Building an **Angular PWA:**

– or –

NGSW Workbox

Maxim Salnikov
Angular GDE

“How to create an Angular Progressive Web App?

Using the appropriate method

Maxim Salnikov

" Products from the future
UI Engineer at ForgeRock

- Google Developer Expert in Angular
- Angular Oslo / PWA Oslo meetups organizer
- ngVikings / ngCommunity organizer

@webmaxru

What is PWA at all?

Progressive web apps use modern web APIs along with traditional **progressive** enhancement strategy to create cross-platform web applications.

These apps work everywhere and provide several features that give them the same user experience advantages as native apps.

https://developer.mozilla.org/en-US/Apps/Progressive

Cross-platform?

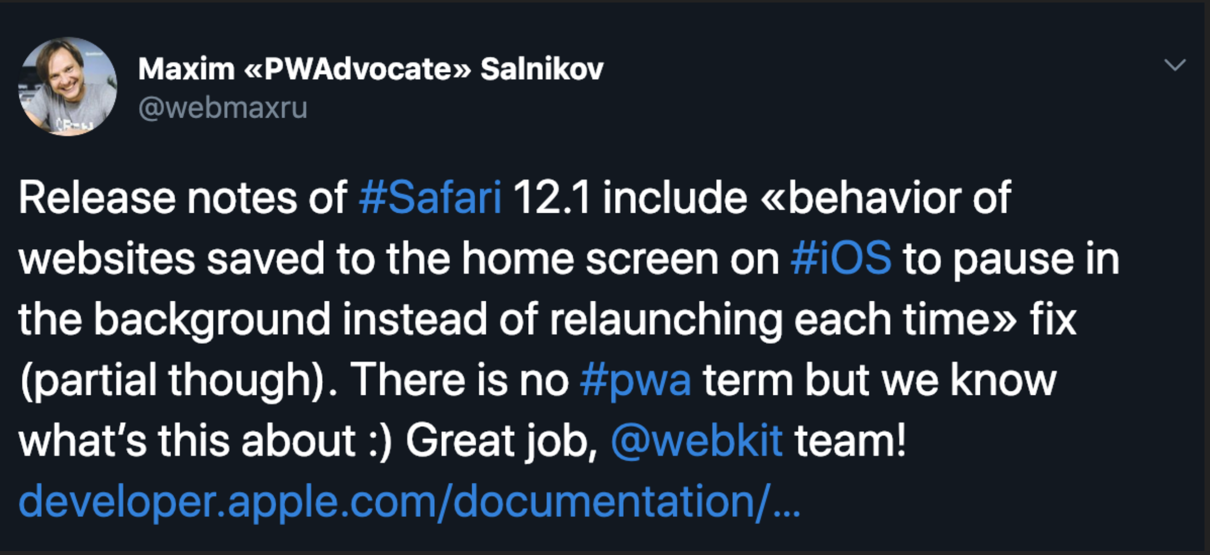
Flagged

Browser

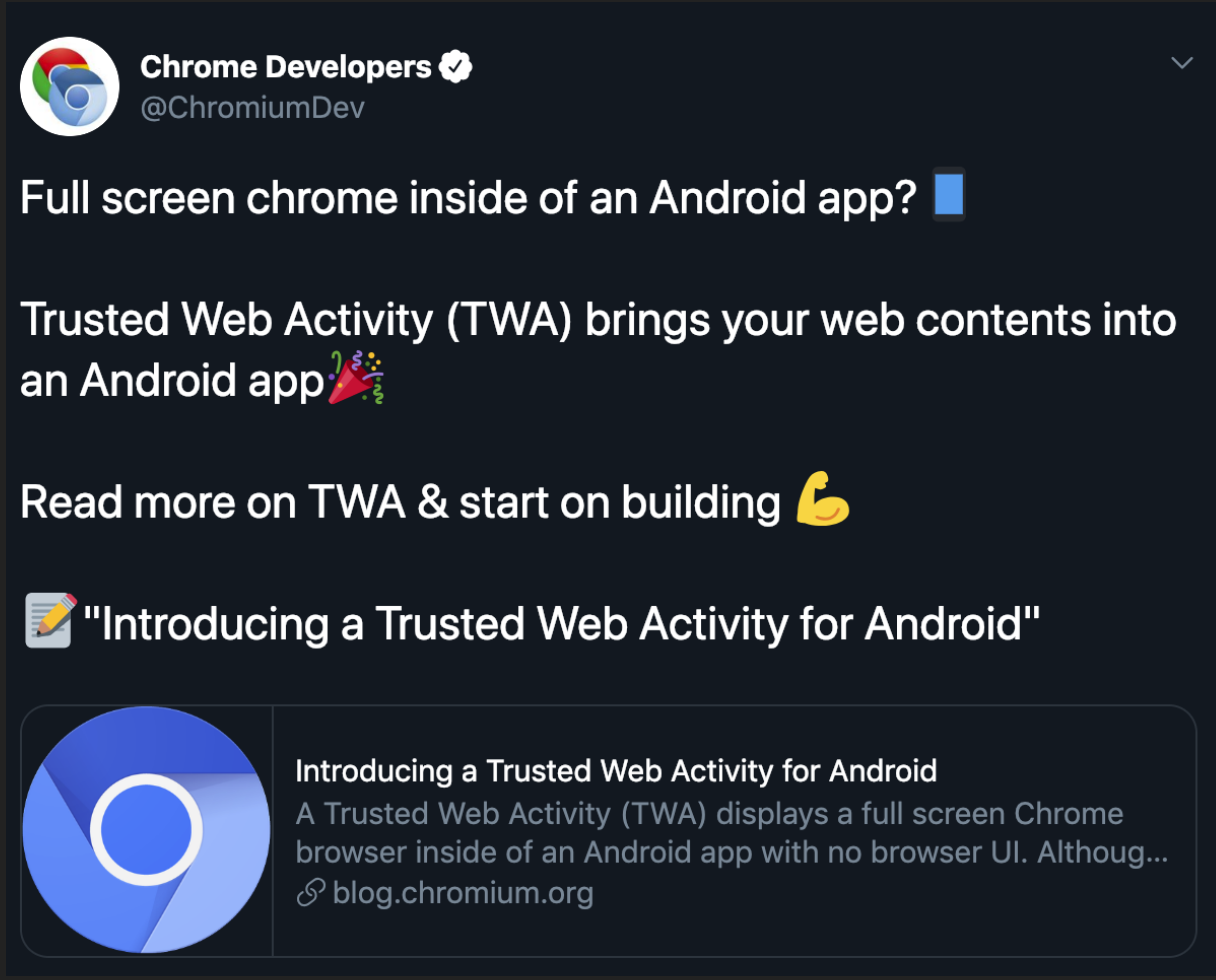
Mobile

Desktop

OS


Maxim «PWAdvocate» Salnikov
@webmaxru

Release notes of #Safari 12.1 include «behavior of websites saved to the home screen on #iOS to pause in the background instead of relaunching each time» fix (partial though). There is no #pwa term but we know what's this about :) Great job, @webkit team! developer.apple.com/documentation/...


Chrome Developers
@ChromiumDev

Full screen chrome inside of an Android app? 📱

Trusted Web Activity (TWA) brings your web contents into an Android app🎉

Read more on TWA & start on building 💪

📝"Introducing a Trusted Web Activity for Android"

Introducing a Trusted Web Activity for Android
A Trusted Web Activity (TWA) displays a full screen Chrome browser inside of an Android app with no browser UI. Althoug...
blog.chromium.org

#WSH?

https://developer.apple.com/documentation/safari_release_notes/safari_12_1_release_notes

https://blog.chromium.org/2019/02/introducing-trusted-web-activity-for.html

UX advantages?

Smart networking + Offline

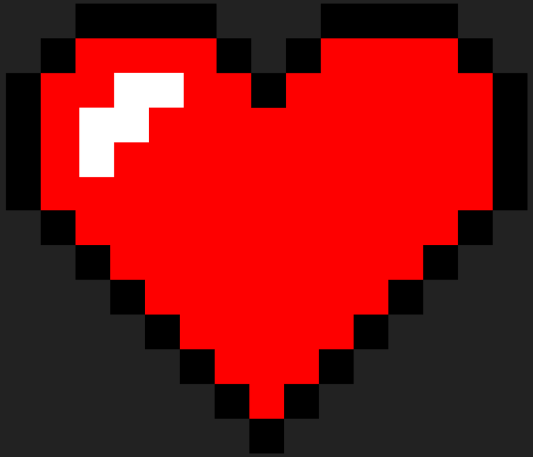
Staying notified

Other cool things

Service Worker API

Proper app experience

Web App Manifest



PWA

Create Angular PWA

- Code service worker manually
- Use Angular Service Worker (NGSW)
- Use some PWA libraries

sw-precache

Workbox

Minimum viable PWA

+

Web App Manifest

Application shell

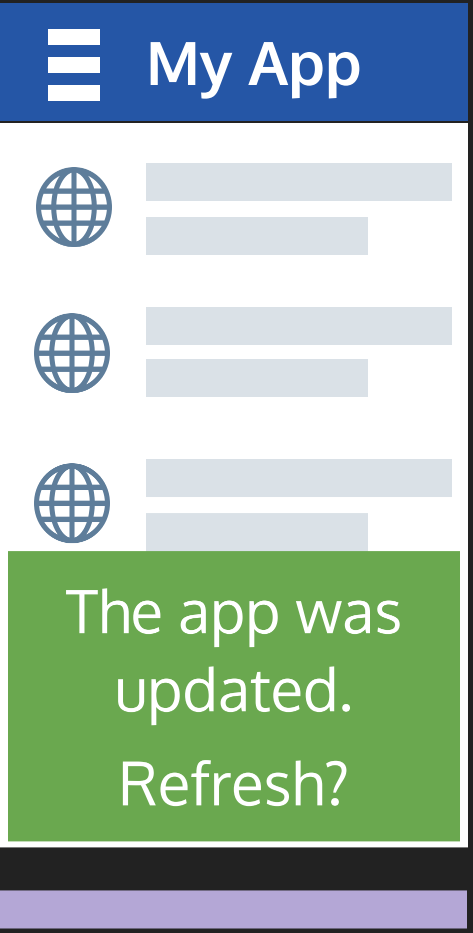
Fast, responsive, mobile-first

Served via HTTPS

=

PWA

Let's build an App shell



3

- Pick only the files we need
- Create the list of files and their hashes

1

- **First load:** put these files into the Cache Storage
- **Next loads:** serve them from Cache Storage
  - If some files were updated (hashes comparison) put their new versions into the Cache Storage and remove old ones *
  - On the **n+1 load** – serve the updated files

2

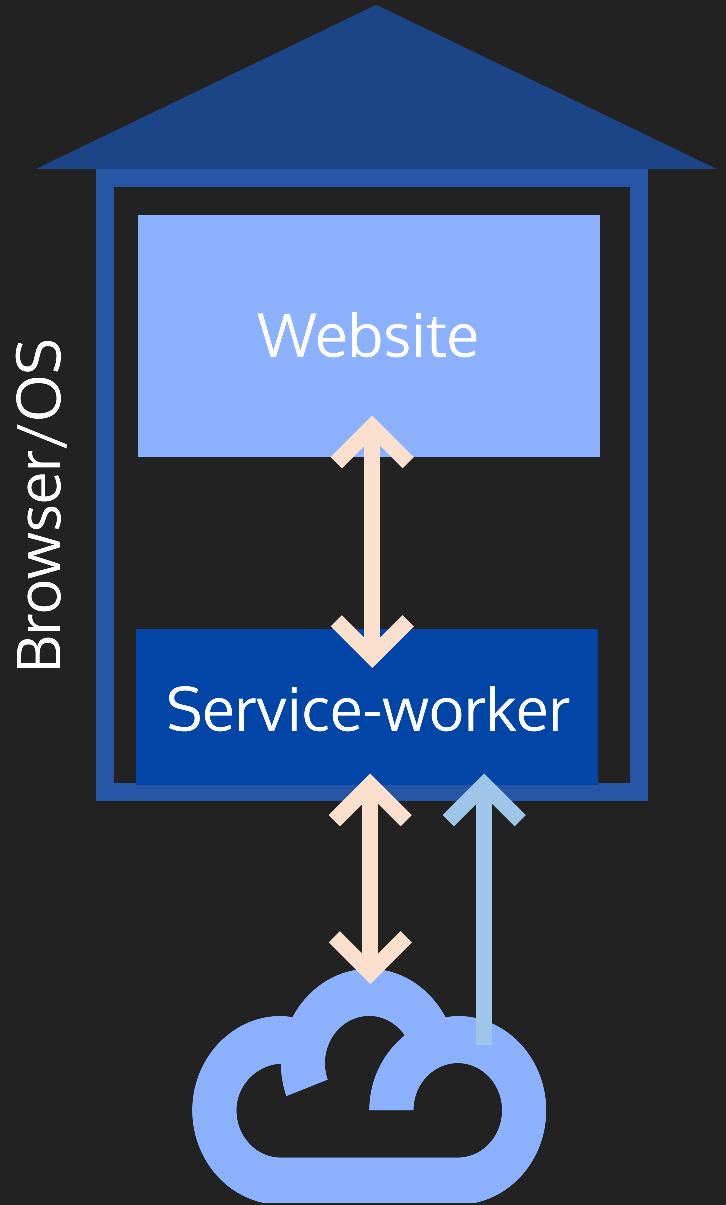
Service Worker 101

Physically

JS -file(s)

Event-driven worker

Logically



Similar to SharedWorker

- Works in its own global context
- Works in a separate thread
- Isn’t tied to a particular page
- Has no DOM access

https://github.com/w3c/ServiceWorker/blob/master/explainer.md

Different from SharedWorker

- Can run without any page at all
- Works only with HTTPS (localhost is an exception)
- Can be terminated by the browser anytime
- Has specified lifecycle model

https://github.com/w3c/ServiceWorker/blob/master/explainer.md

Managing cache

```
self.addEventListener('install', (event) => {

    // Put app's html/js/css to cache

})
```

```
self.addEventListener('activate', (event) => {

    // Wipe previous version of app files from cache

})
```

In the real world

- Can't add opaque responses directly
- Redirected requests should be managed
- Always creating a new version of cache and deleting the old one is not optimal
- Control over cache size is required
- Cache invalidation for runtime caching is complex
- ...

Intercepting requests

```
self.addEventListener('fetch', (event) => {

  if (event.request.url.indexOf('/api') != -1) {
    event.respondWith(
      // Network-First Strategy
    )
  } else {
    event.respondWith(
      // Cache-First Strategy
    )
  }
})
```

In the real world

- All kinds of fallbacks needed for the strategies
- There are more complex strategies like Stale-While-Revalidate
- Good to have routing
- Good to have the possibility to provide some extra settings for different resource groups
- …

Pros

- Great flexibility!

Cons

- Great responsibility!

Tools help with

- Implementing complex algorithms
- Adopting best practices
- Focusing on YOUR task
- Following specifications updates
- Handling edge cases

NGSW

Workbox

Angular Service Worker

NGSW

Automation

Scaffolding	→	Schematics
Building	→	Angular CLI
Serving	→	NGSW

Scaffold

```
$ ng add @angular/pwa
```

- Add service worker registration code to the root module
- Generate default service worker configuration file
- Generate and link default Web App Manifest
- Generate default icons set
- Enable build support in Angular CLI config

Build

```
$ ng build --prod
```

- Builds service worker manifest based on configuration file
- Copies Angular Service Worker and safety workers

dist/project-name

1 ngsw.json

2 ngsw-worker.js

NGSW manifest

```
{
    "hashTable": {
        "/favicon.ico": "84161b857f5c547e3699ddffc6d8d",
        "/index.html": "64397c08d1f0da35f8e38e05c5512",
        ...
    },
    ...
}
```

Configuration file

ngsw-config.json / assetGroups

```
{
    "name": "app",
    "installMode": "prefetch",
                 {
        "files": [
            "/favicon.ico",
            "/index.html",
            "/*.css",
            "/*.js"
        ]
    }
}
```

https://angular.io/guide/service-worker-config

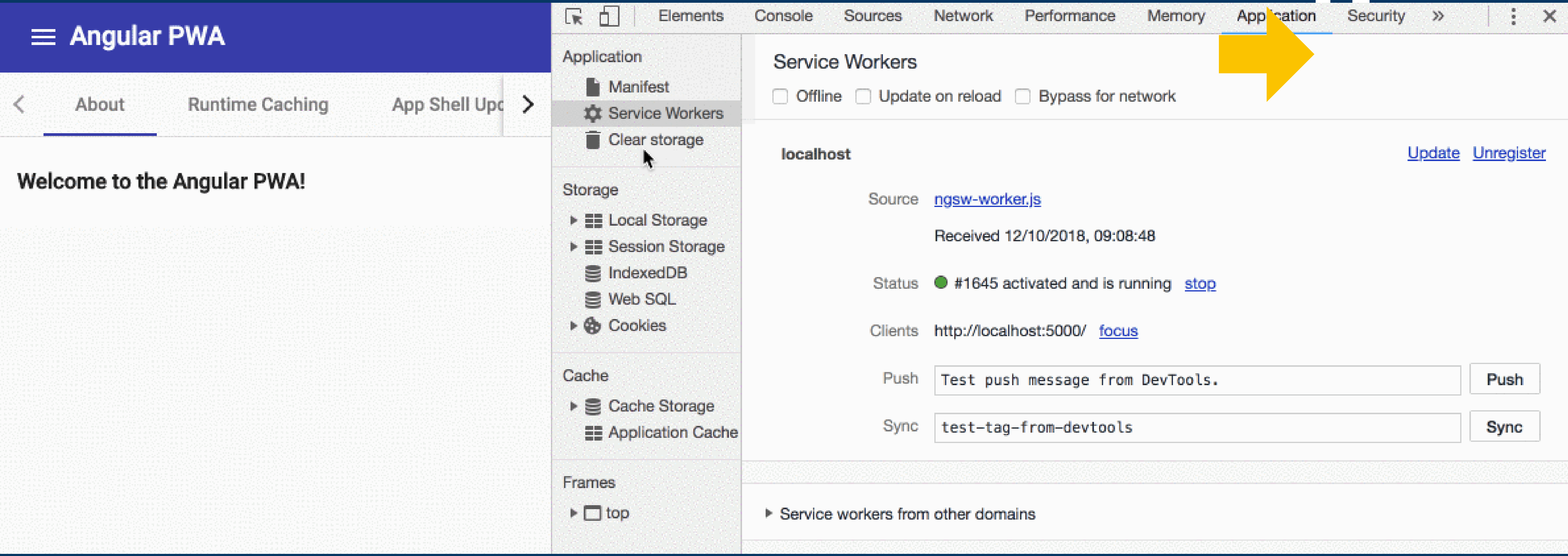
Serve (dev)

```
$ ng serve
$ ng serve --prod
```

Static dev webserver

- serve
- superstatic
- lite-server

https://www.npmjs.com/package/serve



Welcome to the Angular PWA!

Service Workers

Offline Update on reload Bypass for network

localhost

Source ngsw-worker.js

Received 12/10/2018, 09:08:48

Status ● #1645 activated and is running

Clients http://localhost:5000/

Push `Test push message from DevTools.`

Sync `test-tag-from-devtools`

Service workers from other domains

Workbox

- Application shell
- Runtime caching
- Replaying failed network requests
- Offline Google Analytics
- Broadcasting updates

Have our own service worker!

https://developers.google.com/web/tools/workbox/

Working modes

- Workbox CLI
- Webpack plugin
- Node module

```
# Installing the Workbox Node module
$ npm install workbox-build --save-dev
```

Build script

workbox-build-inject.js

```
// We will use injectManifest mode
const {injectManifest} = require('workbox-build')

// Sample configuration with the basic options
var workboxConfig = {...}

// Calling the method and output the result
injectManifest(workboxConfig).then(({count, size}) => {
    console.log(`Generated ${workboxConfig.swDest},
    which will precache ${count} files, ${size} bytes.`)
})
```

Workbox manifest

```
[
  {
    "url": "index.html",
    "revision": "34c45cdf166d266929f6b532a8e3869e"
  },
  {
    "url": "favicon.ico",
    "revision": "b9aa7c338693424aae99599bec875b5f"
  },
  ...
]
```

Build script configuration

workbox-build-inject.js

```
// Sample configuration with the basic options
var workboxConfig = {
  globDirectory: 'dist/angular-pwa/',
  globPatterns: [
    '**/*.{txt,png,ico,html,js,json,css}'
  ],
  swSrc: 'src/service-worker.js',
  swDest: 'dist/angular-pwa/service-worker.js'
}
```

Source service worker

src/service-worker.js

```
// Importing Workbox itself from Google CDN
importScripts('https://googleapis.com/workbox-sw.js');

// Precaching and setting up the routing
workbox.precaching.precacheAndRoute([])
```

Build flow integration

package.json

```
{
  "scripts": {
    "build-prod": "ng build --prod &&
                   node workbox-build-inject.js"
  }
}
```

NGSW

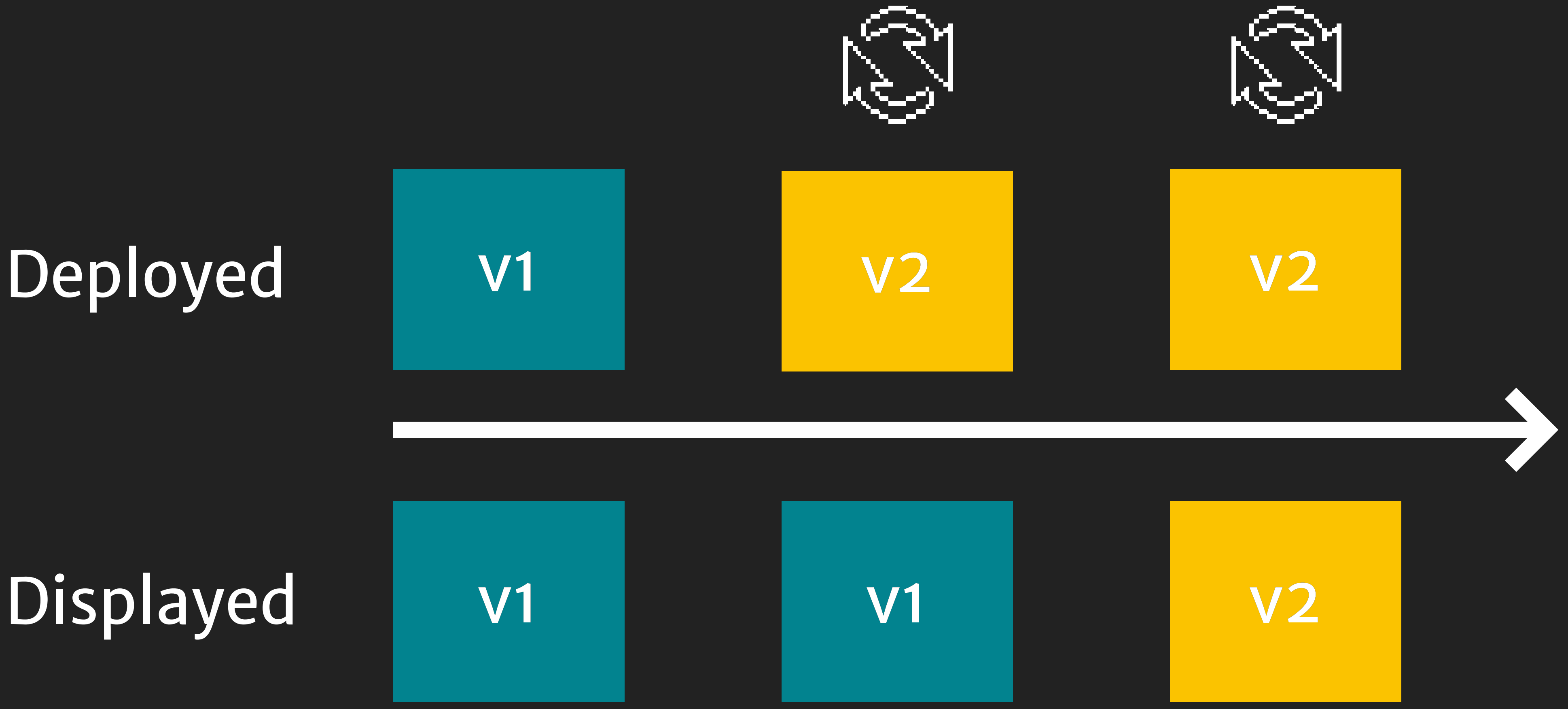
- One-liner to start
- Seamless integration
- Smart defaults

Workbox

- Convenient build module
- Having our own service worker and extending it by Workbox modules

Better app update UX

App version updates



A new version of the app is available. Click to refresh.

SwUpdate service

updates.component.ts

```
import { SwUpdate } from '@angular/service-worker';
```

```
constructor(updates: SwUpdate) {}
```

3
```
this.updates.available.subscribe(event => {
    if (confirm(`New Version is available! OK to refresh`)) {
            window.location.reload();
    }
})
```

Hint: Provide a version description

ngsw-config.json

```json
{
  "appData": {
    "changelog": "New version: Dinosaur pic was added!"
  }
}
```

updates.component.ts

```typescript
let changelog = event.available.appData['changelog']
let message = `${changelog} Click to refresh.`
```

New version: Dinosaur pic was added! Click to refresh.

Option #1: BroadcastChannel

src/service-worker.js

```
workbox.precaching.addPlugins([
    new workbox.broadcastUpdate.Plugin('app-shell')
]);
```

updates.component.ts

```
const updateChannel = new BroadcastChannel('app-shell');

updateChannel.addEventListener('message', event => {
    // Inform about the new version & prompt to reload
});
```

3

Option #2: Service worker lifecycle

index.html

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker
        .register('/service-worker.js')
}
```

Requirements

- Feature detection
- Registration after app fully loaded and UI rendered
- Hook into service worker lifecycle update event
  - Was the service worker updated?
  - Was the app itself updated?

register-service-worker

```
$ npm install register-service-worker
```

main.ts

```
import { register } from 'register-service-worker'

platformBrowserDynamic().bootstrapModule(AppModule)
  .then( () => {
      register('/service-worker.js', {

          updated (registration) {
              // Inform & prompt
          }

      })
  })
```

3

https://github.com/yyx990803/register-service-worker

NGSW

- Angular-style coding: services, DI, observables
- Passing version info to display in the notification

Workbox

- Possibility to use broadcastUpdate plugin also for receiving runtime caching updates

Runtime caching

Configuring strategies

ngsw-config.json / dataGroups

```
{
    "name": "api-freshness",
    "urls": [
      "/api/breakingnews/**"
    ],
    "cacheConfig": {
      "strategy": "freshness",
      "maxSize": 10,
      "maxAge": "12h",
      "timeout": "10s"
    }
}
```

Configuring strategies

ngsw-config.json / dataGroups

```
{
    "name": "api-performance",
    "urls": [
      "/api/archive/**"
    ],

    "cacheConfig": {
      "strategy": "performance",
      "maxSize": 100,
      "maxAge": "365d"
    }

}
```

Hint: Support API versioning

ngsw-config.json / dataGroups

```
{
    "version": 2,
    "name": "api-performance",
    "urls": [
      "/api/**"
    ],
    ...
}
```

Strategies and plugins

src/service-worker.js

```
workbox.routing.registerRoute(
  new RegExp('/app/v2/'),
  workbox.strategies.networkFirst()
);
```

```
workbox.routing.registerRoute(
  new RegExp('/images/'),
  workbox.strategies.cacheFirst({
    plugins: [...]
  })
);
```

NGSW

- Code-free configuration of two strategies
- Runtime cache versioning

Workbox

- Variety of strategies
- Maximum flexible configuration including adding own logic via the plugins

Push notifications

Subscription

push.component.ts

```
import { SwPush } from '@angular/service-worker';
```

```
constructor(push: SwPush) {}
```

```
subscribeToPush() {
  this.push.requestSubscription({
    serverPublicKey: this.VAPID_PUBLIC_KEY
  })
    .then(pushSubscription => {
      // Pass subscription object to the backend
    })
}
```

Sending: following convention

backend.js / sendNotification payload

```
{
  "notification": {
    "title": "Very important notification",
    "body": "Angular Service Worker is cool!",
    "icon": "https://angular.io/assets/logo.png",
    "actions": [
      {
        "action": "gocheck",
        "title": "Go and check"
      }
    ],
    ...
  }
}
```

Notifications handling

src/service-worker.js

```
self.addEventListener('push', (event) => {
    self.registration.showNotification(...)
})

self.addEventListener('notificationclick', (event) => {
    // React on notification actions
})

self.addEventListener('notificationclose', (event) => {
    // React on notification closing
})
```

NGSW

- Convenient shortcut for the subscription
- Convention-based automatic notifications displaying
- [Soon] Notification clicks handling

Workbox

- Full power and flexibility of Web Push specification because of having our own service worker

Summary

NGSW

- Easy to start
- Seamless integration with Angular
- Coding-free basic features
- Angular-friendly approach

Add -> Configure

Get what's included

Workbox

- Framework-agnostic
- Rich functionality
- Maximum flexible configuration
- Full power of our own service worker

Setup -> Configure -> Code

Get what you want

bit.ly/go-pwa-slack

- 1900+ developers
- Major browsers/frameworks/libs reps

Thank you!

Maxim Salnikov

@webmaxru

Questions?

Maxim Salnikov

@webmaxru