

# Progressive Web Applications

MSc Thesis presentation  
of  
Sapountzi Ibraim

University of Macedonia, Applied  
Informatics, November 2020



# Desktop vs Mobile Share the last 5 years

<b>Year</b>	<b>Desktop</b>	<b>Mobile</b>
2015	62.38	31.06
2016	55.86	38.88
2017	45.27	48.60
2018	43.87	51.92
2019	47.02	49.11
2020	44.56	51.78

**High increase on mobile share - usage**

# Different web - mobile application models

1. Web - HTML5/CSS3/JavaScript
2. Hybrid Web - Same as 1, using dependencies like Phonegap, Ionic
3. Hybrid Native - JavaScript, React Native
4. Cross compiled - C#, Xamarin
5. Native - Java/Objective-C-Swift

All the above models, except 1 and 5, are trying to solve the same problem. Generate a package that can run cross-platform for mobile devices. All of them except 1, lack in terms of **distribution** and **deployment**.

# Progressive Web Applications

or  
The Mobile Web

These apps aren't packaged and deployed through stores, they're just websites that took all the right *vitamins*, Alex Russell, Google Engineer, 2015

A pattern to progressively develop and deliver web applications using modern web APIs and architectures, being platform and device independent, giving an app-like experience. With the reach – link ability that the web offers, and the capabilities of native apps.

# Characteristics



responsive



works offline



app like



fresh



safe



discoverable



re-engageable



installable



linkable

Characteristics of mobile applications mixed with web attributes

# Technical components - Web APIs

## Core

1. HTTPS - requirement to enable below technologies
2. Service Workers - offline ability, installability, engageability, app updates, background synchronization and more
3. Web App Manifest - installability, discoverability

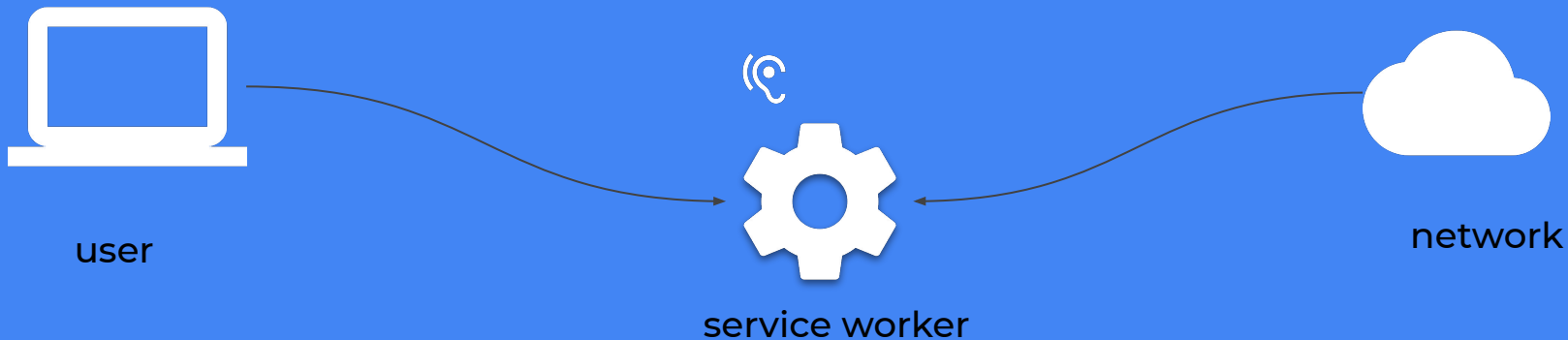
## Optional - Good to have

- Web Storage APIs - Cache(static application assets), IndexedDB(application state data), File Access API
- Web Push & Notifications API
- Background Synchronization API
- Payment Request API
- Shape Detection API
- More under development

# Service Workers

*A web worker, that acts like a proxy in the background, and sits between the application and the network.*

- No direct DOM access, runs in separate thread
- Non blocking and asynchronous - use of promises
- Can have more than one registered in each scope, e.g "example/foo", "example/bar"
- Communicates with other APIs such as Cache API, Web Push, Background Sync and more



# Service Worker Lifecycle

1. **Register** - registration of the script, if supported from the browser
2. **Install** - once in scope
  - a. Cache your static files in order to work offline, with a cache first strategy failing back to network if not available, re-update cache with new versioned file
3. **Activate** - remove old redundant cache items, update the instance if necessary
4. **Activated**
  - a. - Listing to fetch / message / push events and others, ready to take control on events.



# Web App Manifest

```
{  
  "short_name": "Depo",  
  "name": "Depo Warehouse Management System",  
  "icons": [  
    {  
      "src": "depo-512x512.png",  
      "type": "image/png",  
      "sizes": "512x512"  
    }  
  ],  
  "start_url": ".",  
  "display": "standalone",  
  "theme_color": "#000000",  
  "background_color": "#ffffff"  
}
```

# Advantages - Disadvantages

## Advantages

- Cost of development, using only web technologies and no external dependencies
- Cross platform and cross device, having a browser
- Distribution is easy, shared link or through Play Store(TWA) atm
- Deployment, as a typical web-site application, just update your server with the new assets

## Disadvantages

- Vendor browsers cold war, Google vs Apple, some features implemented only in one 'platform', but lately we see Apple to follow up
- APIs new and not yet mature - tested
- Device integration, more APIs must be developed and re-define which 'device' we are targeting
- Performance, it depends in the type of the application, for high resource applications maybe with the power of Web Assembly we will see performance demanded applications in the browser

# Live presentation of use case developed

Depo, a Warehouse Management System  
client as progressive web app

Live instance [here](#)

Thank you !