PWA

Progressive Web Application

Mi az a PWA??

Progressive

Progressive /prəˈgrɛsɪv/ adjective

happening or developing gradually or in stages.

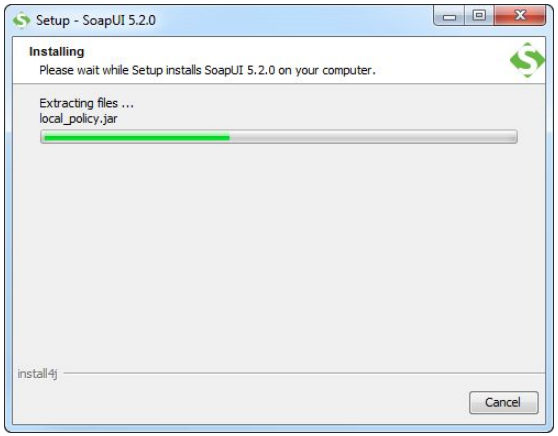
"a progressive decline in popularity"

PWA: Fokozatosan, szakaszonként fejlődő webalkalmazás

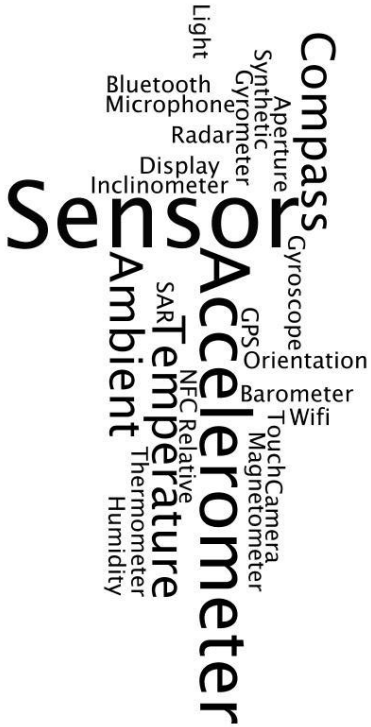
Natív App feeling webes platformon

Offline működés

Telepíthetőség



Device közeli API-k



Miért érdekeljen?

Miért érdekeljen?

PWA

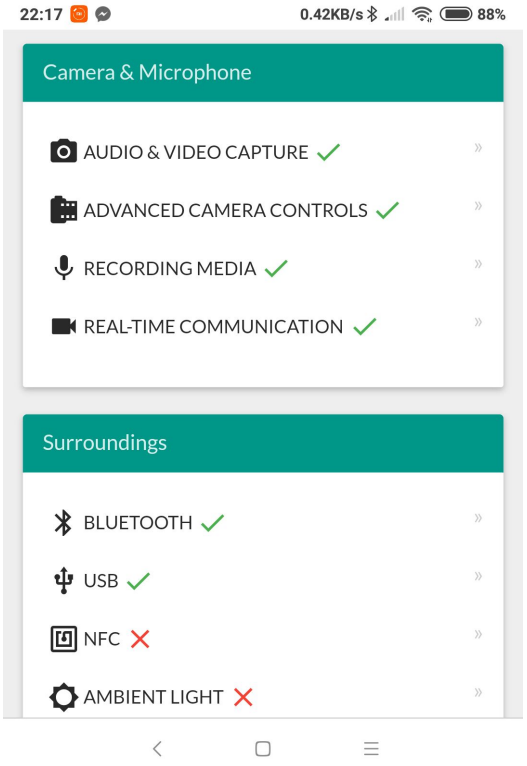
- Appstore nélkül elérhető, telepíthető
- “Fokozatosan” alakul alkalmazássá
- Költséghatékony
- Mindig friss

Natív alkalmazás

- AppStore elérhető
- Felhasználótól azonnal elköteleződést igényel
- Költségesebb
- Minden platformra külön fejlesztendő

- Jól kiegészíti a PWA a többi jelenlétet
- Fokozatosság
- Kis költségvetésnél esetleg helyettesítheti

whatwebcando.today



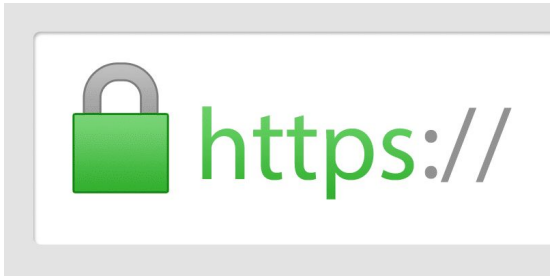
Web API-k

- Storage (LocalStorage, SessionStorage, IndexedDB)
- WebSocket
- File API
- Geolocation
- Web Push Notifications
- Push API
- Web Workers
- Bluetooth
- Vibration
- Device orientation
- Media Capture (video, audio)
- Orientation, Accelerometer
- HW accelerated 2D/3D graphics

Hogyan?

Technológiák

- HTTPS
- Promises
- Fetch API



Promise - létrehozás

```
new Promise(function (resolve, reject) { ... } )


let willIGetNewPhone = new Promise(
    function (resolve, reject) {
        if (isMomHappy) {
            var phone = new Phone()
            resolve(phone) // fulfilled
        } else {
            var reason = new Error('mom is not happy')
            reject(reason) // reject
        }
    }
)
```

Promises - használat

```
function askMom() {
    willIGetNewPhone.then(function (phone) {
        console.log(phone)
    }).catch(function (error) {
        console.log(error)
    })
}
```

```
async function askMom() {
    try {
        let phone = await willIGetNewPhone
        console.log(phone)
    } catch (error) {
        console.log(error)
    }
}
```

fetch()

```
try {

    const request = new Request(url, {
        method: 'POST',
        body: 'SomeData'
    })

    let response = await fetch(request)

} catch (error) {
    console.log(error)
}
```

Web App Manifest

Alkalmazás

- Neve
- Ikonja
- Splash screen
- Megjelenés
  - standalone
  - fullscreen
- Orientation

```
{
        "short_name": "PWA test",
        "name": "PWA test",
        "icons": [
                {
                        "src": "w9-192.png",
                        "type": "image/png",
                        "sizes": "192x192"
                },
                {
                        "src": "w9-512.png",
                        "type": "image/png",
                        "sizes": "512x512"
                }
        ],
        "start_url": "/pwa/",
        "background_color": "#ccf",
        "display": "standalone",
        "scope": "/pwa/",
        "theme_color": "#008"
}
```

```
<link rel="manifest" href="...">
```

Service Worker

- Cache API
- Fetch API
- Push API
- Background Sync API

Service Worker - registration

```
var savedBIPEvent
function onBeforeInstallPrompt(evt) {
    evt.preventDefault()
    savedBIPEvent = evt          // Later: savedBIPEvent.prompt()
}

if ('serviceWorker' in navigator) {
    (async function () {
        try {
            let reg = await navigator.serviceWorker.register('sw.js')
            window.addEventListener('beforeinstallprompt', onBeforeInstallPrompt)
        } catch (err) {
            console.error('Service Worker registration failed: ', err)
        }
    }())
}
```

Service Worker - Életciklus

- install
  - telepítéskor fut
  - új verziókor újra
  - Cache inicializálás
- activate
  - upgrade esetén megvárja, amíg a futó oldalak bezáródnak
  - Régi cache cleanup

```
self.addEventListener('install', onInstall)
self.addEventListener('activate', onActivate)

self.addEventListener('fetch', onFetch)
self.addEventListener('push', onPush)
self.addEventListener('pushsubscriptionchange', onPushC)
self.addEventListener('sync', onSync)
```

Cache policy

- Offline first
- Network first
- Native-like

Service Worker - install event

```
const CACHE = 'cache-v1'
const PRECACHE_URLS = [
    './',
    'main.js',
    'image.png'
]

function onInstall(evt) {
    evt.waitUntil(async function() {
        let cache = await caches.open(CACHE)
        await cache.addAll(PRECACHE_URLS)
        self.skipWaiting()
    }());
}
```

Service Worker - activate event

```
const CACHE = 'cache-v2'

function onActivate(evt) {
    evt.waitUntil(async function() {
        let cacheList = (await caches.keys()).filter(name => name !== CACHE)
        await Promise.all(cacheList.map(name => caches.delete(name)))
        await self.clients.claim()
    }())
}
```

Service Worker - fetch event

```
function onFetch(evt) {
    if (evt.request.method !== 'GET' || !evt.request.url.startsWith(self.location.origin)) return
    evt.respondWith(async function() {
        let res = await caches.match(evt.request)
        if (res) return res
        let cache = await caches.open(CACHE)
        try {
            res = await fetch(evt.request)
            await cache.put(evt.request, res.clone())
            return res
        } catch(err) {
            return null
        }
    }())
}
```

Service Worker - push eventek

```
function onPush(evt) {
    const {title, body, icon} = evt.data.json()
    evt.waitUntil(
        self.registration.showNotification(title, {body, icon})
    )
}

function onPushSubscriptionChange(evt) {
    evt.waitUntil(
        self.registration.pushManager.subscribe({ userVisibleOnly: true })
        .then(function(subs) {
            /* Save subscription */
        })
    )
}
```

Köszönöm a figyelmet!

Hajba Szilárd

products.symbion.hu

szilard@symbion.hu