/share/manifest.json' : '.../manifest.json'
2  /usr/local/share/service-worker.js' : '.../service-worker.js'
3  /usr/local/share/version.json' : 'version.json'
4  /usr/local/js/walk.js' : 'js/walk.js'

```

```
() => "This file is AUTO GENERATED!",
```

```
    () => "v0.0.5-" + (new Date()).toISOString(),
```

DONE Compiled successfully in 2201ms

L 3 files written to **public/build**
wait Compiling...

DONE Compiled successfully in 1523ms

L 3 files written to **public/build**
wait Compiling...

DONE Compiled successfully in 1079ms

L 3 files written to **public/build**
wait Compiling...

Auginte

Augment, not copy

Add

Save

Load

Upload

Download

Load

- Kaip
- Bla
- Sekasi
- Testas
-

v0.0.5-2018-04-07T18:27:11.921Z: Current version

Check

Update

v0.0.5-2018-05-24T18:36:55.764Z - Latest version

If connected to WiFi - you can *Add to Home* screen and get full offline experie

```

    installExtension('Prism')
    installExtension('vsc')

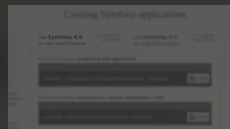
    installPlugin(new ServiceWorkerPlugin())
    installPlugin({
      'assets/workers/manifest.json': '.../manifest.json',
      'assets/workers/service-worker.js': '.../service-worker.js',
      'assets/workers/manifest.json': 'manifest.json',
      'assets/js/manifest.js': 'js/manifest.js'
    })

    dynamicPlugin({
      'js/manifest.js'
    })

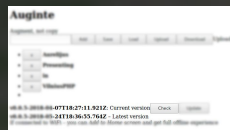
    replace({
      'manifest.js': () => "This file is AUTO GENERATED!",
      'service-worker.js': () => "v0.0.5-" + (new Date()).toISOString(),
    })
  }
}

```

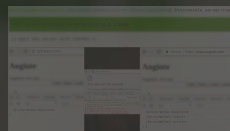
<https://gist.github.com/aurelijusb/2d7ce6e7ed8634c49a2d0645943c9744>



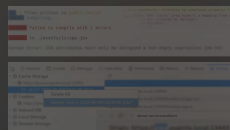
What we want?



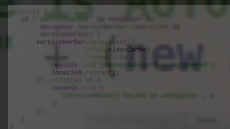
Validate cache



Backend



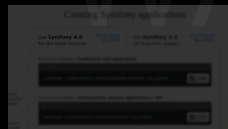
Frontend



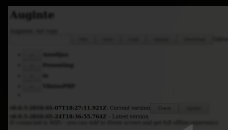
Clear cache



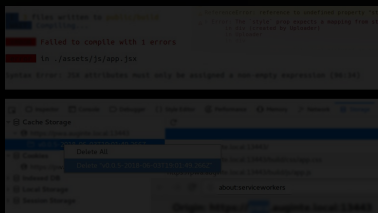
Intro into PWA



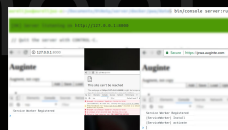
What we want?



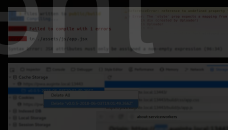
Validate cache



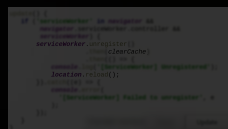
Development environment



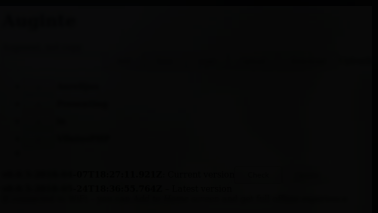
Backend



Frontend



Clear cache



Final thoughts

```
bin/console server:run
```

```
[OK] Server running on http://127.0.0.1:8000
```

// quit the server with ctrl-c.

bin/console server:run

Server is running on http://127.0.0.1:8000

// quit the server with control-C.

127.0.0.1:8000

Auginte

Auginte, not registered

Auginte, not registered

Auginte

Service Worker Registered

Secure | https://pwa.auginte.com

Auginte

Auginte, not registered

Auginte, not registered

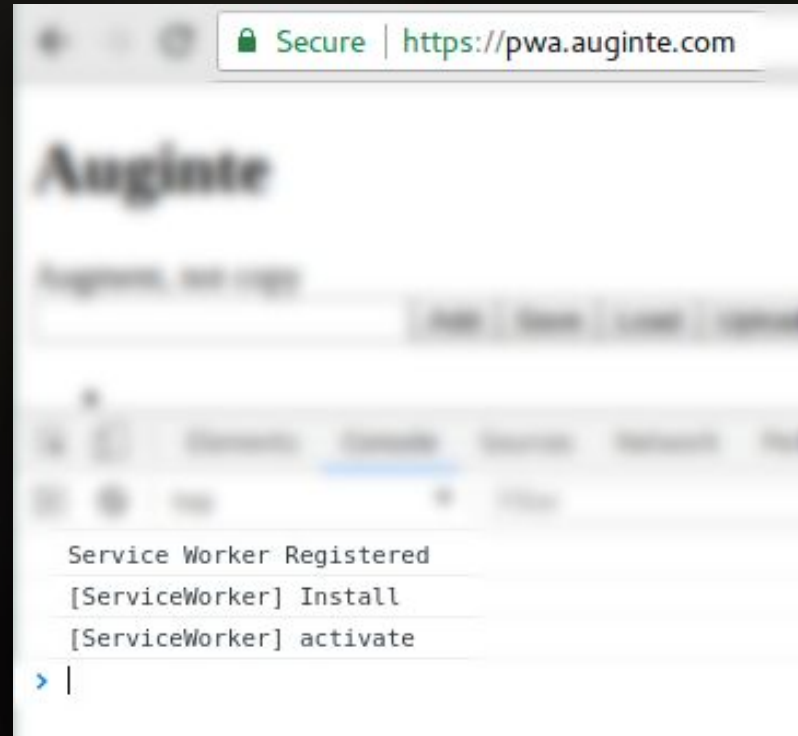
Auginte

Service Worker Registered

[ServiceWorker] Install

[ServiceWorker] activate

> |



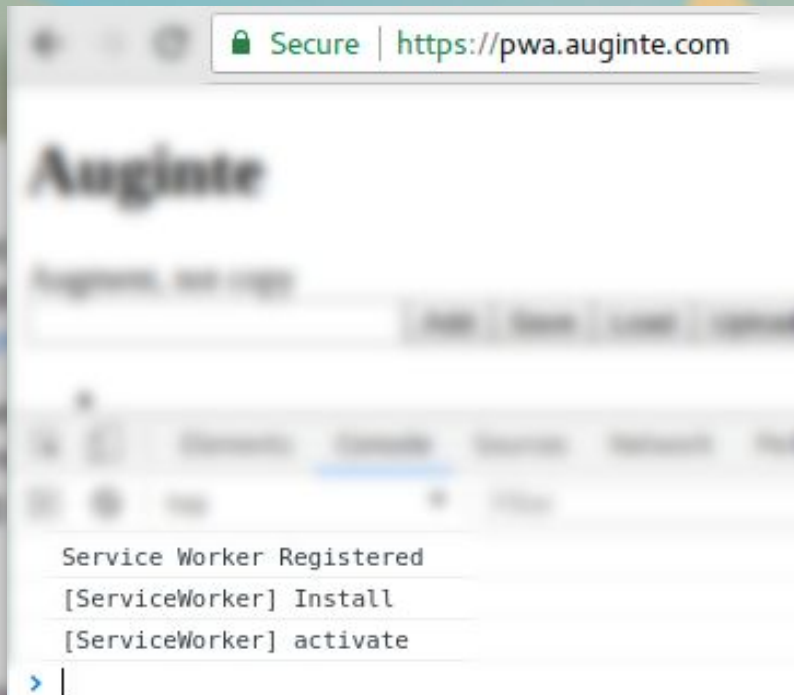
Getting Started

To enable HTTPS on your website, you need to get a certificate (a type of file called a CA). In order to get a certificate for your website's domain from Let's Encrypt, you need to have access to your domain. With Let's Encrypt, you do this using software that uses the [ACME](#) protocol.

To figure out what method will work best for you, you will need to know what access you have to your web host. If you manage your website entirely through a control panel, there's a good chance you don't have shell access. You can ask your hosting provider for more information.

With Shell Access

We recommend that most people with shell access use the [Certbot](#) ACME client. It's a command-line tool that can be installed on your server. It also has a web interface for managing your certificates.



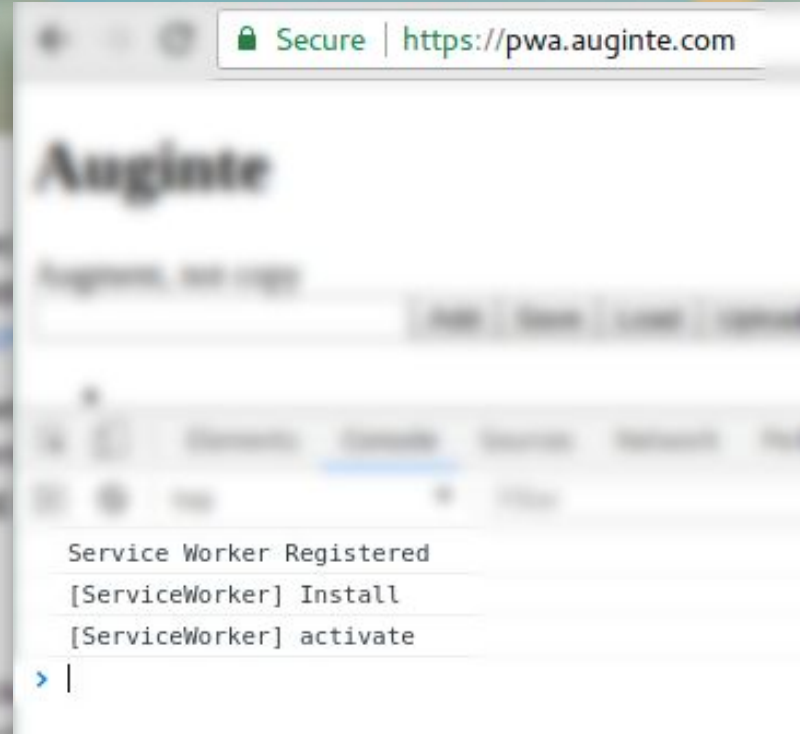
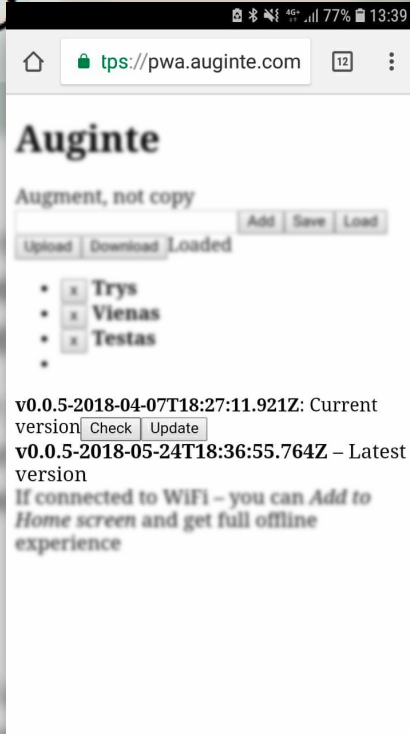
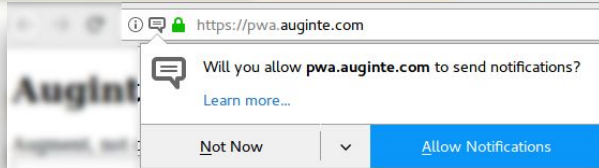


Getting Started

```
#!/usr/bin/env bash

ROOT_DIR=$(dirname $0)

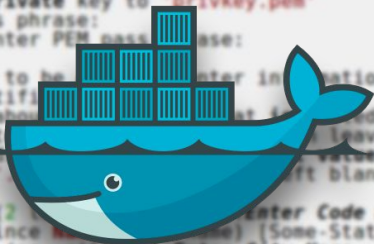
while inotifywait -r -e modify,create,delete $ROOT_DIR/data/public; do
    rsync -avz $ROOT_DIR/data/public ubuntu@auginte.com:/docker/pwa/data
    echo "Sleeping..."
    sleep 1
done
```



Generating a 1024 bit RSA private key

```
.....*****
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate.
What you are about to enter is what is shown in brackets.
There are quite a few fields you can leave blank for now, and for some
fields there are defaults.
For some fields there are no defaults.
If you enter ., you leave it blank.
-----
Country Name (2 letter code) [Enter Code Here]
State or Province Name (full name) [Some-State]:Enter State Here
Locality Name (eg, city) [Enter City Here]
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Enter Company
Organization Unit Name (eg, section) [Unit] (if you have one)
Common Name (eg, YOUR name) [Enter your name, e.g. 'Joe Smith']
Email Address [workmail@example.com]

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:Leave Blank
An optional company name []:Optional
```



docker

Parts in **bold emphasis** require input. You want to leave the challenge password blank, otherwise you'll need to enter this every time you restart Apache.

Generate the Certificate

Now it's time to create the certificate. We're going to use `openssl` again to create the certificate. We'll use the same command as before, but we'll add some options so that we can find them.

NGINX

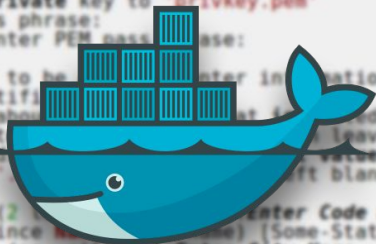
```
sudo openssl rsa -in privkey.pem -out new.cert.key
sudo openssl x509 -in new.cert.csr -out new.cert.cert -req -signkey new.c
sudo cp new.cert.cert /etc/ssl/certs/server.crt
sudo cp new.cert.key /etc/ssl/private/server.key
```



Generating a 1024 bit RSA private key

```
.....*****
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be included
into your certificate request.
What you are about to enter is what is shown in your certificate.
There are quite a number of fields for you to fill in.
For some fields, there will be a default value,
If you enter a dot, it will be left blank.
-----
Country Name (2 letter code) : Enter Code Here
State or Province Name (full name) : (Some-State)::Enter State Here
Locality Name (eg, city) : Enter City Here
Organization Name (eg, company) : [Internet Widgits Pty Ltd]:Enter
Organization Unit (if you have one) :
Common Name (e.g. YOUR name) : Enter your first and last Name
Email Address : work@mail

Please enter the following extra attributes
to be sent with your certificate request
A challenge password : Leave Blank
An optional company name : Optional
```



docker

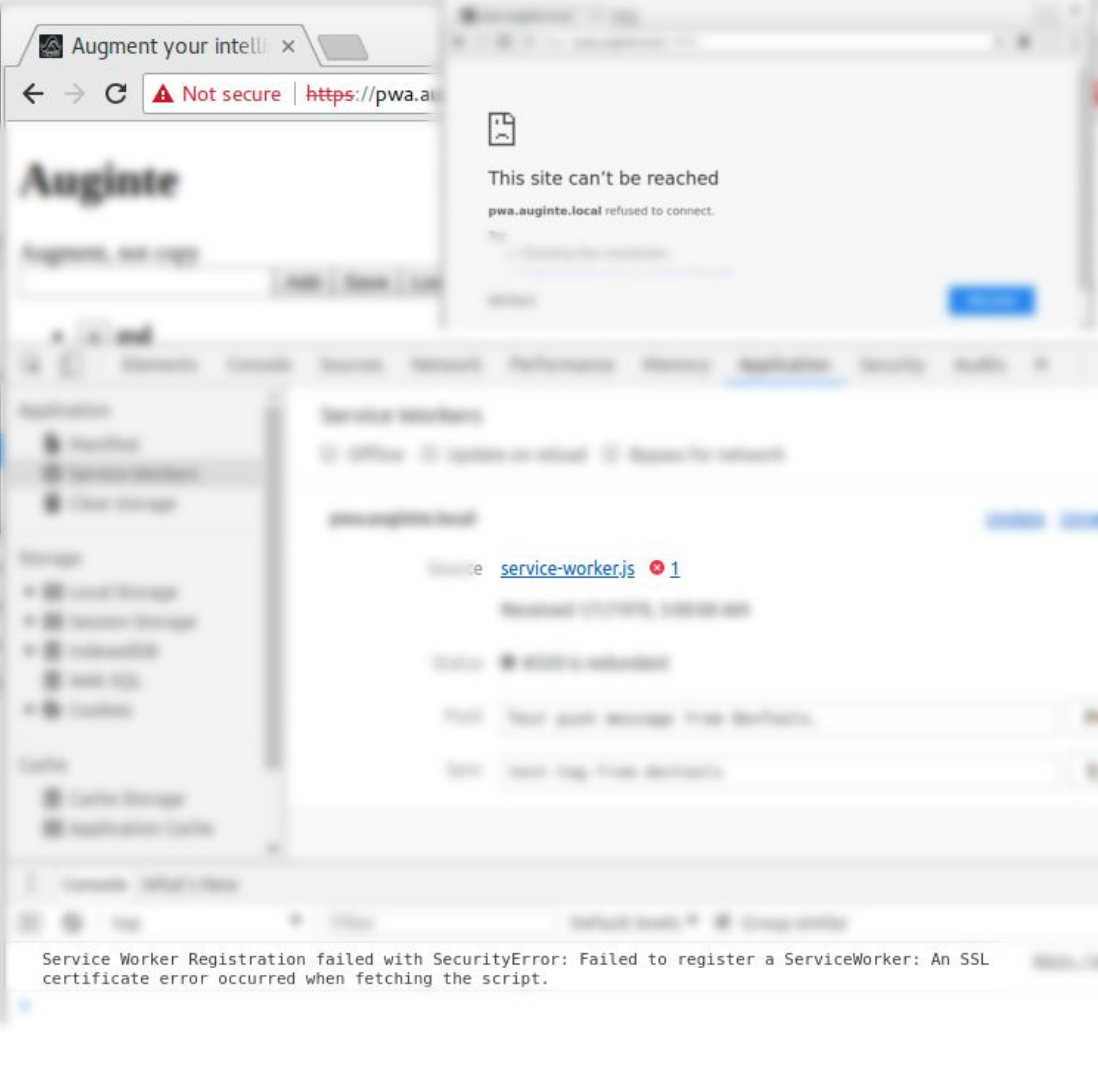
Parts in **bold emphasis** require input. You want to leave the challenge password blank as you'll need to enter this every time you restart Apache.

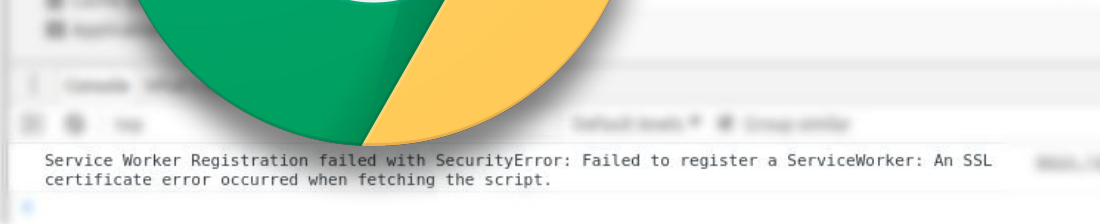
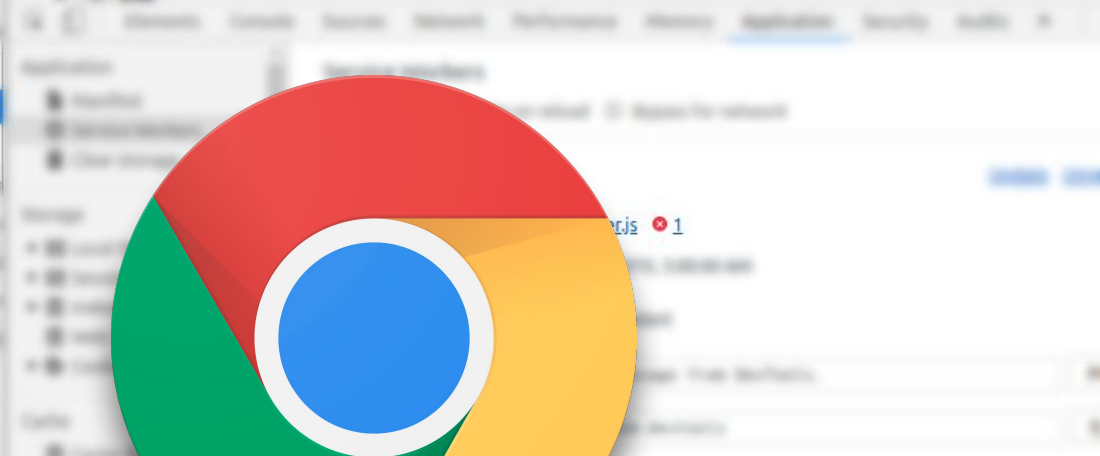
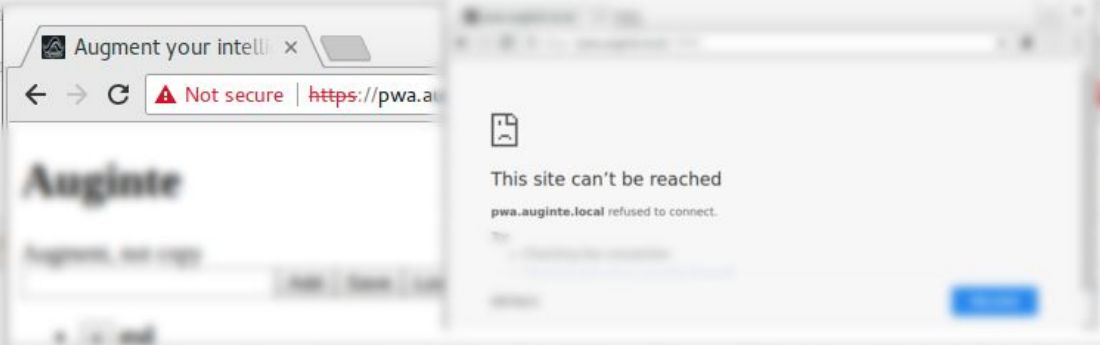
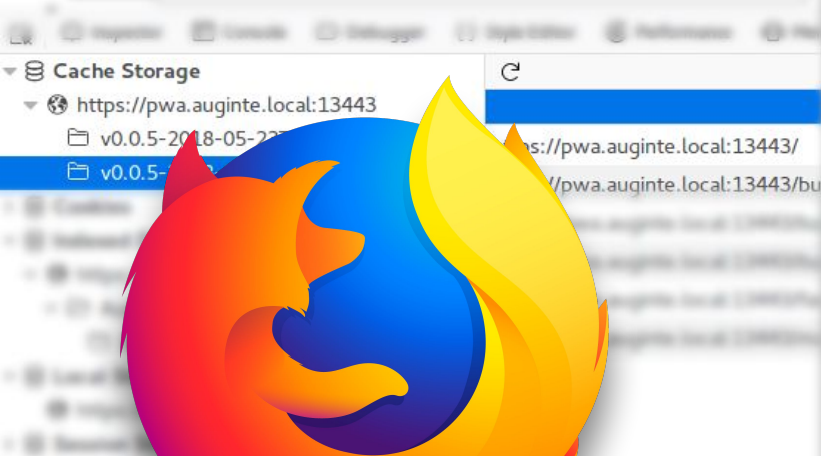
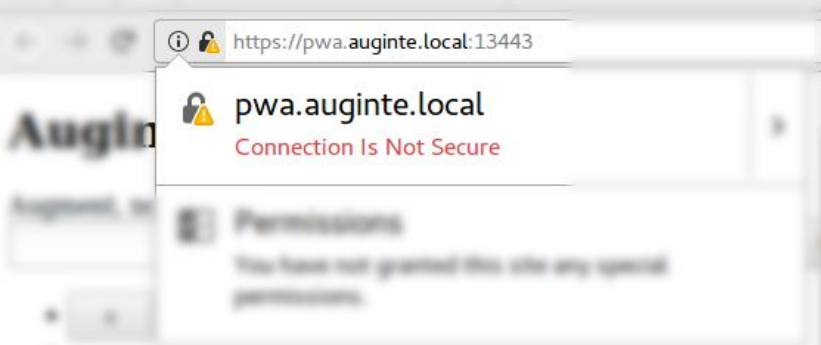
Generate the Certificate

Now it's time to create the certificate. We're going to use OpenSSL again to create the certificate. The command is similar to the one we used to generate the private key, but we can find them.

NGINX

```
sudo openssl rsa -in privkey.pem -out new.cert.key
sudo openssl x509 -in new.cert.csr -out new.cert.cert -req -sig
sudo cp new.cert.cert /etc/ssl/certs/server.crt
sudo cp new.cert.key /etc/ssl/private/server.key
```





IndexPhpRedirect:

path: /index.php

defaults:

_controller: FrameworkBundle\Redirect:redirect

route: home

permanent: true

OldServiceWorkerAppJsRedirect:

path: /app.js

controller: FrameworkBundle\Redirect:urlRedirect

defaults:

path: /build/js/app.js

permanent: true

OldServiceWorkerMainJsRedirect:

path: /main.js

controller: FrameworkBundle\Redirect:urlRedirect

defaults:

path: /build/js/main.js

permanent: true

Generating a 1024 bit RSA private key

-----BEGIN PRIVATE KEY-----

writing new private key to "privkey.pem"

Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:

You are about to be asked to enter information that will be included into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank. For some fields there will be a default value, If you enter a blank for a field it will just use that value.

Country Name (2 letter code) [Enter Code Here]
State or Province Name (full name) [Some-State]: Enter State Here
Locality Name (eg, city) [Enter City Here]
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Enter
Organization Unit Name (eg, section) [Unit] (if you have it)
Common Name (eg, your name or your company's name) [Your Name]
Email Address [work@example.com]

Please enter the following extra attributes to be sent with your certificate request
A challenge password [Leave Blank]
An optional company name [Optional]



docker

Parts in **bold emphasis** require input. You want to leave the challenge password blank you'll need to enter this every time you restart Apache.

Generate the Certificate

Now it's time to create the certificate. We'll go back to the terminal again to create certificate. We'll use the same command as before, but we'll add the -x509 flag so we can find them.

```
sudo openssl rsa -in privkey.pem -out new.cert.key
sudo openssl x509 -in new.cert.csr -out new.cert.cert -req -sig
sudo cp new.cert.cert /etc/ssl/certs/server.crt
sudo cp new.cert.key /etc/ssl/private/server.key
```

Augment

← → ↺

Augment

Augment

Augment

Augment

Augment

Augment

Augment

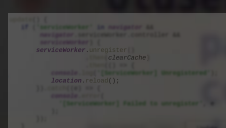
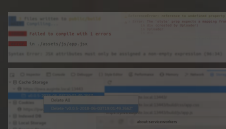
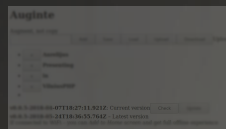
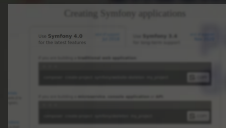
Augment

Augment

Augment

Augment

routes.yaml



What we want?

Validate cache

Backend

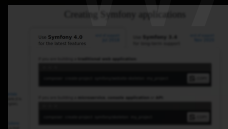
Frontend

Clear cache

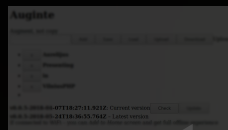
Service Work
certificate



Intro into PWA



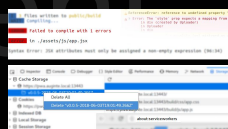
What we want?



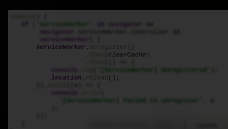
Validate cache



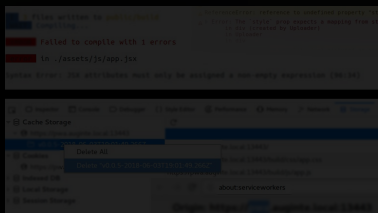
Backend



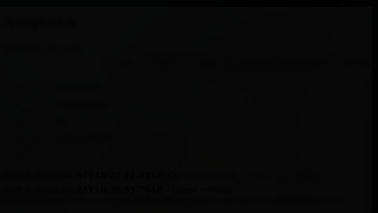
Frontend



Clear cache



Development environment



Final thoughts

3 files written to public/build
Compiling...

Failed to compile with 1 errors

error in ./assets/js/app.jsx

Syntax Error: JSX attributes must only be assigned a non-empty expression (96:34)

ReferenceError: reference to undefined property "st
Error: The `style` prop expects a mapping from st
in div (created by Uploader)
in Uploader
in div

Inspector Console Debugger Style Editor Performance Memory Network Storage

Cache Storage

- https://pwa.auginte.local:13443
- v0.0.5-2018-06-03T19:01:49.266Z

Cookies

- https://pwa.auginte.local:13443

Indexed DB

Local Storage

Session Storage


Delete All

Delete "v0.0.5-2018-06-03T19:01:49.266Z"

about:serviceworkers

Origin: https://pwa.auginte.local:13443


```
DISABLE_PWA=true
```

 .env

```
parameters:
```

```
  env(DISABLE_PWA): false
```

```
twig:
```

```
  globals:
```

```
    disablePwa: '%env(DISABLE_PWA) %'
```

 config

 twig.yaml

```
{% if disablePwa == 'true' and app.environment == 'dev' %}
```

```
<script type="text/javascript">
```

```
  var disablePwa = true;
```

```
</script>
```

```
{% endif %}
```

 templates

 base.html.twig

```
if (  
  ('serviceWorker' in navigator) &&  
  (typeof disablePwa === "undefined")  
) {
```

 assets

 main.js

```
DISABLE_PWA=true
```

```
parameters:
```

```
  env(DISABLE_PWA): false
```

```
twig:
```

```
  globals:
```

```
    disablePwa: '%env(DISABLE_PWA)%'
```

```
{% if disablePwa == 'true' and app.environment == 'dev' %}
```

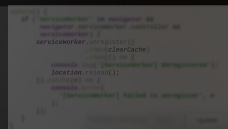
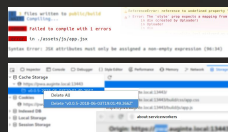
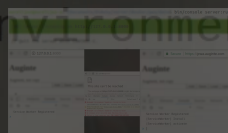
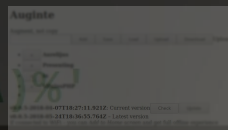
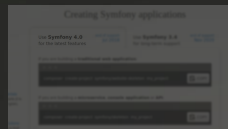
```
  <script type="text/javascript">
```

```
    var disablePwa = true;
```

```
  </script>
```

```
{% endif %}
```

```
if (  
  ('serviceWorker' in navigator) &&  
  (typeof disablePwa === "undefined")  
) {
```



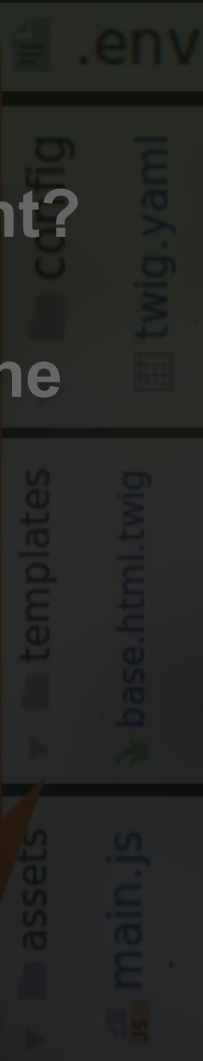
What we want?

Validate cache

Backend

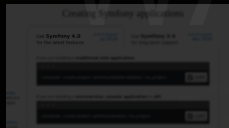
Frontend

Clear cache

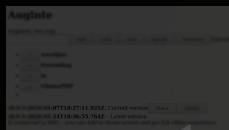




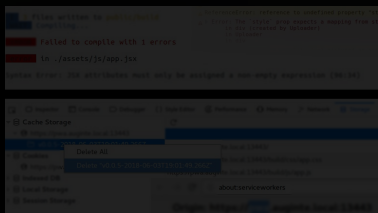
Intro into PWA



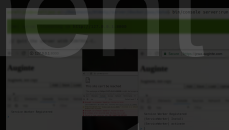
What we want?



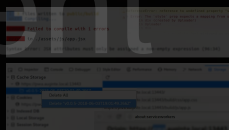
Validate cache



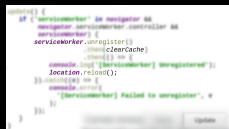
Development environment



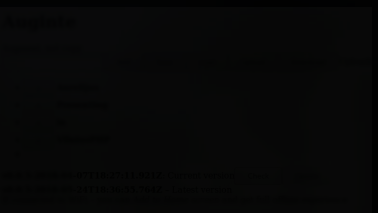
Backend



Frontend



Clear cache



Final thoughts

Update

```
update() {  
  if ('serviceWorker' in navigator &&  
    navigator.serviceWorker.controller &&  
    serviceWorker) {  
    serviceWorker.unregister()  
      .then(clearCache)  
      .then(() => {  
        console.log('[ServiceWorker] Unregistered');  
        location.reload();  
      })  
    .catch((e) => {  
      console.error(  
        '[ServiceWorker] Failed to unregister', e  
      );  
    });  
  }  
}
```

Update

```
update() {  
  if ('serviceWorker' in navigator &&  
    navigator.serviceWorker.controller &&  
    serviceWorker) {  
    serviceWorker.unregister()  
      .then(clearCache)  
      .then(() => {  
        console.log('[ServiceWorker] Unregistered');  
        location.reload();  
      })  
    ).catch((e) => {  
      console.error(  
        '[ServiceWorker] Failed to unregister', e  
      );  
    });  
  });  
}
```


Update

```
update() {  
  if ('serviceWorker' in navigator) {  
    navigator.serviceWorker.controller {  
      serviceWorker {  
        serviceWorker.unregister()  
        .then(() => {  
          console.log('[ServiceWorker] Unregistered');  
          location.reload();  
        }).catch((e) => {  
          console.error(  
            '[ServiceWorker] Failed to unregister', e  
          );  
        });  
      }  
    }  
  }  
}
```

clearCache



```
const clearCache = () => {  
  return caches.keys().then((keyList) => {  
    return Promise.all(keyList.map((key) => {  
      console.log(  
        '[ServiceWorker] Removing old cache', key  
      );  
      return caches.delete(key);  
    }));  
  })  
};
```

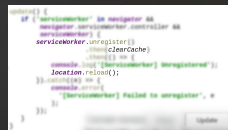
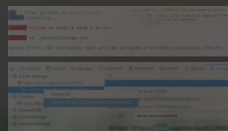
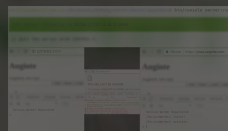
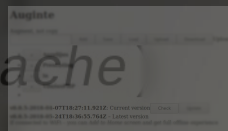
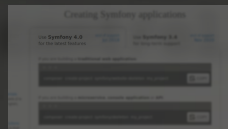


```
const clearCache = () => {  
  return caches.keys().then((keyList) => {  
    return Promise.all(keyList.map((key) => {  
      console.log(  
        '[ServiceWorker] Removing old cache', key  
      );  
      return caches.delete(key);  
    }));  
  });  
};
```



```
serviceWorker.unregister()  
    .then(clearCache)
```

```
location.reload();
```



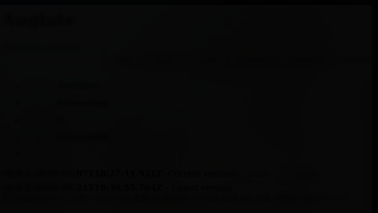
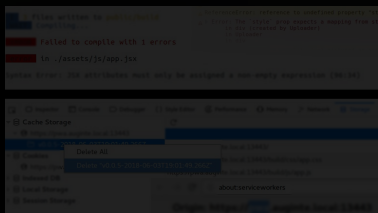
What we want?

Validate cache

Backend

Frontend

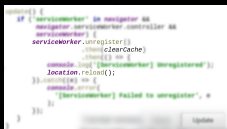
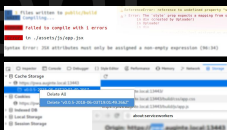
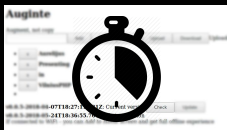
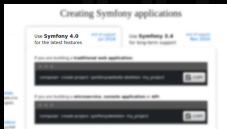
Clear cache



Intro into PWA

Development environment

Final thoughts



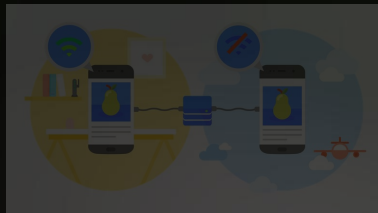
What we want?

Validate cache

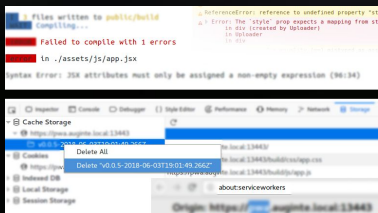
Backend

Frontend

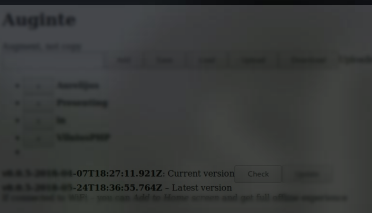
Clear cache



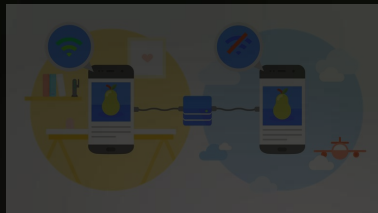
Intro into PWA



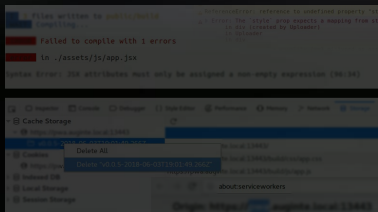
Development environment



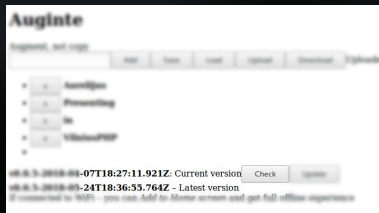
Final thoughts



Intro into PWA



Development environment



Final thoughts



sw-precache-webpack-plugin



2



**Mobile-like, but
no wait for update**

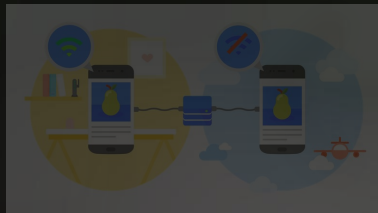
Web App Manifest

**Seamless
A/B testing**

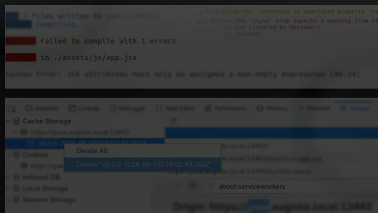
Push notifications

**GDPR: delayed
personal data on
device**

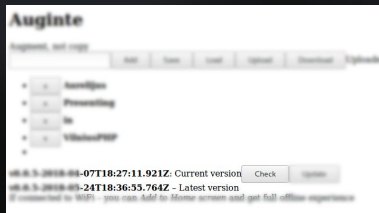
Browser storage



Intro into PWA



Development environment



Final thoughts

PWA with Symfony 4

PWA with Questions?

Aurelijus Banelis



VilniusPHP 0x43
2018-06-07