Maximiliano Firtman
@firt firt.dev

JavaScript in the Background

JS

FrontendMasters

mobile+web developer & trainer

JS

PWA

Maximiliano Firtman
@firt firt.dev

- HTML since 1996
- JavaScript since 1998
- Authored 13 books
- Published 150+ webapps
- Publish **web.dev/learn/pwa**
- +20 workshops on Web Push

Zombie Tab

Background notification hack for iPad and iPhone websites

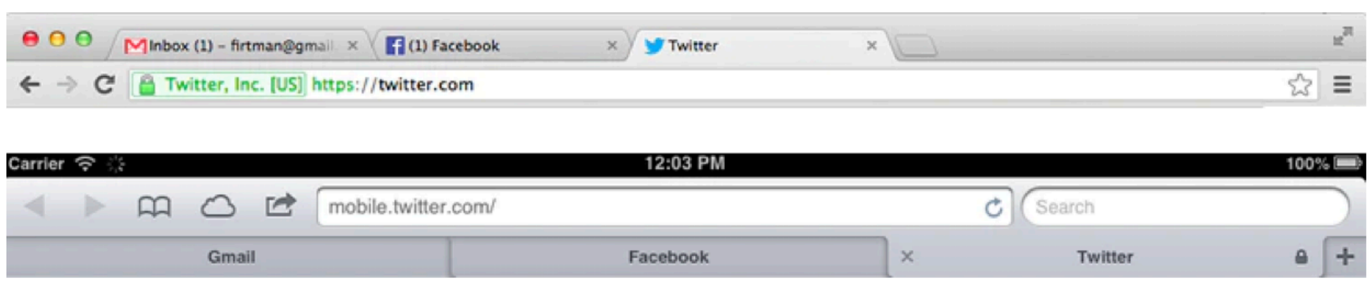
by **Maximiliano Firtman**
Twitter @firt About Newsletter

Published 9 years ago (09 May 2013)
About 4 min reading time

#IOS

Mobile browsers may look like desktop browsers, but their behavior sometimes is different and we need to understand them to provide the right user experience. In this post I'll show you a trick to notify the user about an update while our website is on a background tab.

> **Table of Contents:**



At the top, Chrome for desktop showing updates on background tabs; at the bottom, Safari for iPad freezing background tabs

What we'll cover

- Web App Lifecycle
- Background's Life
- Detection
- Notifications
- Page Visibility API
- Service Workers
- Execution in the Background
- Web Push

Pre-requisites

github.com/firtman/javascript-background

Questions?

JavaScript in the Background

1

Introduction

Let's reveal the mystery

Classic Web Development

Is it still valid?

First: what's background for code execution?

Background

We don't have a definition for web apps

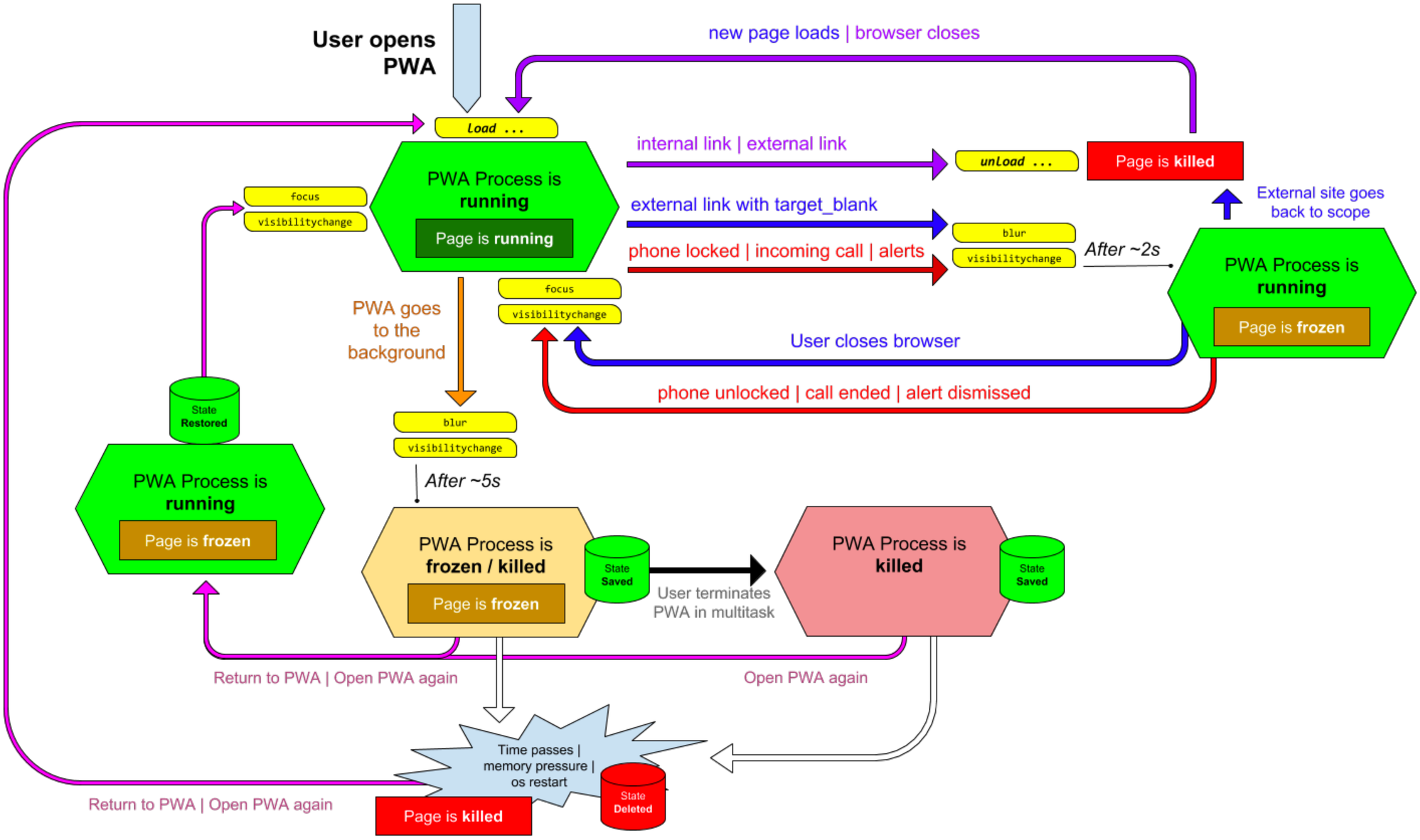
W3C have different definitions for different APIs

There are many concepts around:

- **hide/minimize**: in memory and with some execution rights
- **suspend/freeze**: in memory with no execution rights
- **close/kill/discard:** not in memory

We will try to simplify it

Background: when the user stops or pauses the usage of the web app


User opens PWA
new page loads | browser closes
load ...
PWA Process is running
Page is running
internal link | external link
unload ...
Page is killed
External site goes back to scope
focus
visibilitychange
external link with target_blank
blur
visibilitychange
phone locked | incoming call | alerts
After ~2s
PWA Process is running
Page is frozen
PWA goes to the background
focus
visibilitychange
User closes browser
phone unlocked | call ended | alert dismissed
State Restored
blur
visibilitychange
After ~5s
PWA Process is running
Page is frozen
PWA Process is frozen / killed
Page is frozen
State Saved
User terminates PWA in multitask
PWA Process is killed
State Saved
Return to PWA | Open PWA again
Open PWA again
Time passes | memory pressure | os restart
State Deleted
Return to PWA | Open PWA again
Page is killed

We will understand what
happen on most situations,
how to handle it property and
how to increase the
opportunity to execute code.

Why should we care about the background?

- Improve User Experience
- Save Resources
- Stop Timers
- Abort Pending Requests

JavaScript in the Background

2

Web App Lifecycle

Possibilities

User goes to a new tab

User goes to a new window

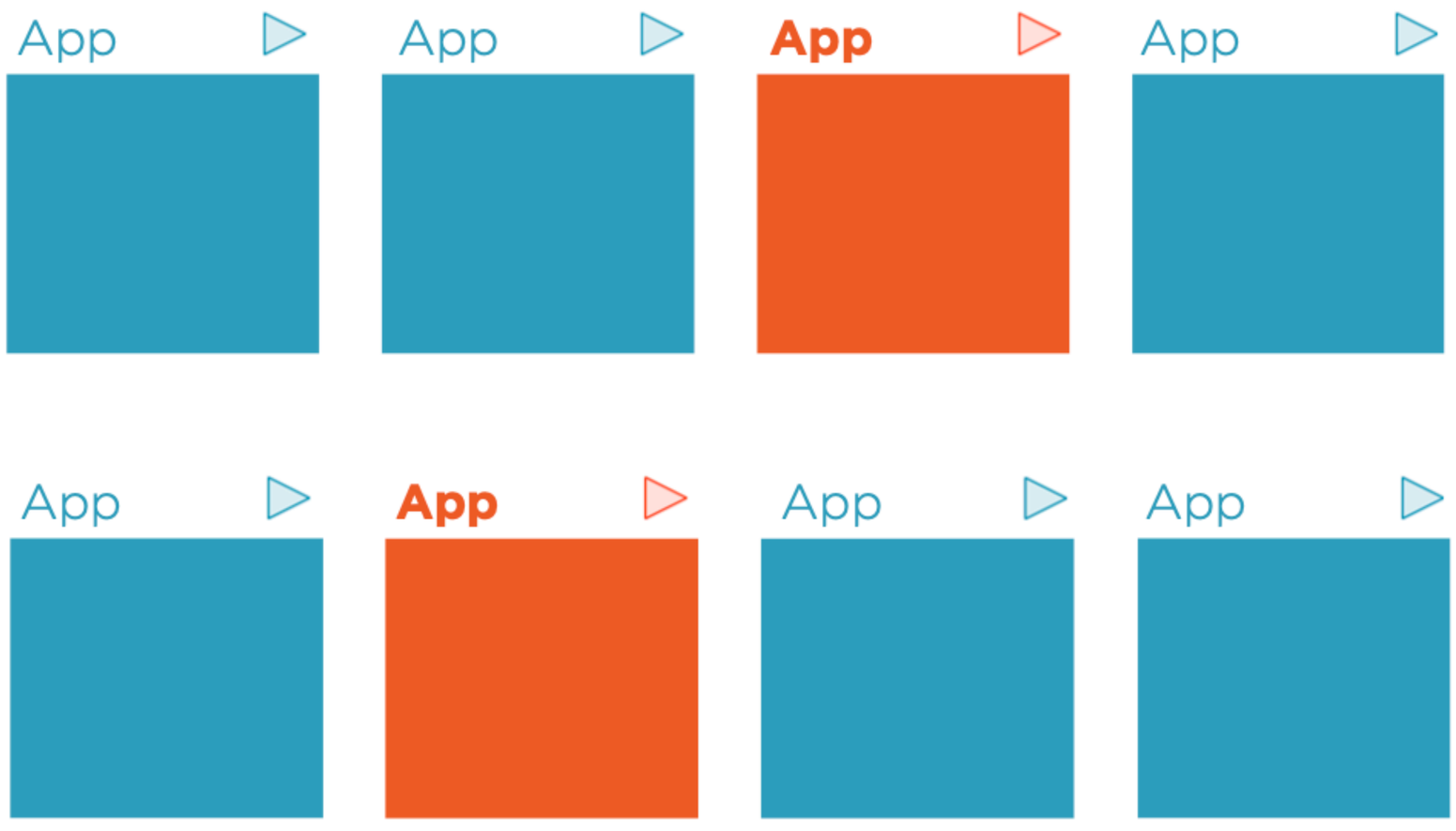
User minimizes the window

User closes the window

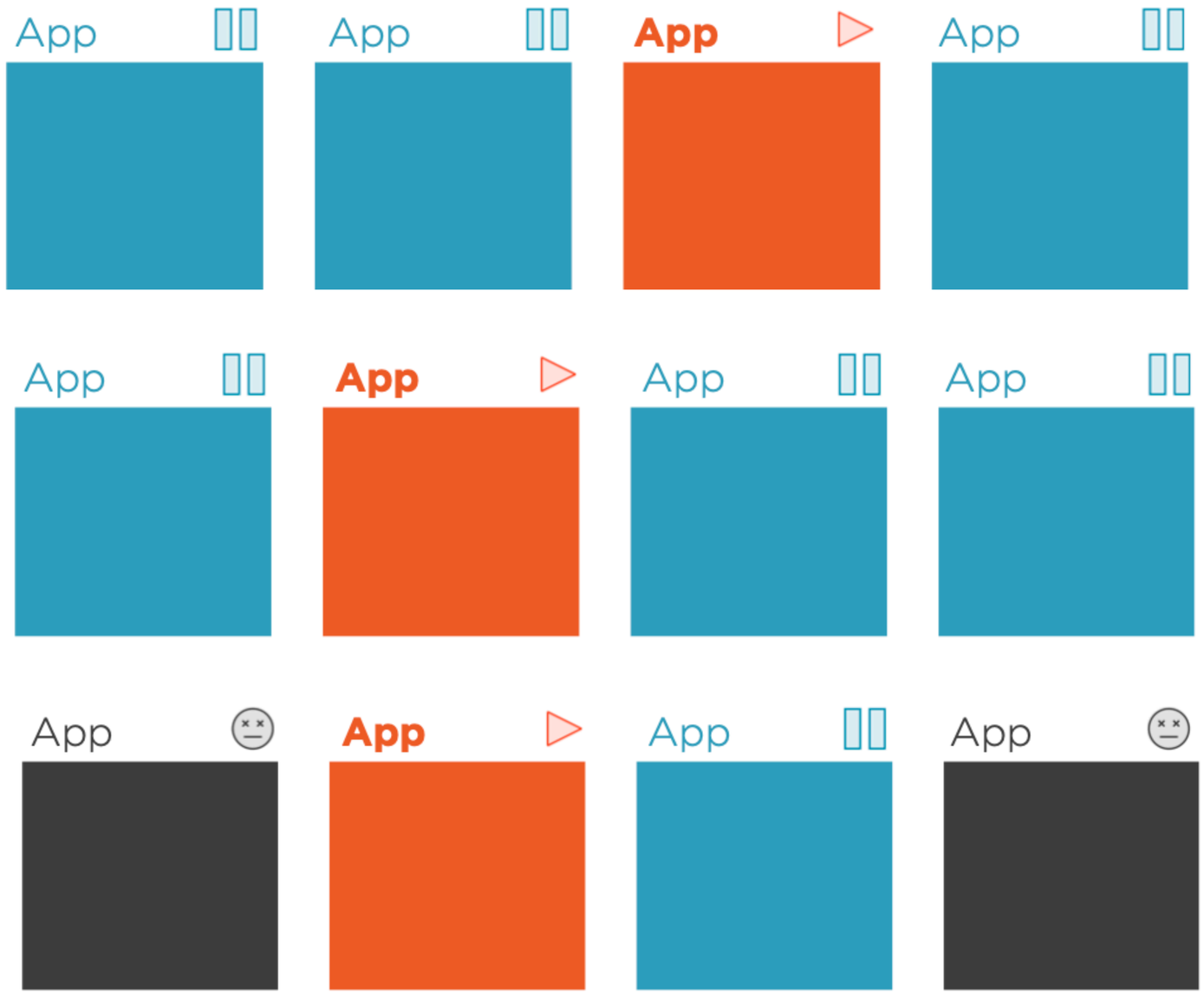
User closes the browser or PWA

What happens on mobile OSs with app's lifecycle in general?

On desktop OS



On mobile OS



We can see this in action for our web apps and PWAs

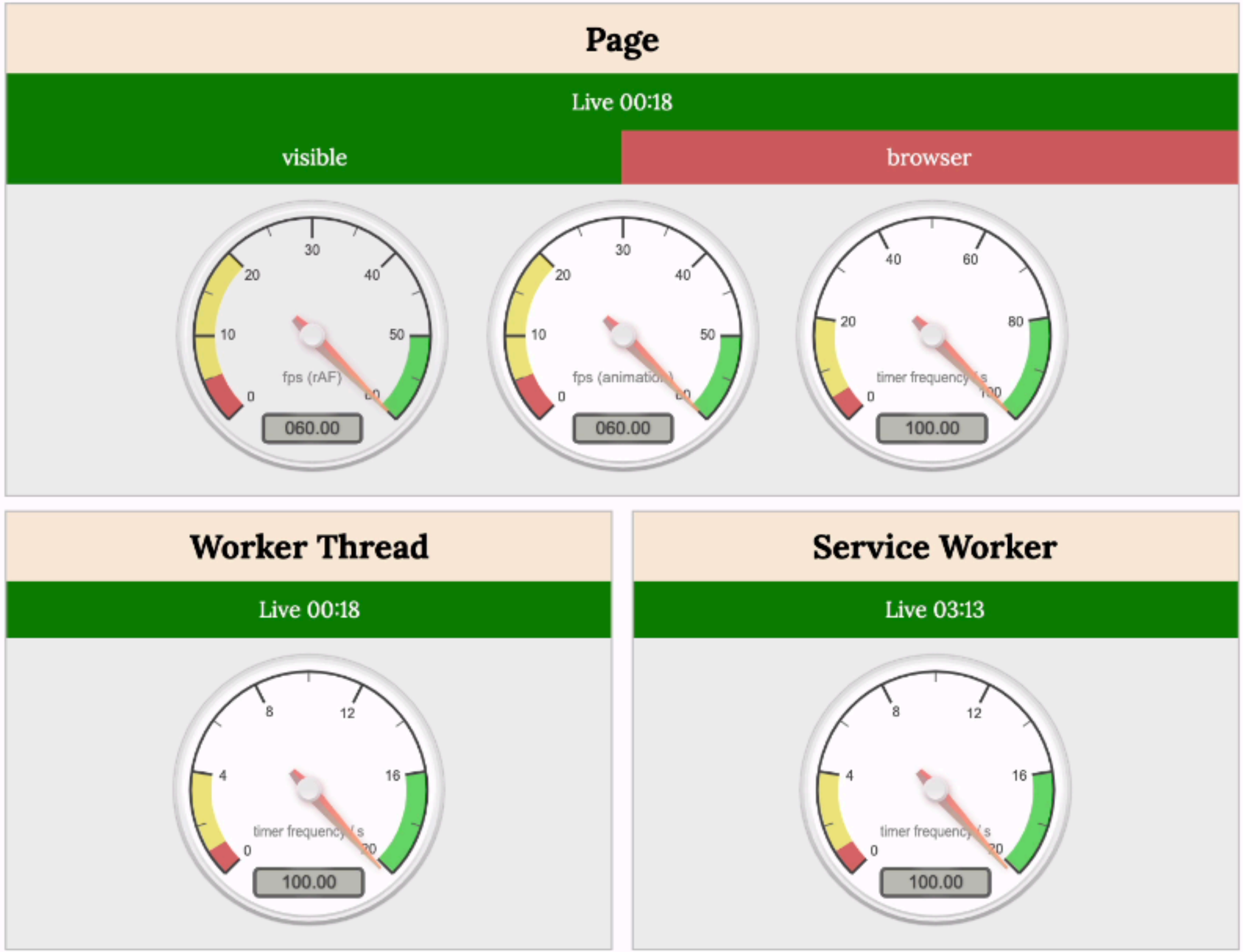
We will be start spying in the background:

FPS

Timers

Web Worker Execution

Service Worker Execution


Page
Live 00:18
visible
browser
fps (rAF)
060.00
fps (animation)
060.00
timer frequency/s
100.00
Worker Thread
Live 00:18
timer frequency/s
100.00
Service Worker
Live 03:13
timer frequency/s
100.00

Spy me!

localhost:3000

Mysterious Life of this Page in the Background

Hidden Cam

localhost:3000/dashboard/

Chrome Browser

Chrome

Page

Live 02:11

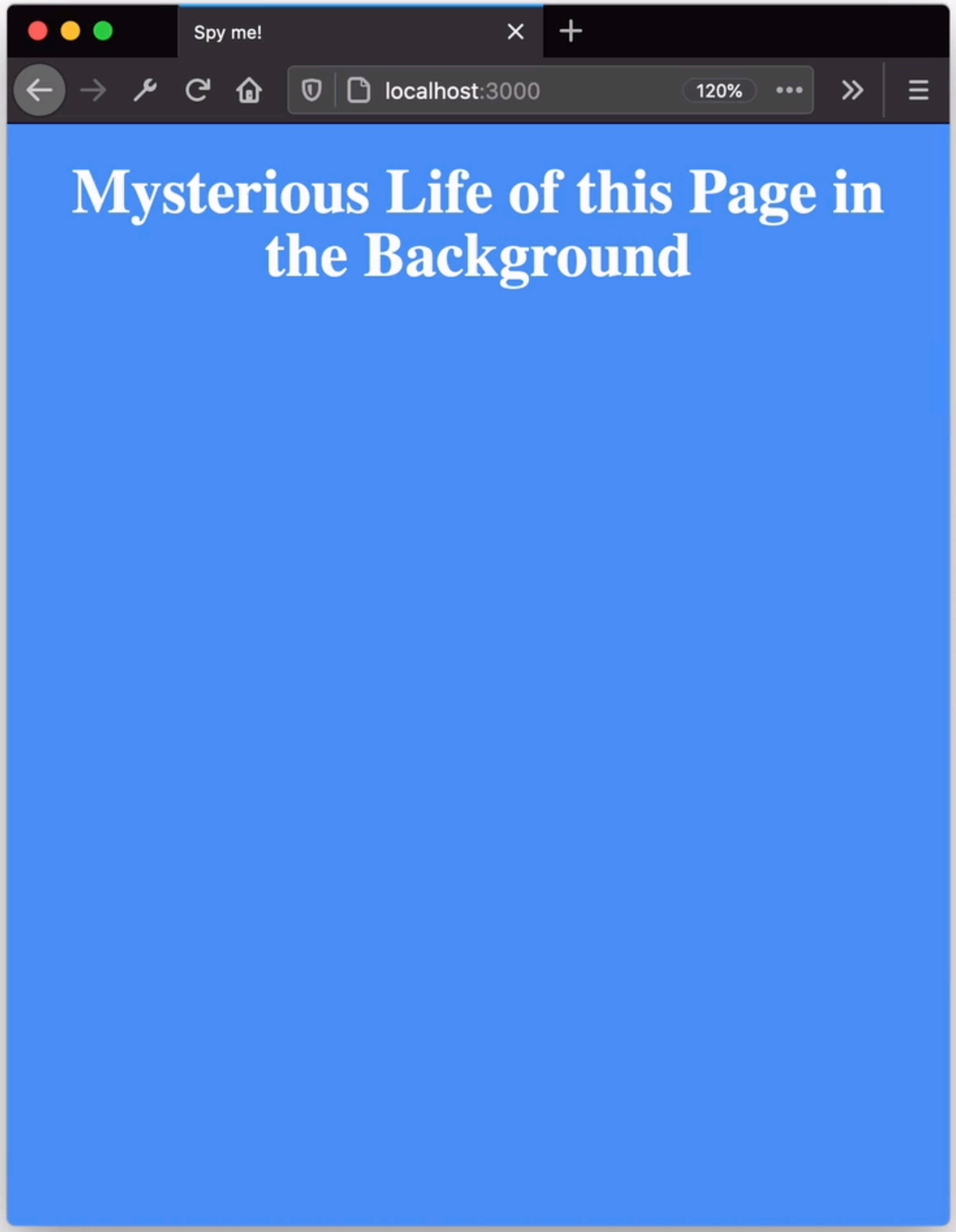
visible | browser

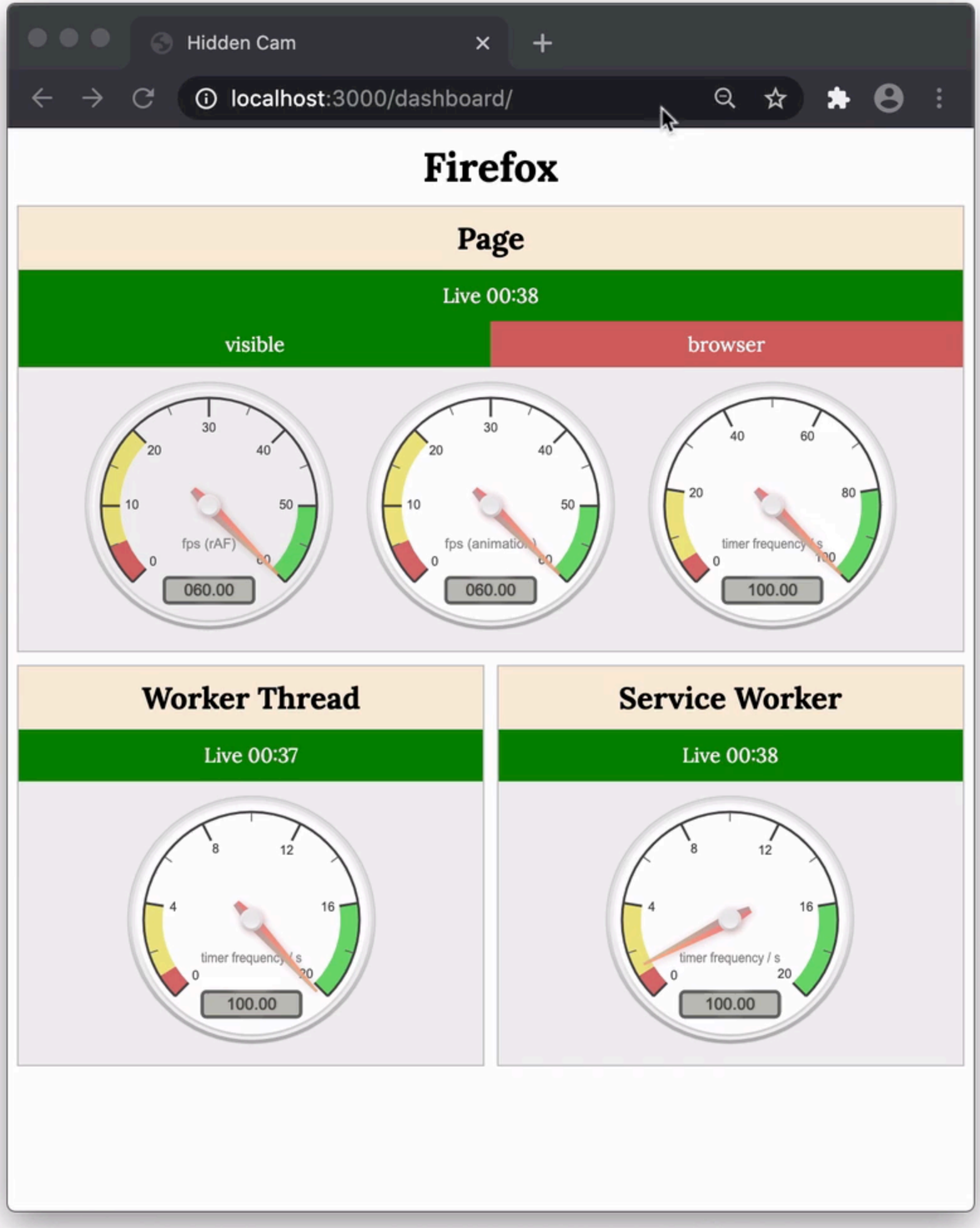
Worker Thread

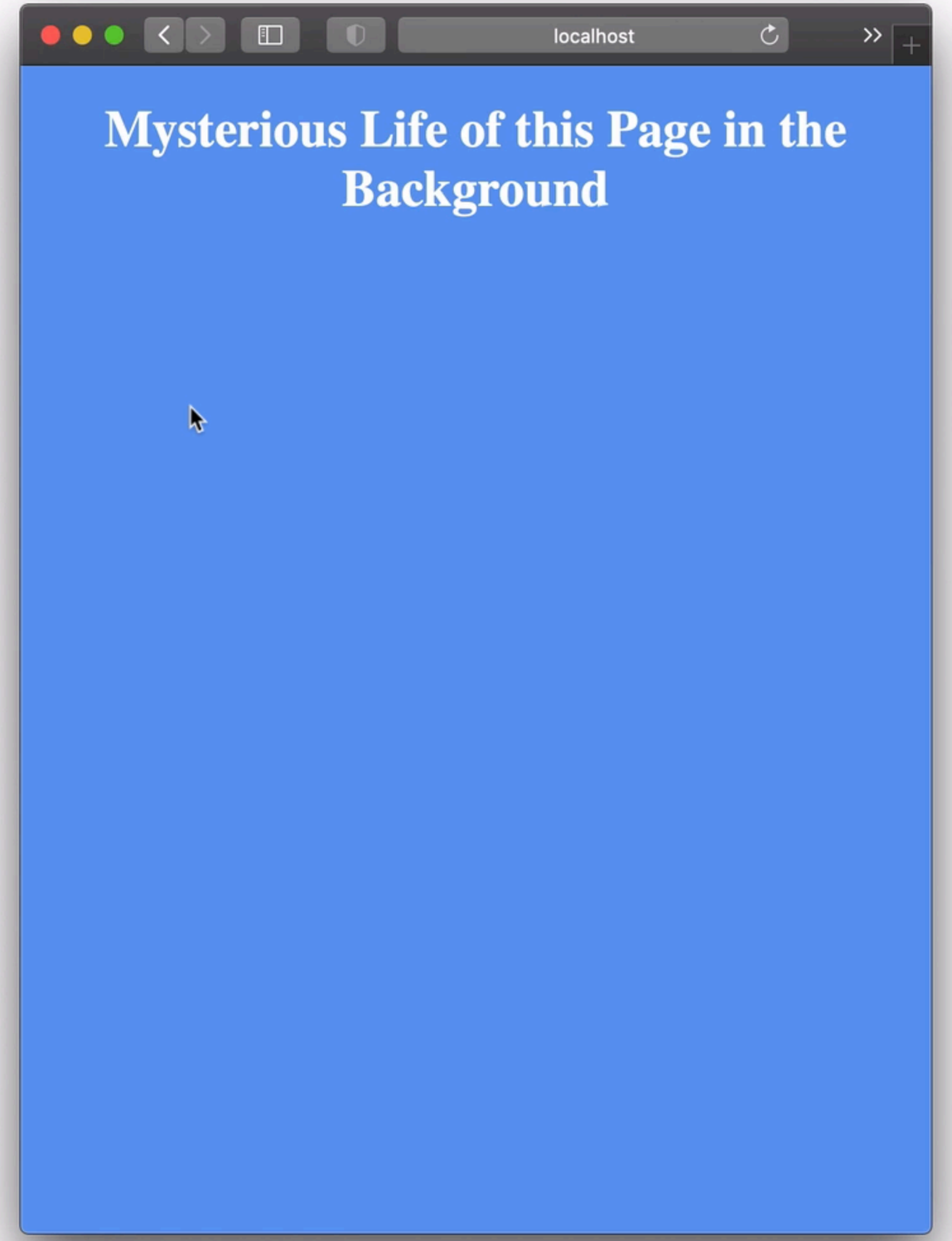
Live 02:11

Service Worker

Live 05:25


Spy me!
localhost:3000
120%
Mysterious Life of this Page in the Background


Hidden Cam
localhost:3000/dashboard/
Firefox
Page
Live 00:38
visible
browser
fps (rAF)
060.00
fps (animation)
060.00
timer frequency / s
100.00
Worker Thread
Live 00:37
timer frequency / s
100.00
Service Worker
Live 00:38
timer frequency / s
100.00


localhost
Mysterious Life of this Page in the Background

Hidden Cam
localhost:3000/dashboard/
Safari desktop
Page
Live 00:30
visible
browser
fps (rAF)
060.00
fps (animation)
059.00
timer frequency
100.00
Worker Thread
Live 00:29
095.00
Service Worker
Live 00:26
095.00

BACKGROUND APPS IN DESKTOP OSs

Windows, macOS, Linux, Chrome OS

Not visible:

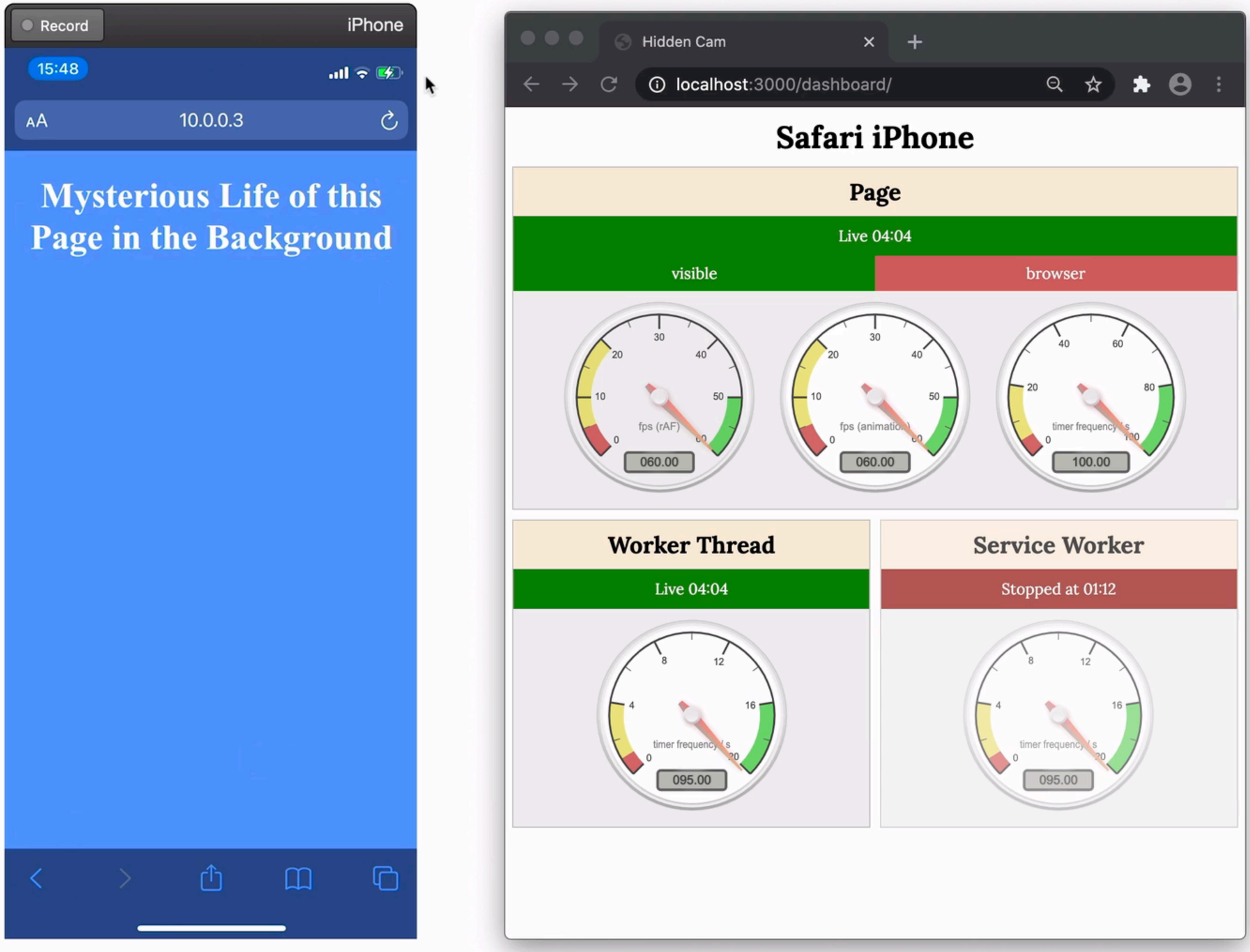
- Background Tab
- Background Browser
- Background PWA
- Totally covered

All Web Apps opened (visible or not):

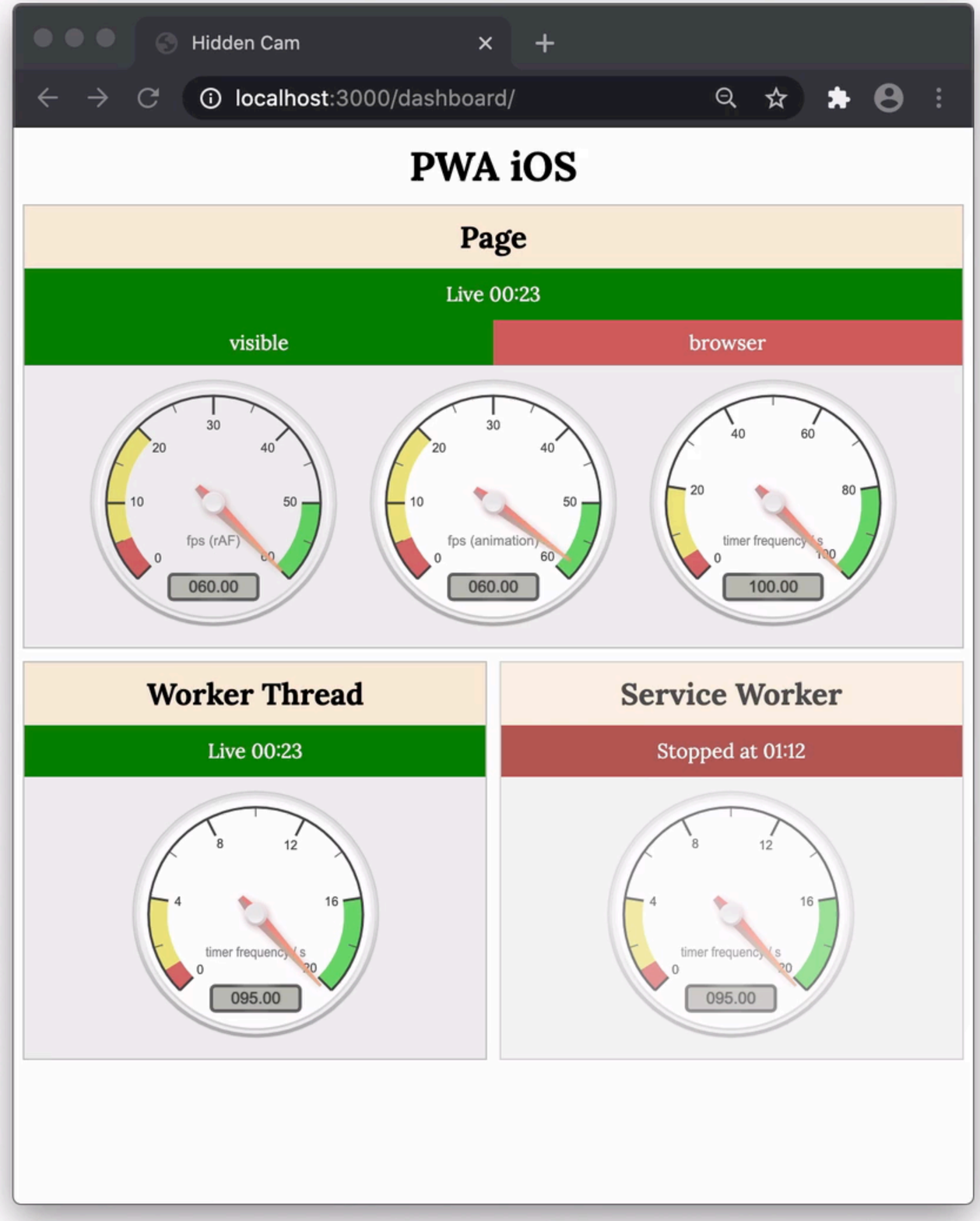
- Are in memory with execution rights

Web Apps not visible:

- May limit timers (1 sec to 10 sec)
- May suspend them (typically on Chromium alternative browsers)
- SW has its own lifecycle


Record
iPhone
15:48
AA
10.0.0.3
Mysterious Life of this Page in the Background
Hidden Cam
localhost:3000/dashboard/
Safari iPhone
Page
Live 04:04
visible
browser
fps (rAF)
060.00
fps (animation)
060.00
timer frequency / s
100.00
Worker Thread
Live 04:04
timer frequency / s
095.00
Service Worker
Stopped at 01:12
timer frequency / s
095.00

Record
iPhone
15:59
Mysterious Life of this Page in the Background


Hidden Cam
localhost:3000/dashboard/
PWA iOS
Page
Live 00:23
visible
browser
fps (rAF)
060.00
fps (animation)
060.00
timer frequency / s
100.00
Worker Thread
Live 00:23
timer frequency / s
095.00
Service Worker
Stopped at 01:12
timer frequency / s
095.00

BACKGROUND APPS IN MOBILE OSs

iOS, iPadOS

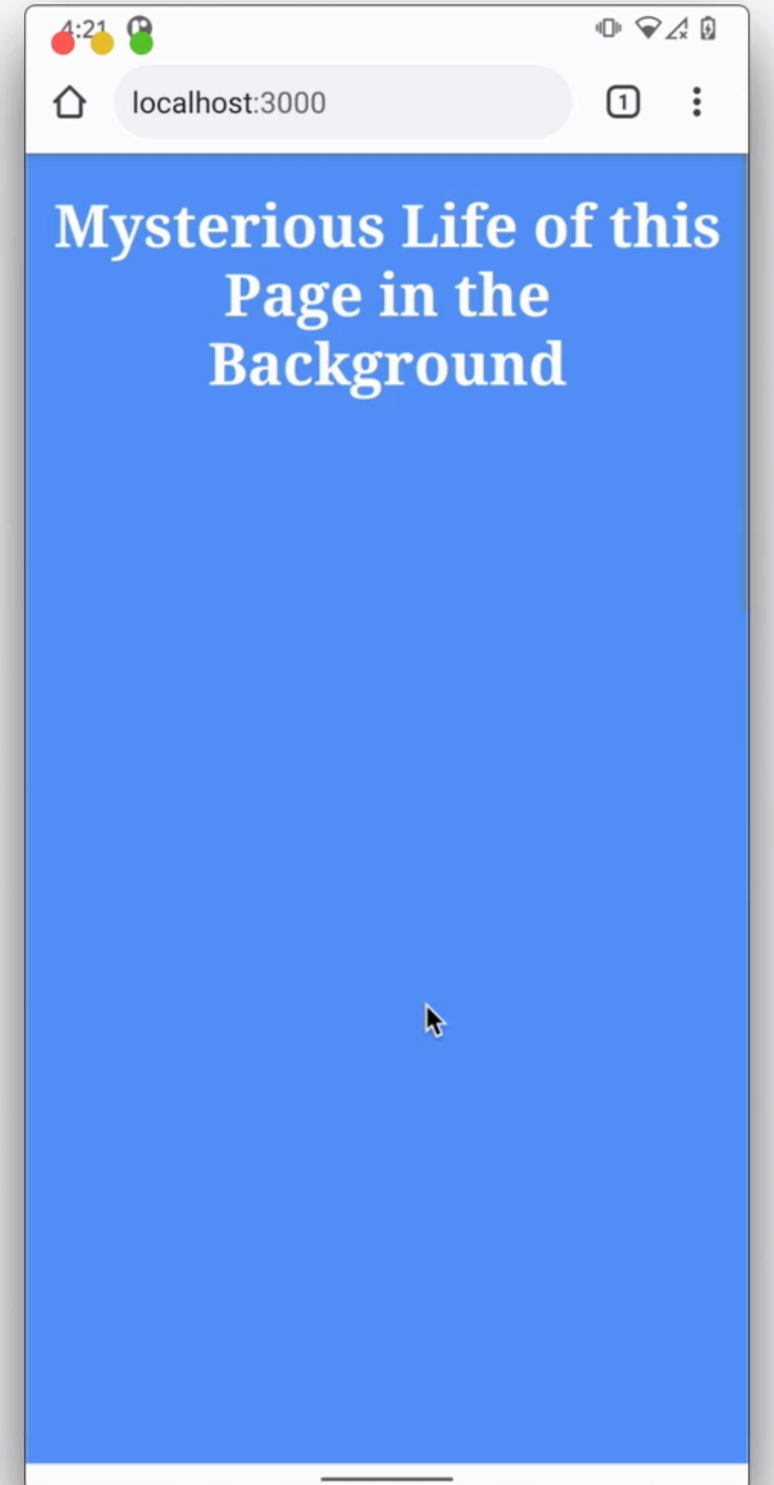
Not visible:

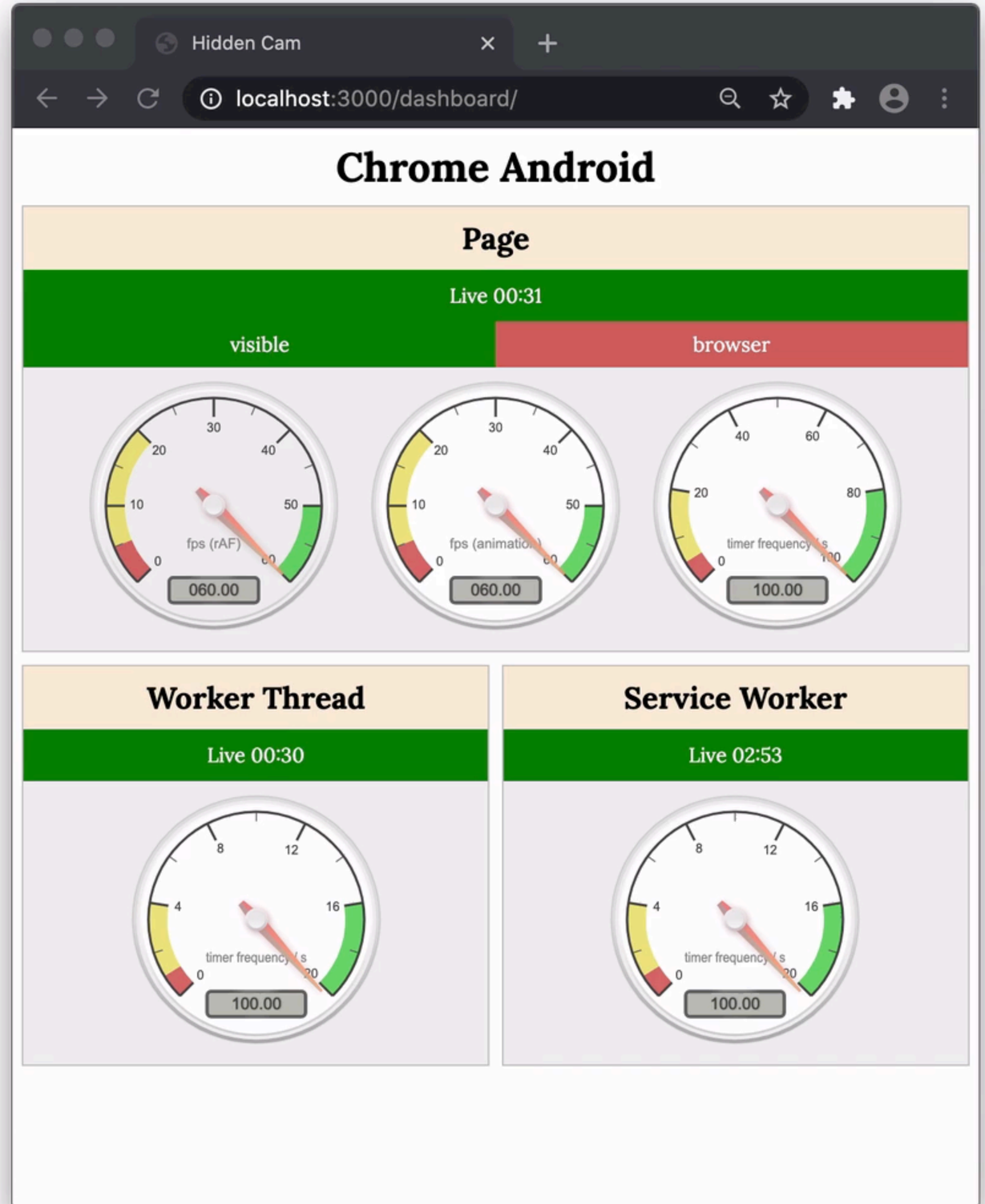
- Lock screen
- Home screen
- Open other app
- Open other tab

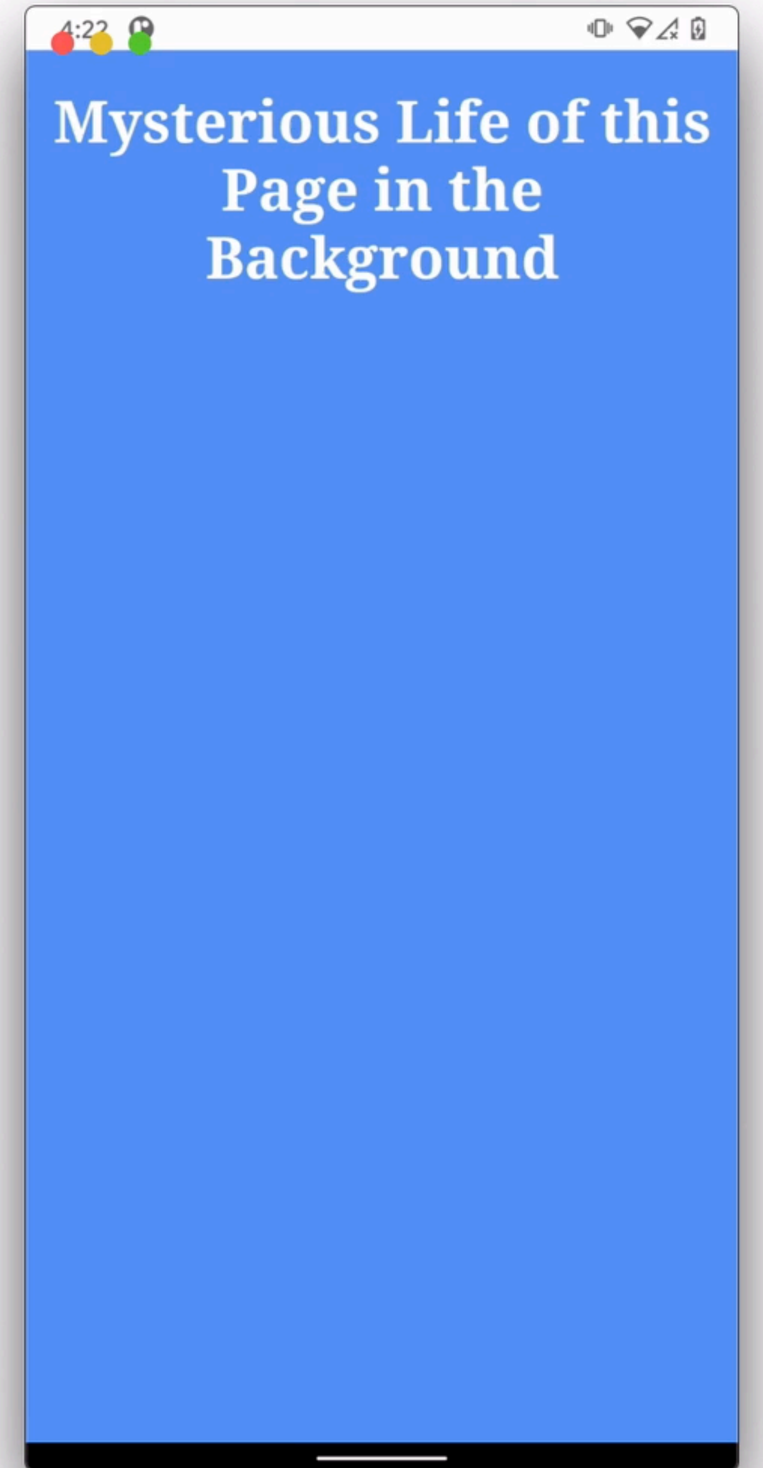
Web Apps in the background

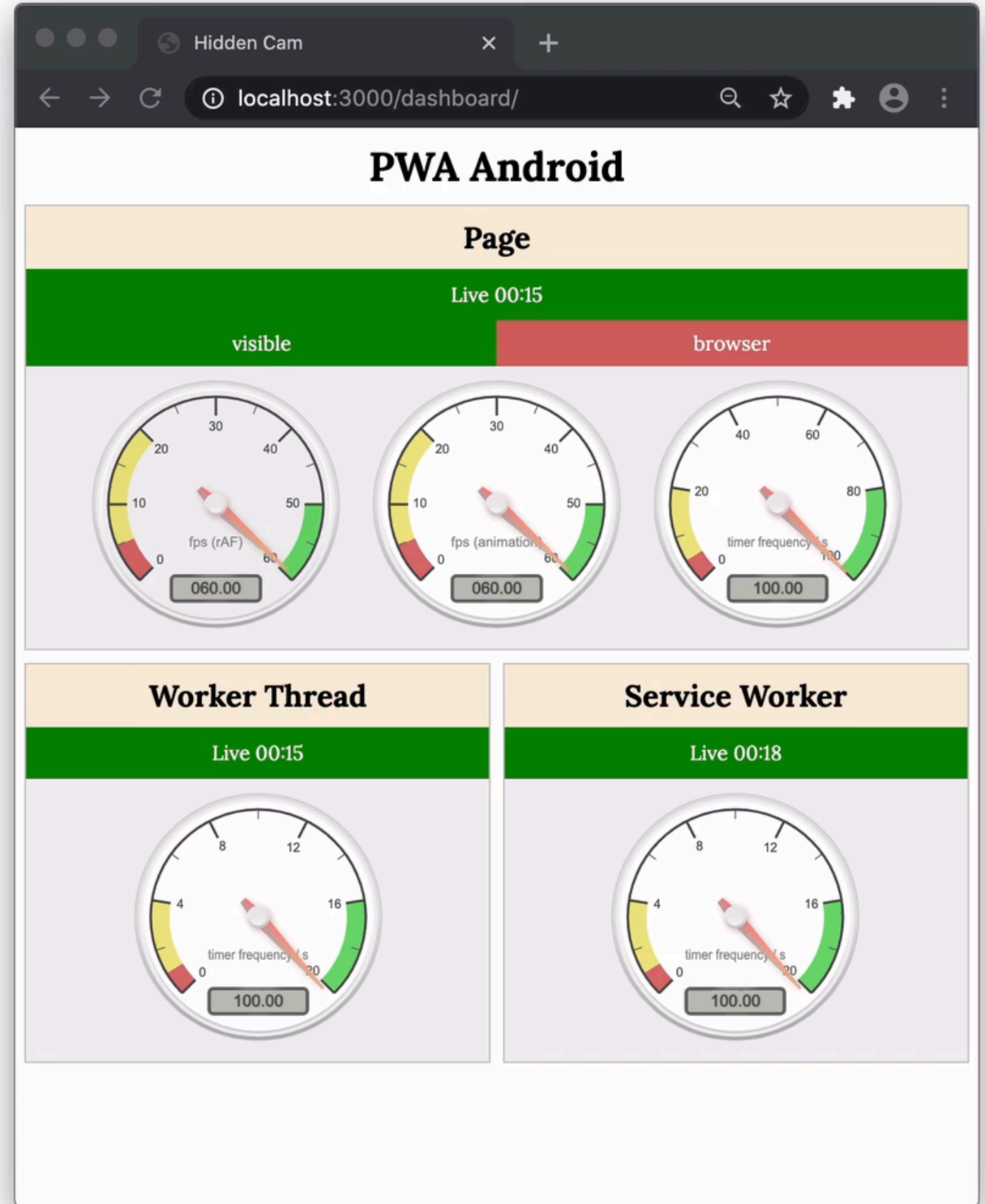
- Page Navigations in memory or not
- UI is just a screenshot, not updated
- 2 seconds or 30 seconds of execution after leaving foreground
- If PWA still in memory, opening other PWAs also give our background PWA execution rights

😯


4:21
localhost:3000
Mysterious Life of this Page in the Background


Hidden Cam
localhost:3000/dashboard/
Chrome Android
Page
Live 00:31
visible
browser
fps (rAF)
060.00
fps (animation)
060.00
timer frequency / s
100.00
Worker Thread
Live 00:30
timer frequency / s
100.00
Service Worker
Live 02:53
timer frequency / s
100.00


4:22
Mysterious Life of this Page in the Background


Hidden Cam
localhost:3000/dashboard/
PWA Android
Page
Live 00:15
visible
browser
fps (rAF)
060.00
fps (animation
060.00
timer frequency
100.00
Worker Thread
Live 00:15
timer frequency
100.00
Service Worker
Live 00:18
timer frequency
100.00

BACKGROUND APPS IN MOBILE OSs

Android

Not visible:

- Lock screen
- Home screen
- Multitask
- Open other app
- Open other tab

Web Apps in the background

- Page Navigations in memory or not
- UI is just a screenshot, not updated
- Chrome: Up to 5 minutes of execution in background (30s+ for Service Worker)
- Firefox: Page is stopped immediately, SW: 30s, Worker: 3 minutes

BACKGROUND SUMMARY

	Desktop Chromium	Desktop Firefox	Desktop Safari	Android Chromium	Android Firefox	iOS & iPadOS
Freezing	Possible	No	No	5 minutes	Immediately	2s - 30s
Destroy	Possible	No	No	Yes Managed by the OS		
Timer Throttling	1s 1m after 5m	1s	10s	1s	1s	500ms
Worker Thread	Unlimited	Unlimited	Unlimited	5m	3m	2s - 30s
Service Worker	5.5m	30s	10s	5.5m	30s	10s

- Every browser can change this behavior anytime

I still want to execute code in the background

Execution Rights in the Background

Options Available Today

Google Play Store PWA

Trusted Web Activity for PWA and native code for background

App Store PWA

WebView for the user interface in the PWA and Native code for background (it can be JS)

Service Worker Background APIs

Additions that will be possible in the browser and with installed PWAs (it works on Play Store PWAs also)

Audio and Video playing

Let the web app run in the background providing video and audio data to keep it playing in the background

SERVICE WORKER BACKGROUND APIS

Abilities on top of the main spec

Compatibility differs per browser

There may be no client running at the time (visible UI) when used

- Web Push
- Background Sync
- Periodic Background Sync
- Background Fetch
- Payment Handler

SERVICE WORKER BACKGROUND APIS

- Web Push
- Background Sync
- Periodic Background Sync
- Background Fetch
- Payment Handler

*** Some non-Chrome browsers may disable APIs**

SERVICE WORKER BACKGROUND APIS

- Web Push*
- *Background Sync*
- *Periodic Background Sync*
- *Background Fetch*
- *Payment Handler*

*** Only on macOS Ventura. iOS and iPadOS coming during 2023**

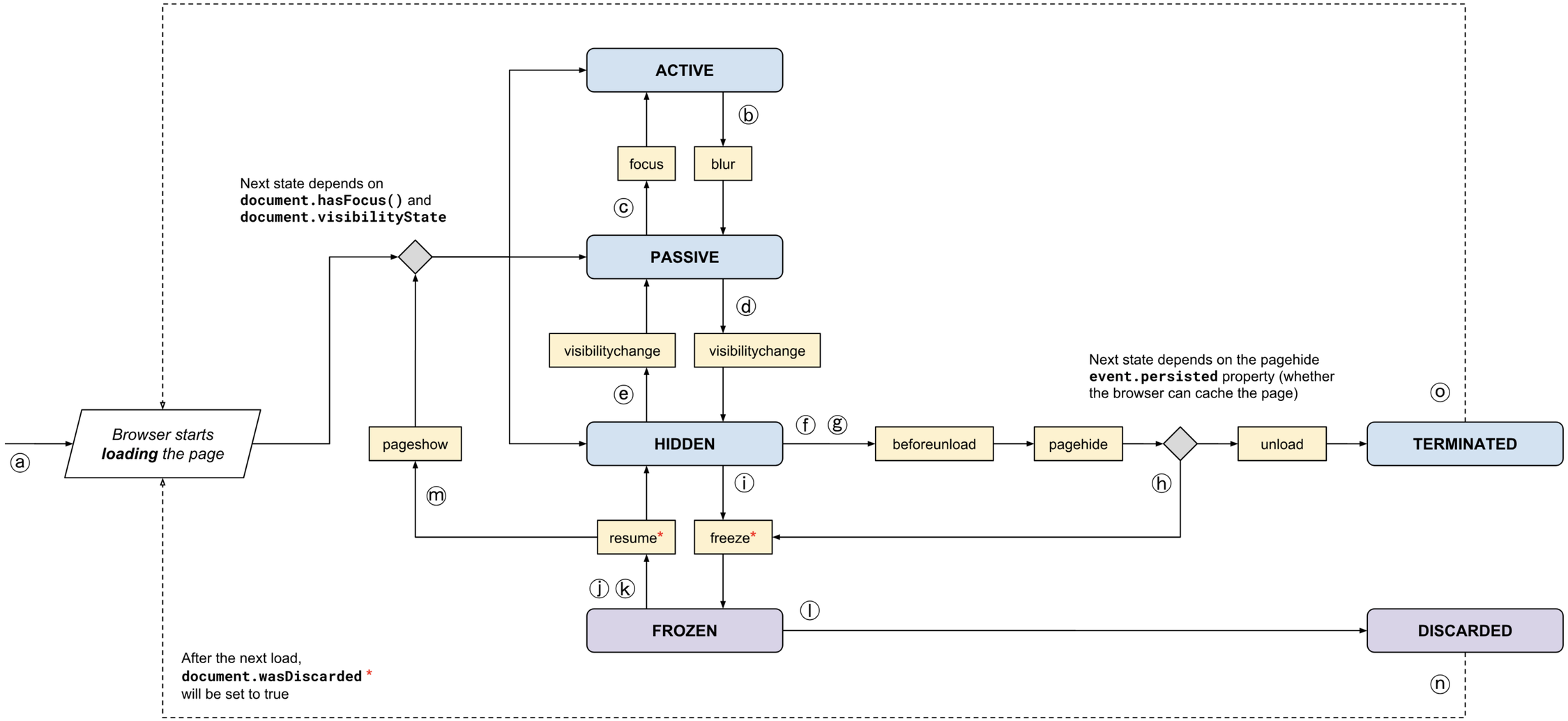
SERVICE WORKER BACKGROUND APIS

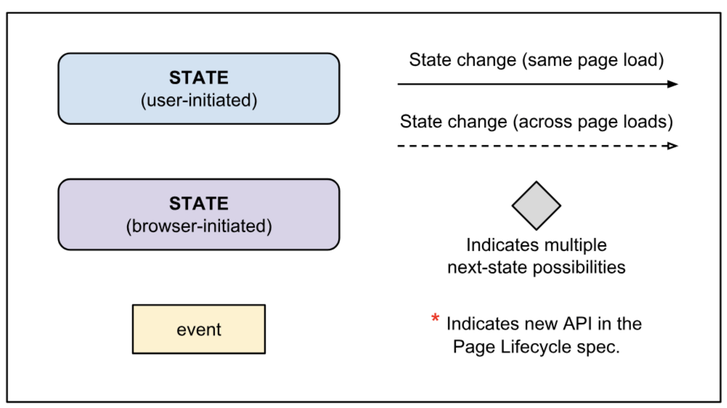
- Web Push
- *Background Sync*
- *Periodic Background Sync*
- *Background Fetch*
- *Payment Handler*

*** Some non-Chrome browsers may disable APIs**

3

Background Detection





ⓐ User navigates to the new page
ⓑ User focuses a page in another tab, window, or app
ⓒ User re-focuses the page
ⓓ User switches tabs away from the page
ⓔ User switches tabs back to the page
ⓕ User navigates to another page
ⓖ User closes the tab, window, or app
ⓗ System puts page into the page cache
ⓘ System freezes the page to conserve CPU
ⓙ System resumes the frozen page
ⓚ User switches tabs back to the frozen page
ⓛ System discards the page to conserve memory
ⓜ User navigates back/forward to the cached page
ⓝ User switches tabs back to the discarded page
ⓞ User navigates back to a terminated page

PWA LifeCycle

App Loading

PWA LifeCycle

App Loading

↓

App goes background

⚡ visibilitychange

PWA LifeCycle

App Loading

↓

App goes background

↓

App might be suspended

⚡ visibilitychange

⚡ freeze

PWA LifeCycle

App Loading

↓

App goes background

↓

App will be suspended

↓

User goes back to App

App in memory

visibilitychange

freeze

PWA LifeCycle

App Loading

↓

App goes background

↓

App will be suspended

↓

App in memory

User goes back to App

⚡ visibilitychange

⚡ freeze

⚡ resume

⚡ visibilitychange

PWA LifeCycle

App Loading

↓

App goes background

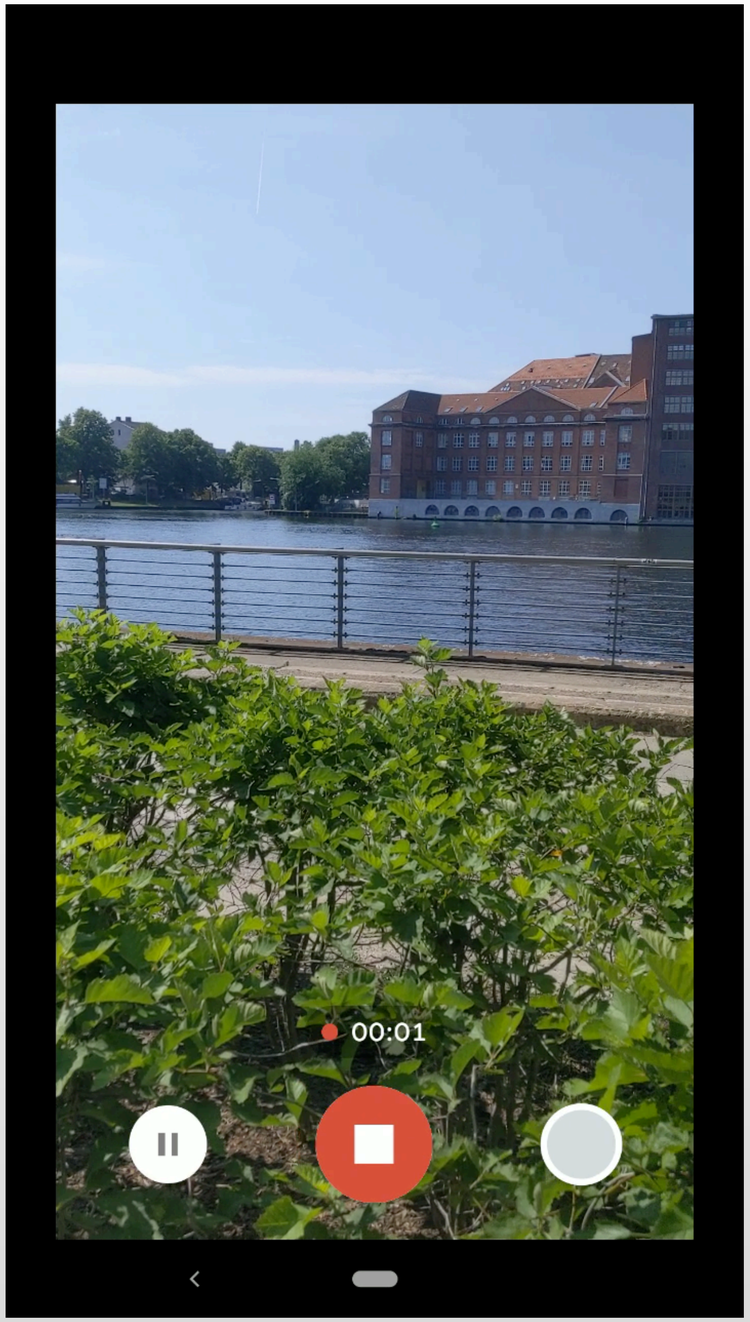
↓

App will be suspended

App was discarded

⚡ visibilitychange

⚡ freeze



PWA LifeCycle

App Loading

↓

App goes background

↓

App will be suspended

↓

App was discarded

User goes back to App

⚡ visibilitychange

⚡ freeze

❓ document.wasDiscarded

⚡ load

From our web app to another (web) app

Desktop

PWA still in memory and running

- ⚡ `visibilitychange`
- ⚡ `blur`

Mobile and some desktop situations

PWA goes to a suspended state

- ⚡ `visibilitychange`
- ⚡ `freeze` (Chromium only)

At any time, it can be discarded

From the other (web) app to our web app

Desktop

Makes visible the PWA

⚡`visibilitychange`

⚡`focus`

Mobile and some desktop situations

Resuming a suspended state

⚡`visibilitychange`

⚡`resume` (Chromium only)

PWA was discarded from memory

⚡`load`

`document.wasDiscarded == true`

(Chromium only)

Detection

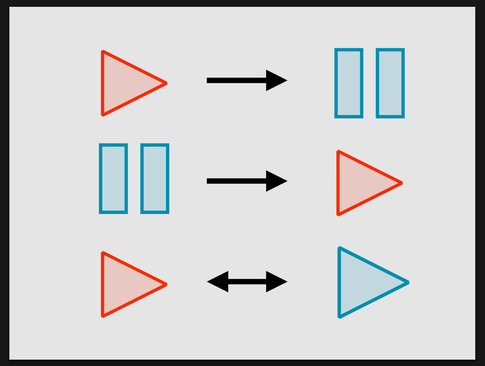
Because there are many possibilities for background, there are many APIs and ways to detect getting in/out of "background".

Only one is cross-platform and cross-possibility: **Page Visibility API**

Visibility Change Detection

All platforms

script.js



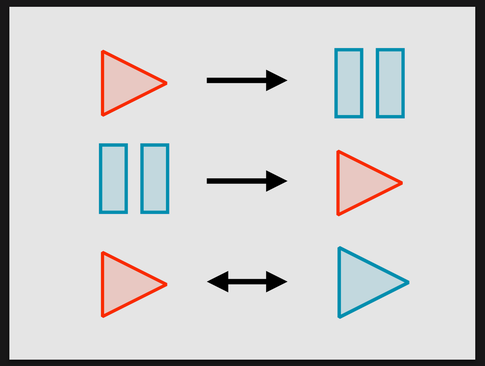
Visibility Change Detection

All platforms

script.js

```
window.addEventListener('visibilitychange', event => {

});
```

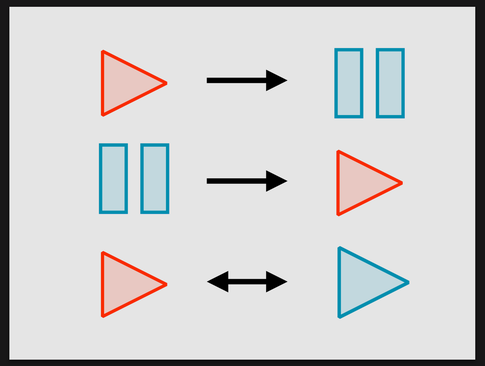


Visibility Change Detection

All platforms

script.js

```
window.addEventListener('visibilitychange', event => {
  if (document.visibilityState === 'hidden') {
      // We are in the background
      // on some devices, last chance to save current state
  } else {
     // We are back in the foreground
  }
});
```



Workshop time

Saving a timestamp with Page Visibility API will let us know how much time has passed since the user left our web app

Freezing Detection

Page Lifecycle API let us detect if a page navigation is frozen by the browser and/or if it's resumed

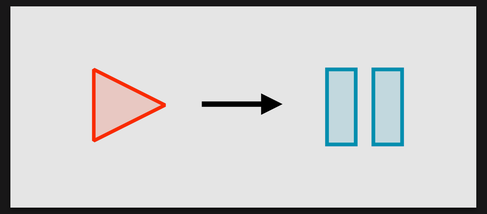
Today is available on some Chromium-browsers, such as Google Chrome for desktop

Debugging on Chrome: chrome://discards/

App State Change Detection

Only available on some chromium browsers

script.js

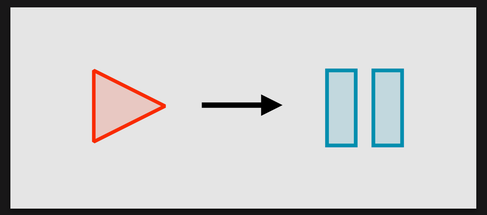


App State Change Detection

Only available on some chromium browsers

script.js

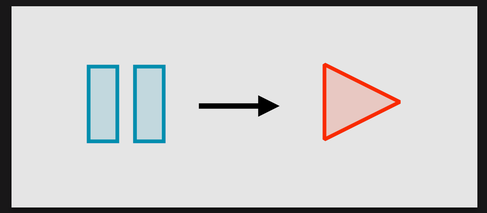
```
window.addEventListener('freeze', event => {
  // we will be suspended
  // we save current app's state (custom
code)
  saveState();
});
```

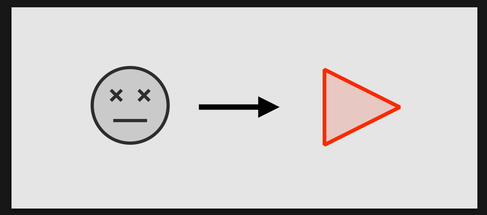


App State Change Detection

Only available on some chromium browsers

script.js



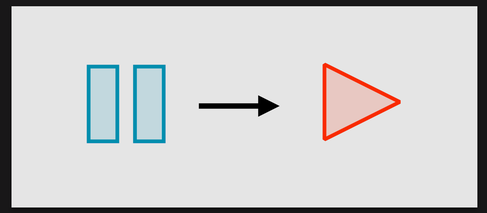


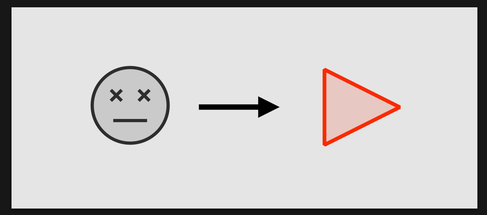
App State Change Detection

Only available on some chromium browsers

script.js

```
window.addEventListener('resume', event => {
   // we are back from suspension
   // No need to restore
});
```





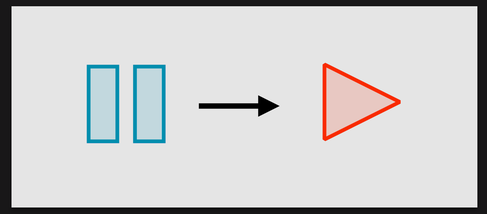
App State Change Detection

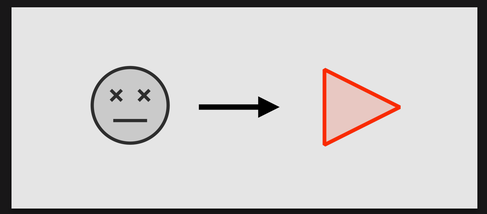
Only available on some chromium browsers

script.js

```
window.addEventListener('resume', event ⇒ {
    // we are back from suspension
    // No need to restore
});

window.addEventListener('DOMContentLoaded', event ⇒ {
  if (document.wasDiscarded) {
      // we are back from suspension but it was discarded
      restoreState();            // our custom code
  }
});
```





PAGE LIFECYCLE API ON MOBILE DEVICES

It's not typically available

You can save state

- On visibilitychange (Include a timestamp)

When should you restore state?

- When the page loads
- Using the saved timestamp you can decide to restore state or start a new navigation

Demo time

JavaScript in the Background

4

Service Workers

Service Worker

A JavaScript file running in its own thread that will act as a middleware offering a local installed web server or web proxy for your PWA, including resources and API calls

Service Worker

Runs client-side in browser's engine

HTTPS required

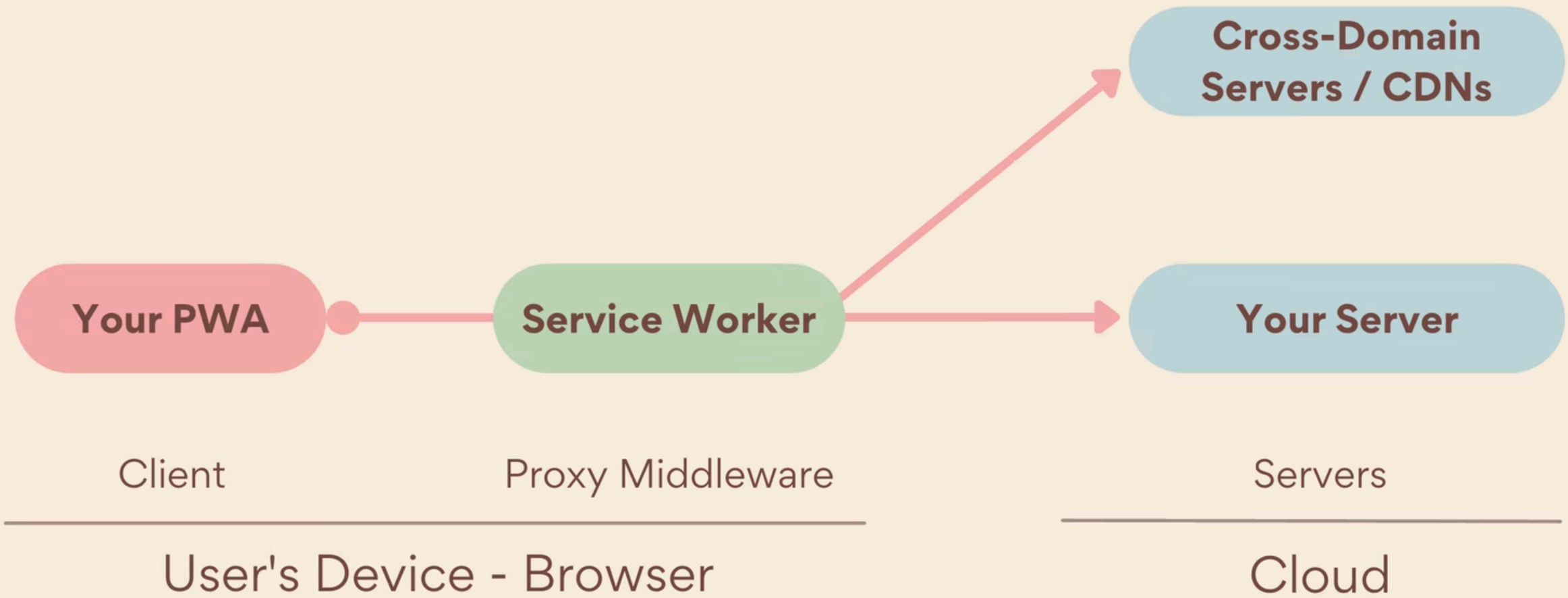
Installed by a web page

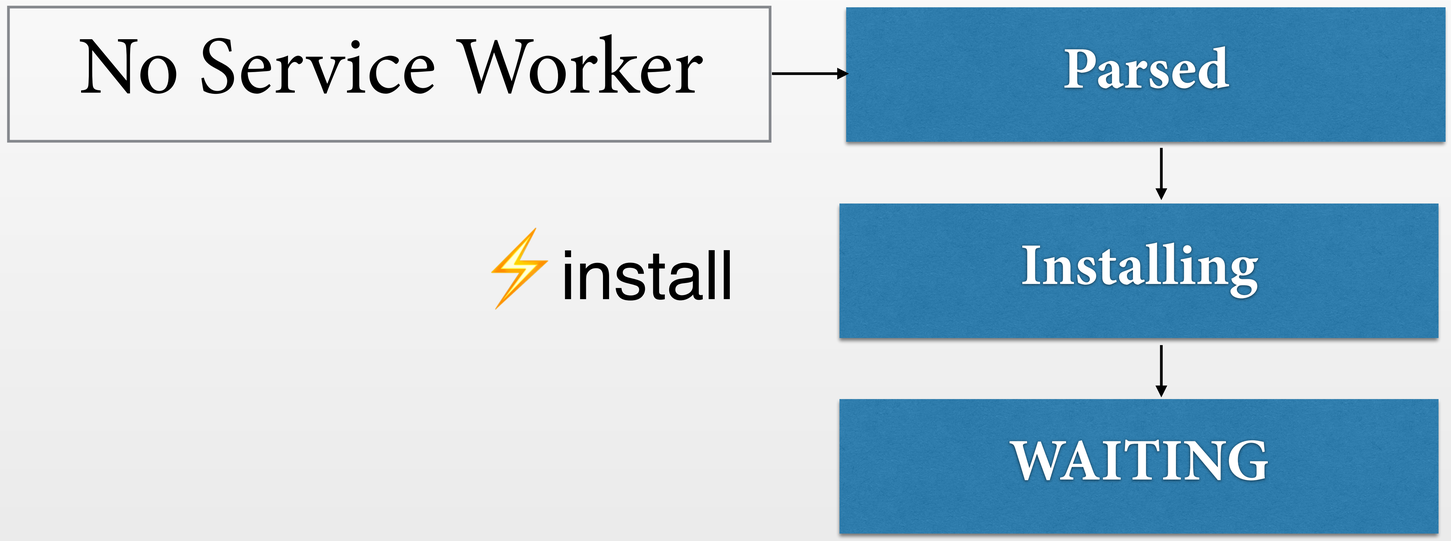
Own thread and lifecycle

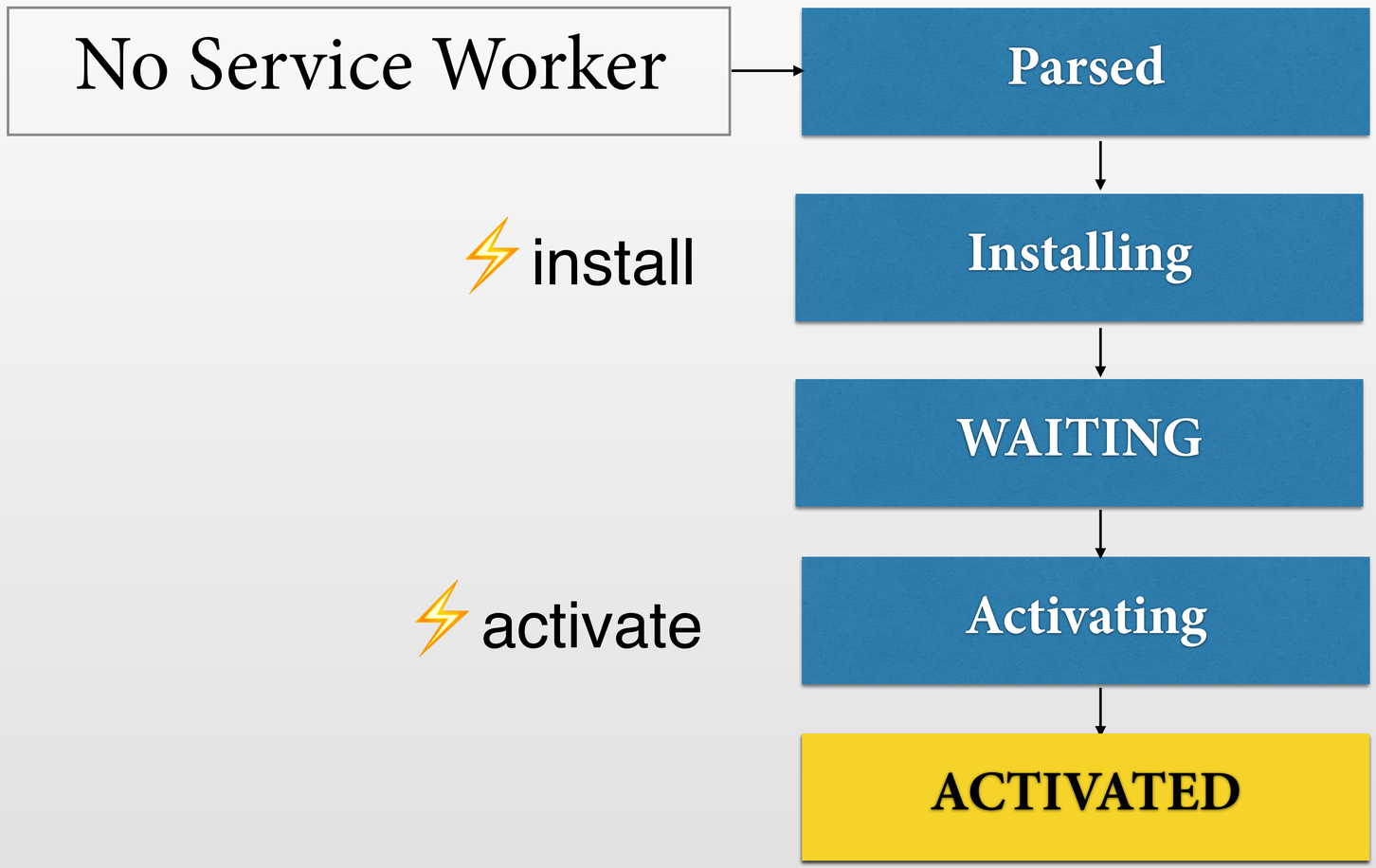
Acts as a network proxy or local web server in the name of the real server

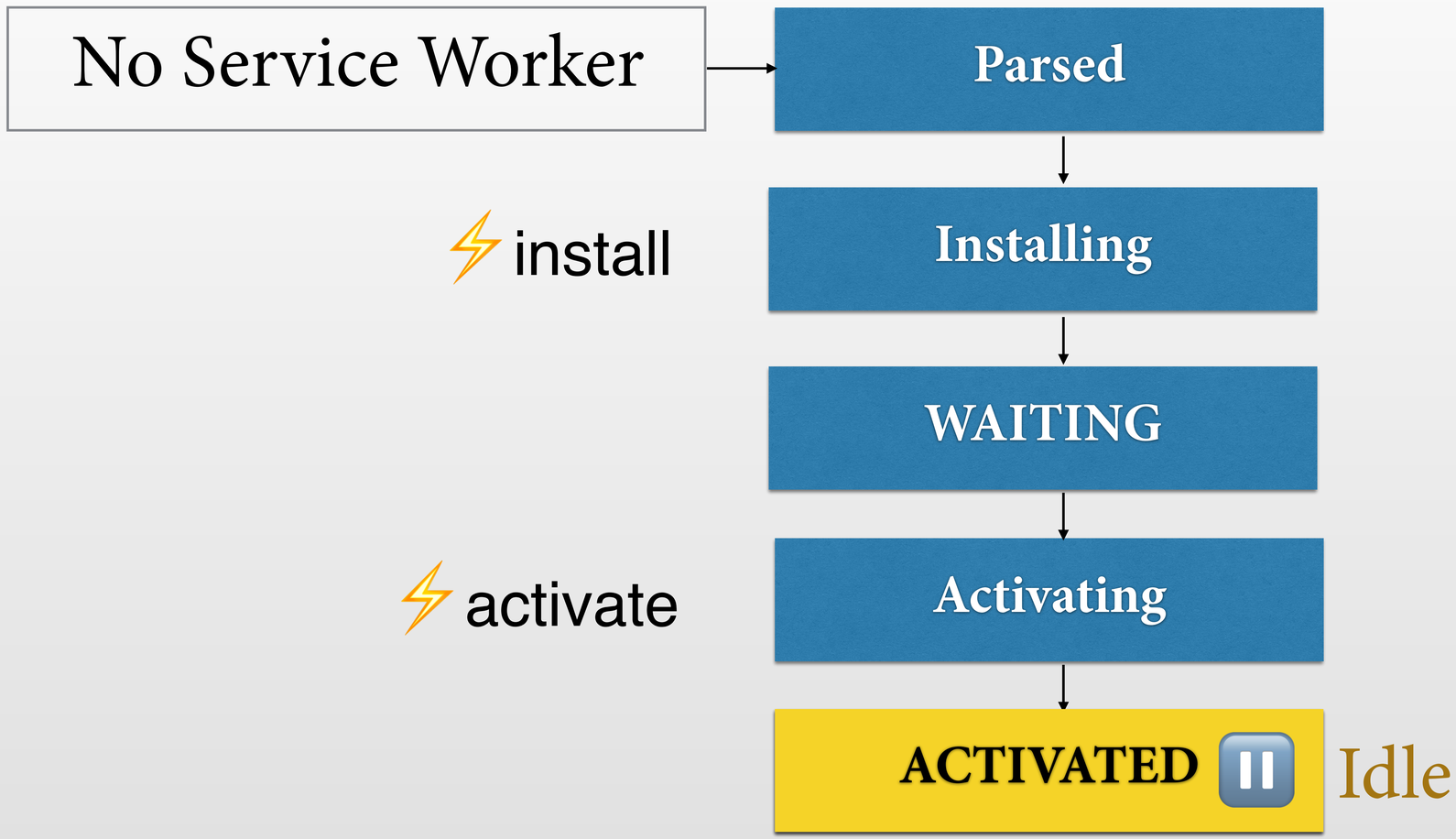
Abilities to run in the background

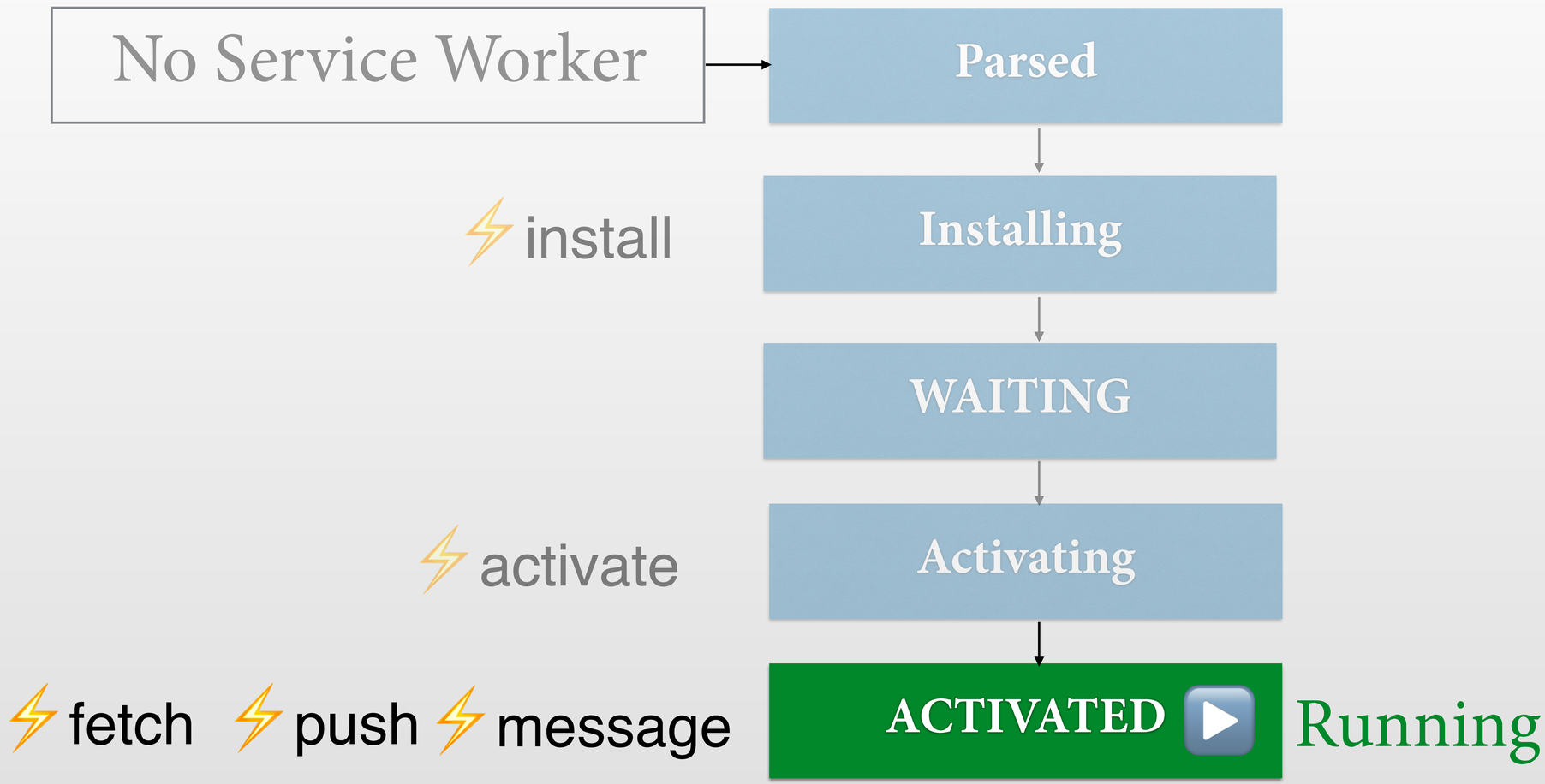
No need for user's permission

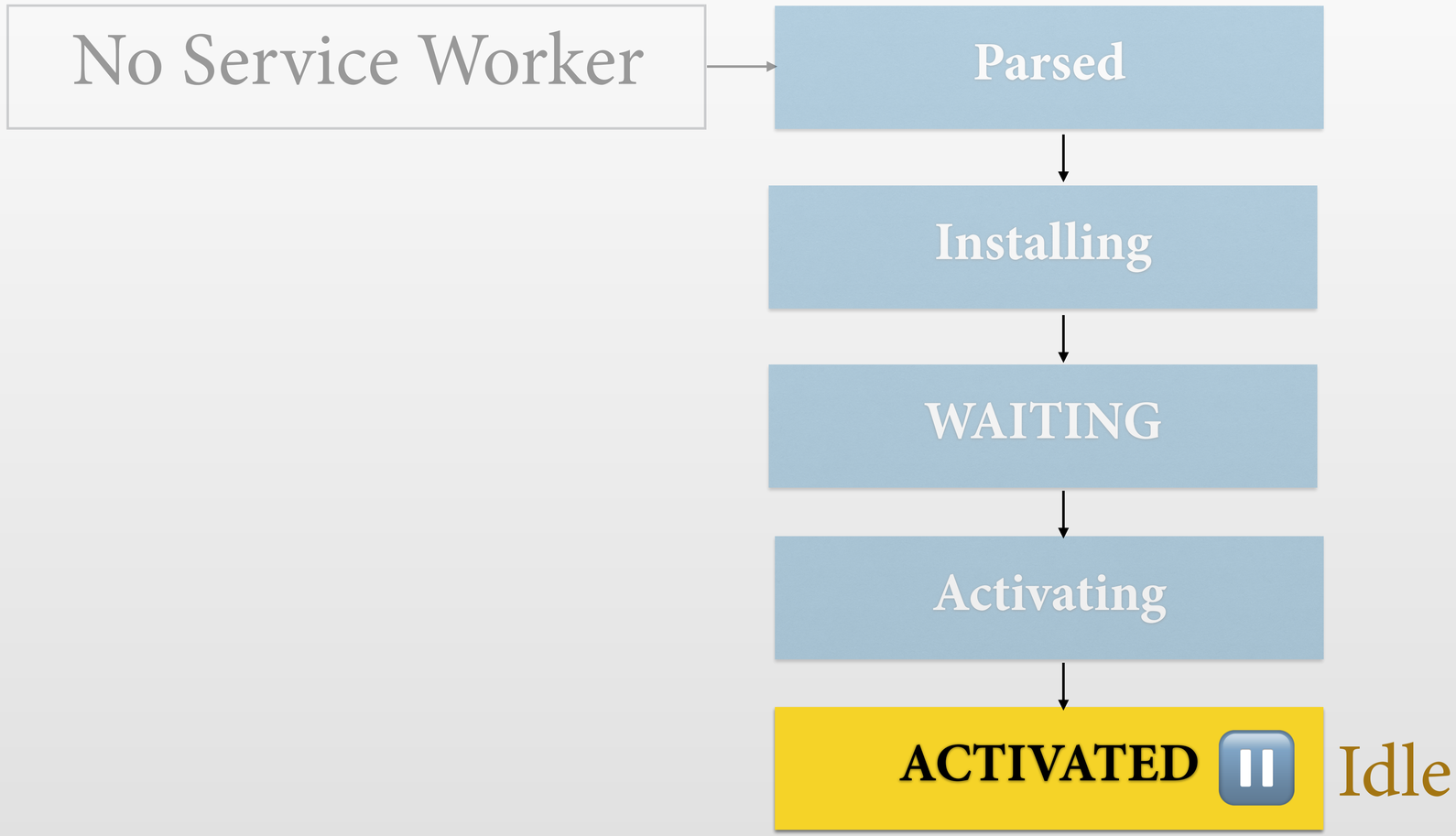

Cross-Domain Servers / CDNs
Your PWA
Service Worker
Your Server
Client
Proxy Middleware
Servers
User's Device - Browser
Cloud

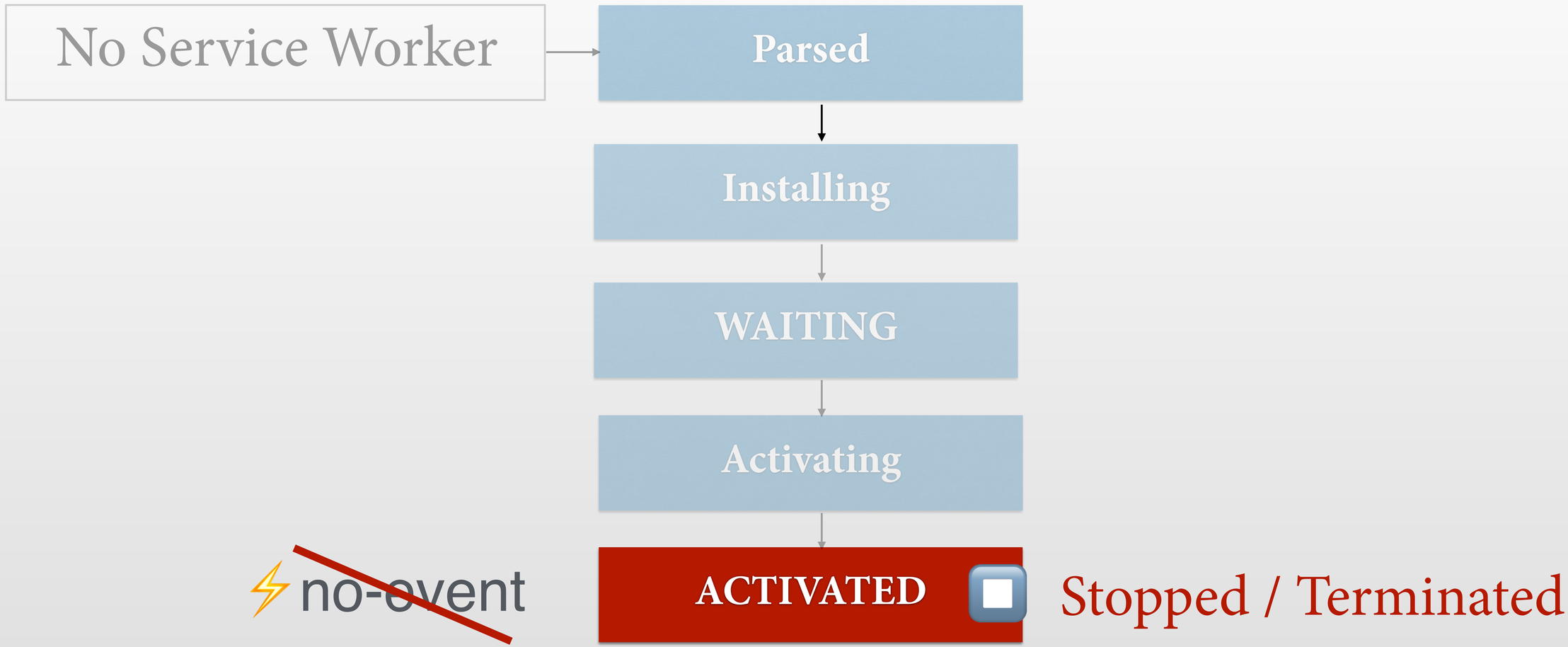

No Service Worker
Parsed
⚡install
Installing
WAITING

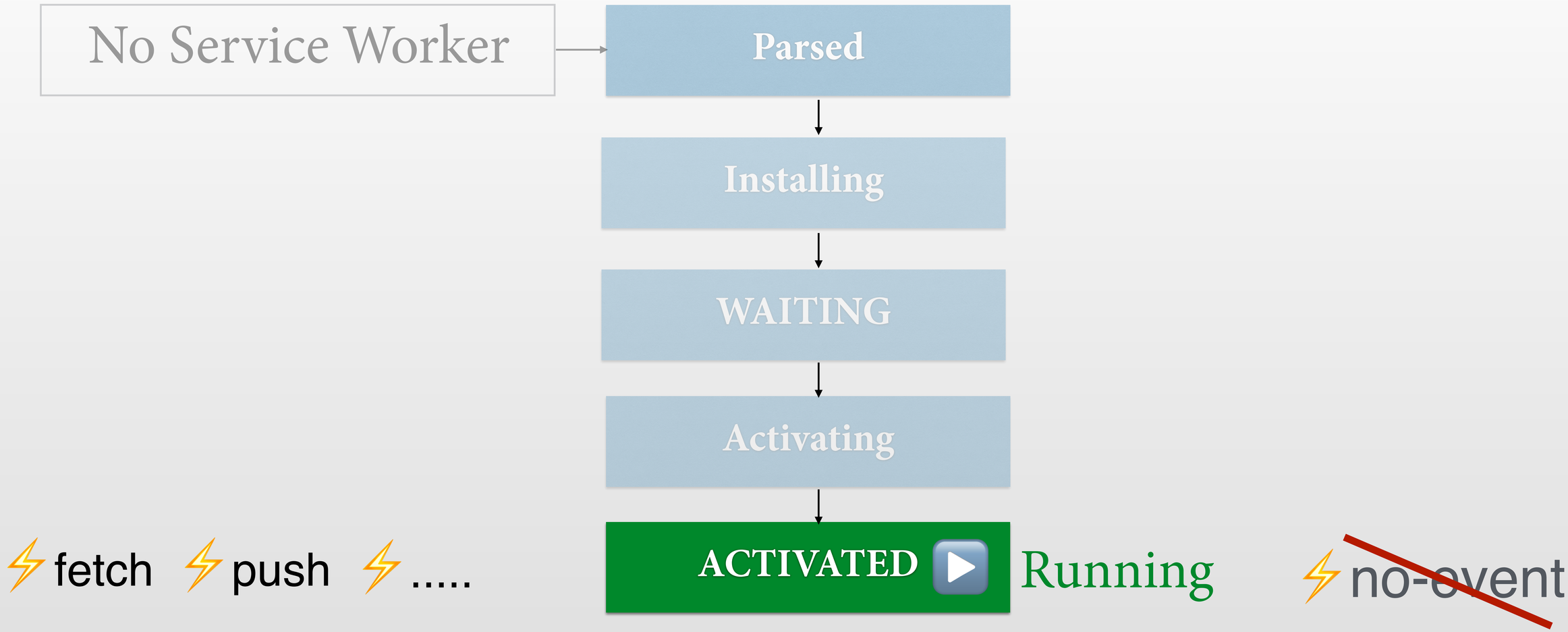

No Service Worker
Parsed
install
Installing
WAITING
activate
Activating
ACTIVATED

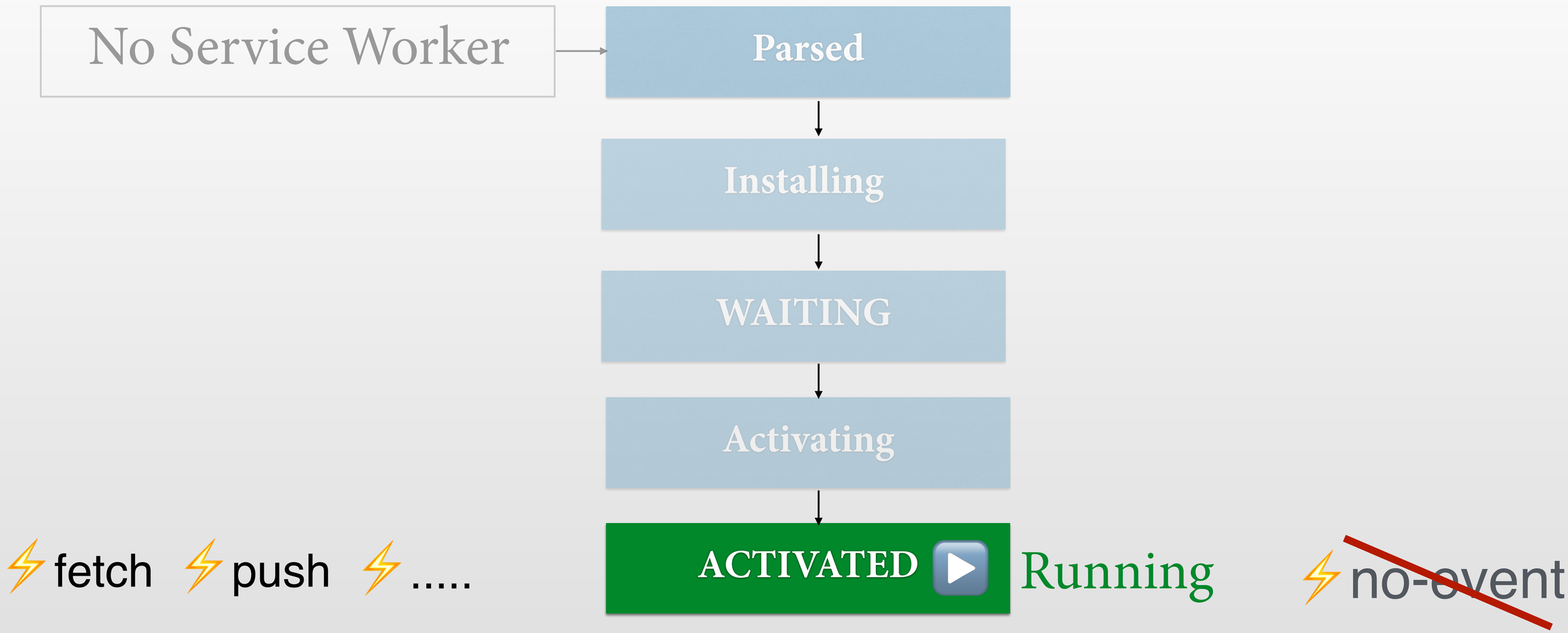

No Service Worker
Parsed
install
Installing
WAITING
activate
Activating
ACTIVATED
Idle


No Service Worker
Parsed
install
Installing
WAITING
activate
Activating
fetch
push
message
ACTIVATED
Running


No Service Worker
Parsed
Installing
WAITING
Activating
ACTIVATED
Idle


No Service Worker
Parsed
Installing
WAITING
Activating
no-event
ACTIVATED
Stopped / Terminated


No Service Worker
Parsed
Installing
WAITING
Activating
ACTIVATED
Running
fetch
push
.....
no-event



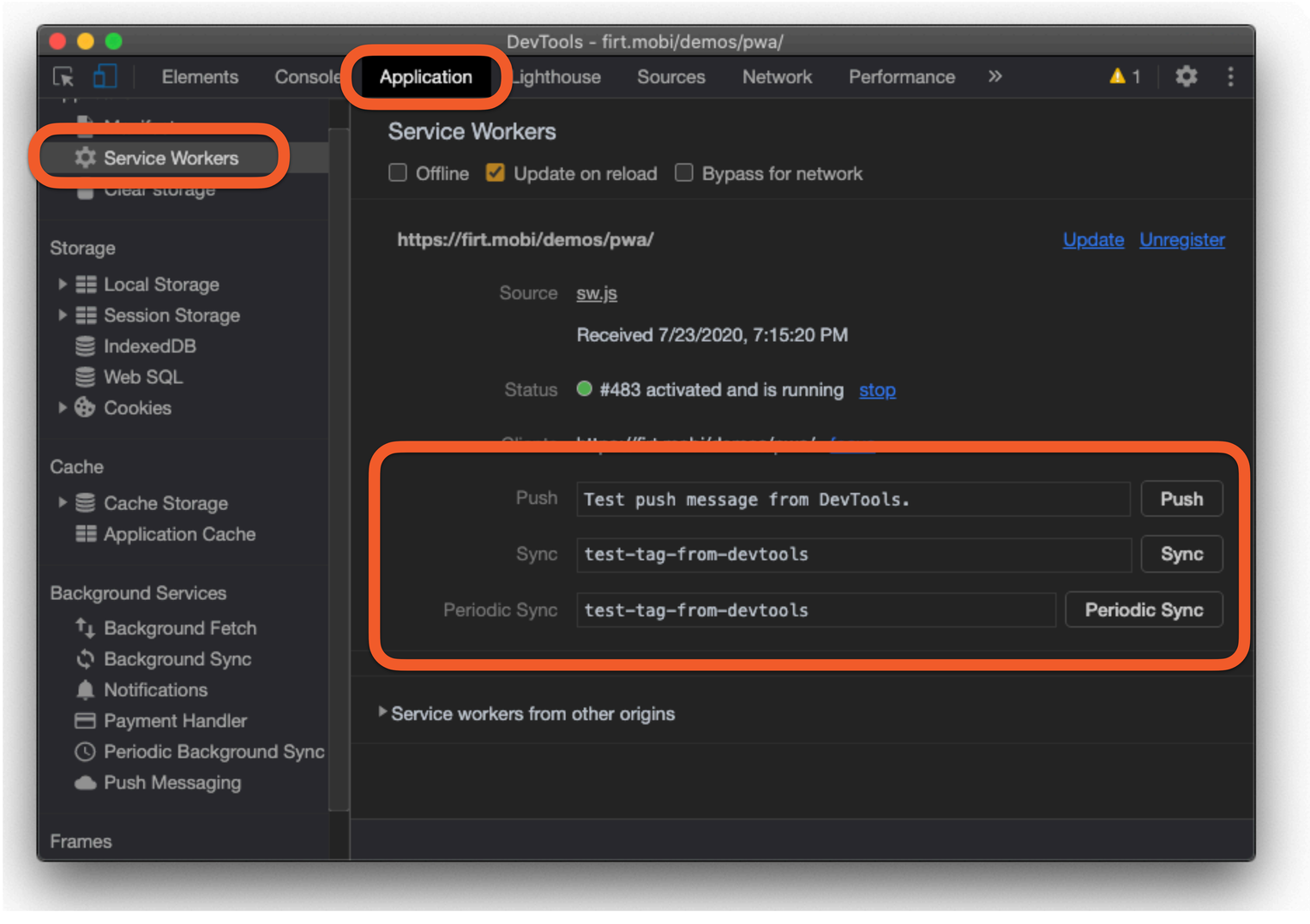
⚠️ *it might not be the same instance*

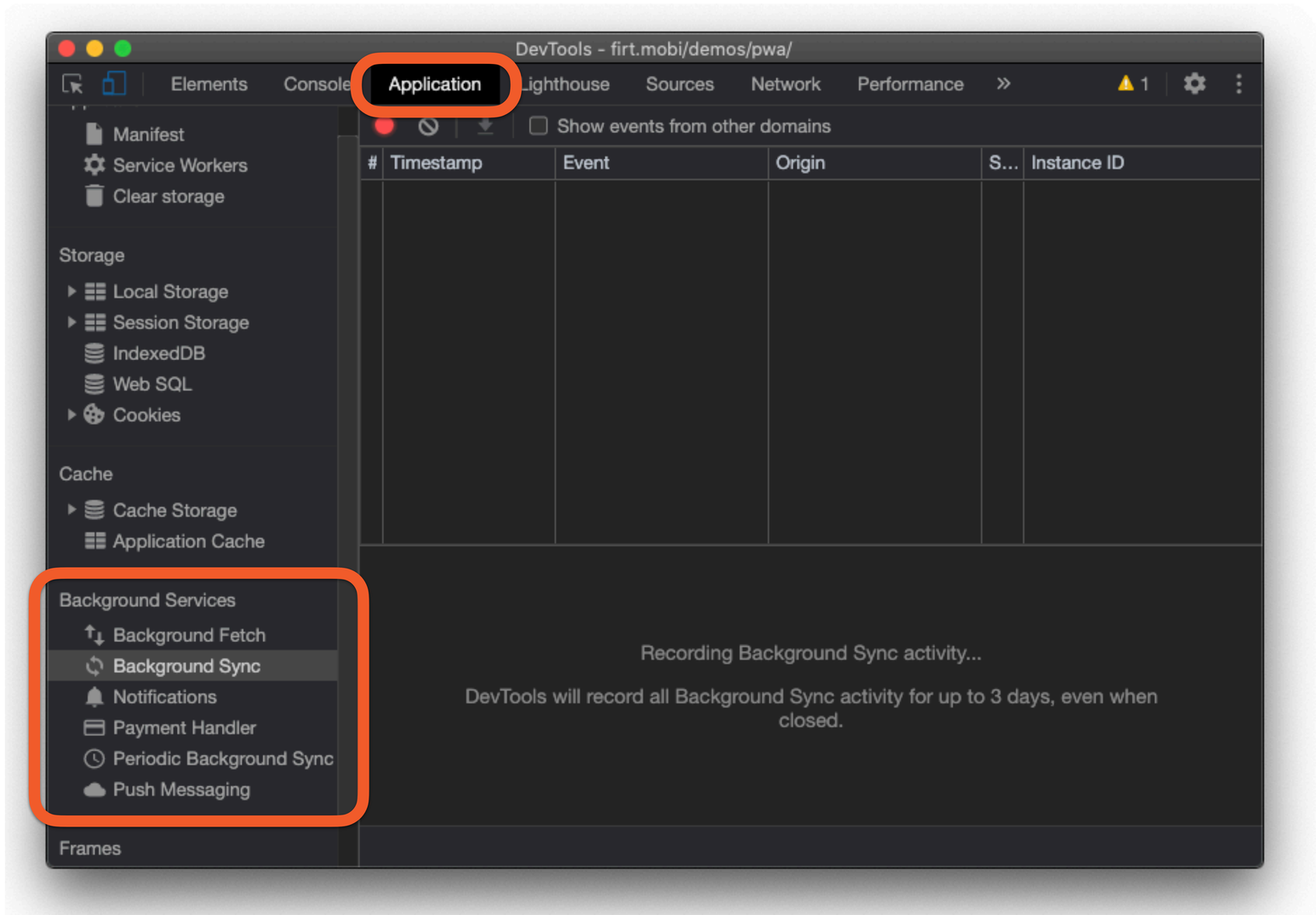
Workshop time

JavaScript in the Background

5

Debugging and Tools


DevTools - firt.mobi/demos/pwa/
Elements
Console
Application
Lighthouse
Sources
Network
Performance
Service Workers
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
Application Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
Service Workers
Offline
Update on reload
Bypass for network
https://firt.mobi/demos/pwa/
Update
Unregister
Source
sw.js
Received 7/23/2020, 7:15:20 PM
Status
#483 activated and is running
stop
Push
Test push message from DevTools.
Push
Sync
test-tag-from-devtools
Sync
Periodic Sync
test-tag-from-devtools
Periodic Sync
Service workers from other origins


DevTools - firt.mobi/demos/pwa/
Elements
Console
Application
Lighthouse
Sources
Network
Performance
1
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
Application Cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Frames
Show events from other domains
#
Timestamp
Event
Origin
S...
Instance ID
Recording Background Sync activity...
DevTools will record all Background Sync activity for up to 3 days, even when closed.

JavaScript in the Background

6

Notifications and User Interface

When we are in the background, updating the UI if possible won't notice the user

History

Historically, we could only notify the user for background tabs changing:

- <title> (changed by JS or a refresh meta)

- favicon (even animating it)

- playing audio

- force a re-focus, such as requesting a modal UI, such as an alert dialog

Well, also we had tricks to make it difficult to unload the page :(

Demo time

rpsthecoder.github.io/square-loading-favicon/

Today

Web Notifications is the only cross-platform solution.

But it requires user's permission, and it's the same permission as the one for Push messaging.

There are two APIs for creating the notification:

- Desktop Notification, for windows and Workers
- Web Push, for Service Workers

Step 1: Ask for notification permissions

Don't do this when the page loads. Follow design patterns.

script.js

Step 1: Ask for notification permissions

Don't do this when the page loads. Follow design patterns.

script.js

```
if ('Notification' in window) {
    if (Notification.permission === "granted") {

    }




}
```

Step 1: Ask for notification permissions

Don't do this when the page loads. Follow design patterns.

script.js

```
if ('Notification' in window) {
   if (Notification.permission === "granted") {

  }
   status = await Notification.requestPermission();
   // it can be'granted', 'denied', 'default'

}
```

Step 2: Create the Notification

Old Desktop Notification API; not available on Service Workers

script.js

```
const n = new Notification('Title', {
  body: 'Text',
  icon: 'image-url'
});

// we can close it later n.close();
```

Desktop Notification API

It's a local notification

Only available on Windows and Workers
(not within Service Workers)

We can update the same notification in the future

Events: show, click, error

Deprecated on Android in favor of the
Web Push notification

Media Playing

In case your web app plays audio or video, there are special cases for background playing:

- Picture in Picture API for video
- Media Session API for metadata

Media Session

Media Session

Media Session API

Provides metadata and events

script.js

```
navigator.mediaSession.metadata = new MediaMetadata({
    title: 'Best Song',
    artist: 'Max Firtman',
    album: 'Progressive Web Apps',
    artwork: [ /* … */ ]
});

// actions: play, pause, stop, previous track, nexttrack, …
navigator.mediaSession.setActionHandler(action, e ⇒ {});
```

Media Session API

Provides metadata and events for media playing

script.js

```
navigator.mediaSession.setActionHandler('play', () ⇒ { });
navigator.mediaSession.setActionHandler('pause', () ⇒ { });
navigator.mediaSession.setActionHandler('stop', () ⇒ { });
navigator.mediaSession.setActionHandler('seekbackward', () ⇒ { });
navigator.mediaSession.setActionHandler('seekforward', () ⇒ { });
navigator.mediaSession.setActionHandler('seekto', () ⇒ { });
navigator.mediaSession.setActionHandler('previoustrack', () ⇒ { });
navigator.mediaSession.setActionHandler('nexttrack', () ⇒ { });
navigator.mediaSession.setActionHandler('skipad', () ⇒ { });
```

Media Session API

It can also be used now for calls and video calls

script.js

```
navigator.mediaSession.setActionHandler('hangup', () ⇒ { });
navigator.mediaSession.setActionHandler('togglecamera', () ⇒ { });
navigator.mediaSession.setActionHandler('togglemicrophone', () ⇒ { });
```

Picture in Picture API

For video playing

script.js

```
// Toggle PiP
if (document.pictureInPictureElement) {
    document.exitPictureInPicture();
} else if (document.pictureInPictureEnabled) {
    video.requestPictureInPicture();
}
```

Picture in Picture

New events for <video>

- enterpictureinpicture
- leavepictureinpicture

You can't add HTML UI to the PiP video

You can customize the <video> using a CSS pseudo class :picture-in-picture

You can toggle PiP in/out with JS

Controls are defined through Media Session

Demo time

pip-rendering-fun.netlify.app

Workshop time

7

Execute in the Background

Why do we want to execute code in the background?

Network requests

If your web app goes to the background while a network request is in process, it may be aborted.

Solutions:

- Beacon API, if suitable
- Web Background Synchronization API
- Background Fetch API

Beacon API

For requests where we don't care about its response

script.js

```
document.addEventListener('visibilitychange', () => {
  if (document.visibilityState === 'hidden') {
    navigator.sendBeacon('/log-hidden', someData);
  }
});
```

Beacon API

Data goes in text by default; if you want to force json, use Blob

script.js

```
document.addEventListener('visibilitychange', () ⇒ {
  if (document.visibilityState ═══ 'hidden') {
    const blob = new Blob([JSON.stringify(data)],
         {type : "application/json"});
    navigator.sendBeacon('/log-hidden', blob);
  }
});
```

You can wake up your code client-side while in the background and/or continue executing code after leaving the web app using:

Background Sync

Periodic Background Sync

Background Fetch

Remember execution happens in the background; maybe no PWA's page is currently loaded.

Use Web Notifications for messaging the user

Background Sync

PWA defers a sync action until the device has a stable connection to the server

A 'sync' event will be fired in the Service Worker to handle a pending sync:

- Immediately if network is stable
- Later, when network go back to stable
- If battery is in good level

We access the network and fulfill the sync or leave it pending

5% of mobile web apps with a Service Worker are already using the API

Step 1: Register a Background Sync Action

Available from the window or service worker's contexts

script.js

Step 1: Register a Background Sync Action

Available from the window or service worker's contexts

script.js

```
if ('SyncManager' in window) {
  const registration = await navigator.serviceWorker.ready;
  registration.sync.register("tag-name");
}
```

Step 2: Handle the Sync Action

This happens in the service worker's context at any moment

serviceworker.js

Step 2: Handle the Sync Action

This happens in the service worker's context at any moment

serviceworker.js

```
self.addEventListener('sync', event => {
  if (event.tag == 'tag-name') {
    event.waitUntil(syncOperation());
  }
});
```

Periodic Background Sync

PWA asks user for permission to periodically execute code in the background

A 'periodicsync' event will be fired in the Service Worker:

- On a synchronization time interval
- If battery and network conditions are met

We typically access the network on each execution, but it's not mandatory

Right now, it's fired with a maximum of once every 12 hours

Execution frequency will be honored based on a Site Engagement Score defined by the browser

Step 1: Ask for Periodic Sync Permission

Permission will be granted based on Site Engagement Score

script.js

Step 1: Ask for Periodic Sync Permission

Permission will be granted based on Site Engagement Score

script.js

```
const permissionStatus = await navigator.permissions.query({
  name: 'periodic-background-sync',
});
```

Step 1: Ask for Periodic Sync Permission

Permission will be granted based on Site Engagement Score

script.js

```
const permissionStatus = await navigator.permissions.query({
  name: 'periodic-background-sync',
});
if (permissionStatus.state === 'granted') {
  track('periodic-sync', 'granted');
} else {
  track('periodic-sync', 'denied');
}
```

Step 2: Register a Background Sync Operation

Permission will be granted based on Site Engagement Score

script.js

Step 2: Register a Background Sync Operation

It requires a tag and a synchronization interval in milliseconds

script.js

```
const registration = await navigator.serviceWorker.ready;

    await registration.periodicSync.register('sync-tag', {
      minInterval: 24 * 60 * 60 * 1000 // One day
    });

}
```

Step 2: Register a Background Sync Operation

It requires a tag and a synchronization interval in milliseconds

script.js

```
const registration = await navigator.serviceWorker.ready;
if ('periodicSync' in registration) {
  try {
    await registration.periodicSync.register('sync-tag', {
      minInterval: 24 * 60 * 60 * 1000 // One day
    });
  } catch (error) { }
}
```

Step 3: Handle the Periodic Sync Operation

It will be executed in the Service Worker

serviceworker.js

Step 3: Handle the Periodic Sync Operation

It will be executed in the Service Worker

serviceworker.js

```
self.addEventListener('periodicsync', (event) ⇒ {
  if (event.tag === 'sync-tag') {
    event.waitUntil(doPeriodicSyncOperation());
  }
});
```

Background Fetch

PWA asks the web engine to make some fetch download requests

The browser will download the requests in the background while showing an OS notification about the process

Events will be fired in the Service Worker when:

- Download finishes
- If the user has clicked the notification
- On abort or failure
- On Progress

Step 1: Start a Background Fetch Request

The operation will be handled entirely by the browser

script.js

Step 1: Start a Background Fetch Request

The operation will be handled entirely by the browser

script.js

```
const registration = await navigator.serviceWorker.ready;

  const fetch = await registration.backgroundFetch.fetch(




  );
```

Step 1: Start a Background Fetch Request

script.js

```
const registration = await navigator.serviceWorker.ready;

  const fetch = await registration.backgroundFetch.fetch(
   'fetch-name', ['url-1', 'url-2', 'url-3'],




  );
```

Step 1: Start a Background Fetch Request

script.js

```
const registration = await navigator.serviceWorker.ready;

  const fetch = await registration.backgroundFetch.fetch(
   'fetch-name', ['url-1', 'url-2', 'url-3'],
   {
     title: 'Offline Content',
     icons: [{ sizes:'300×300', src:'icon.png', type:'image/png'}],
     downloadTotal: 20 * 1024 * 1024,
   }
  );
```

Step 1: Start a Background Fetch Request

script.js

```
const registration = await navigator.serviceWorker.ready;
if ('backgroundFetch' in registration) {
  const fetch = await registration.backgroundFetch.fetch(
    'fetch-name', ['url-1', 'url-2', 'url-3'],
    {
      title: 'Offline Content',
      icons: [{ sizes:'300×300', src:'icon.png', type:'image/png'}],
      downloadTotal: 20 * 1024 * 1024,
    }
  );
}
```

Step 2: Handle events in Service Worker

serviceworker.js

Step 2: Handle events in Service Worker

serviceworker.js

```
self.addEventListener('backgroundfetchsuccess', async (event) => {
  const downloadedFiles = await event.registration.matchAll();
});
```

Step 2: Handle events in Service Worker

serviceworker.js

```
self.addEventListener('backgroundfetchsuccess', async event => {
  const downloadedFiles = await event.registration.matchAll();
});

self.addEventListener('backgroundfetchclick', event => {
  clients.openWindow('/download-status');
});
```

Step 2: Handle events in Service Worker

serviceworker.js

```
self.addEventListener('backgroundfetchsuccess', async event => {
  const downloadedFiles = await event.registration.matchAll();
});

self.addEventListener('backgroundfetchclick', event => {
  clients.openWindow('/download-status');
});

self.addEventListener('backgroundfetchfailure', event => {
});
```

JavaScript in the Background

8

Push Notifications

Web Push Notifications

PWA asks permission to send notifications to the user

If granted, now the service worker can create a notification from the background

Also, the PWA can subscribe the user to Push

A 'push' event will be fired in the Service Worker to handle a push

Firefox, Chrome, Edge, Samsung Internet

Safari from macOS 13 (Ventura), iOS and iPadOS from some version from 2023

Step 1: Ask for notification permissions

Don't do this when the page loads. Follow design patterns.

script.js

Step 1: Ask for notification permissions

Don't do this when the page loads. Follow design patterns.

script.js

```
if ('Notification' in window) {
    if (Notification.permission === "granted") {
        track('notification', "granted");
    }




}
```

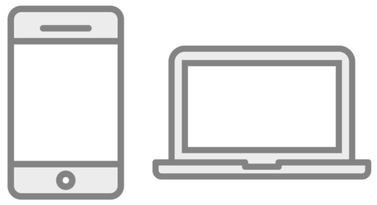
Step 1: Ask for notification permissions

Don't do this when the page loads. Follow design patterns.

script.js

```
if ('Notification' in window) {
    if (Notification.permission === "granted") {
        track('notification', "granted");
    }
    status = await Notification.requestPermission();
    // it can be'granted', 'denied', 'default'
    track('notification-request', status);
}
```

Web Push Subscription Architecture

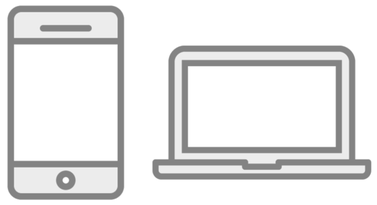


Web server

Push server

owned by browser

Web Push Subscription Architecture



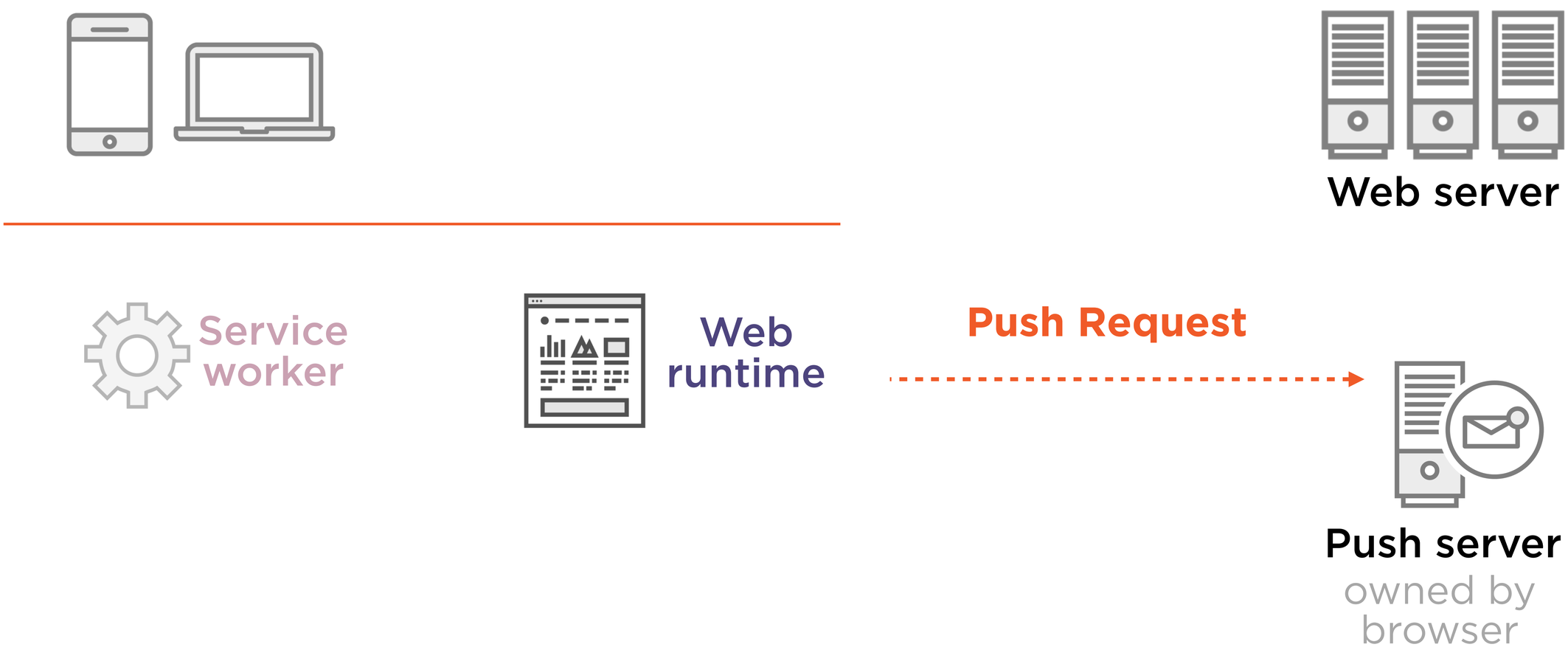
Web server

Web
runtime

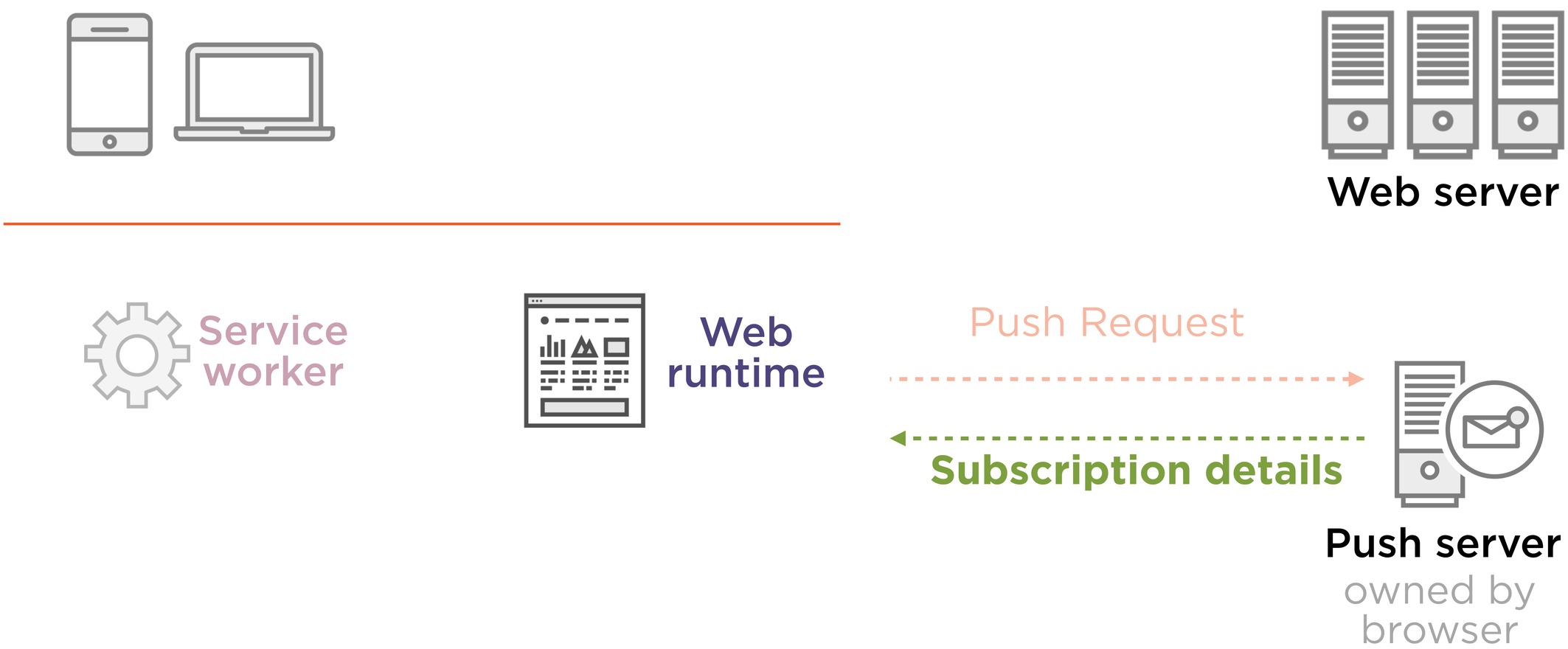
Push server

owned by
browser

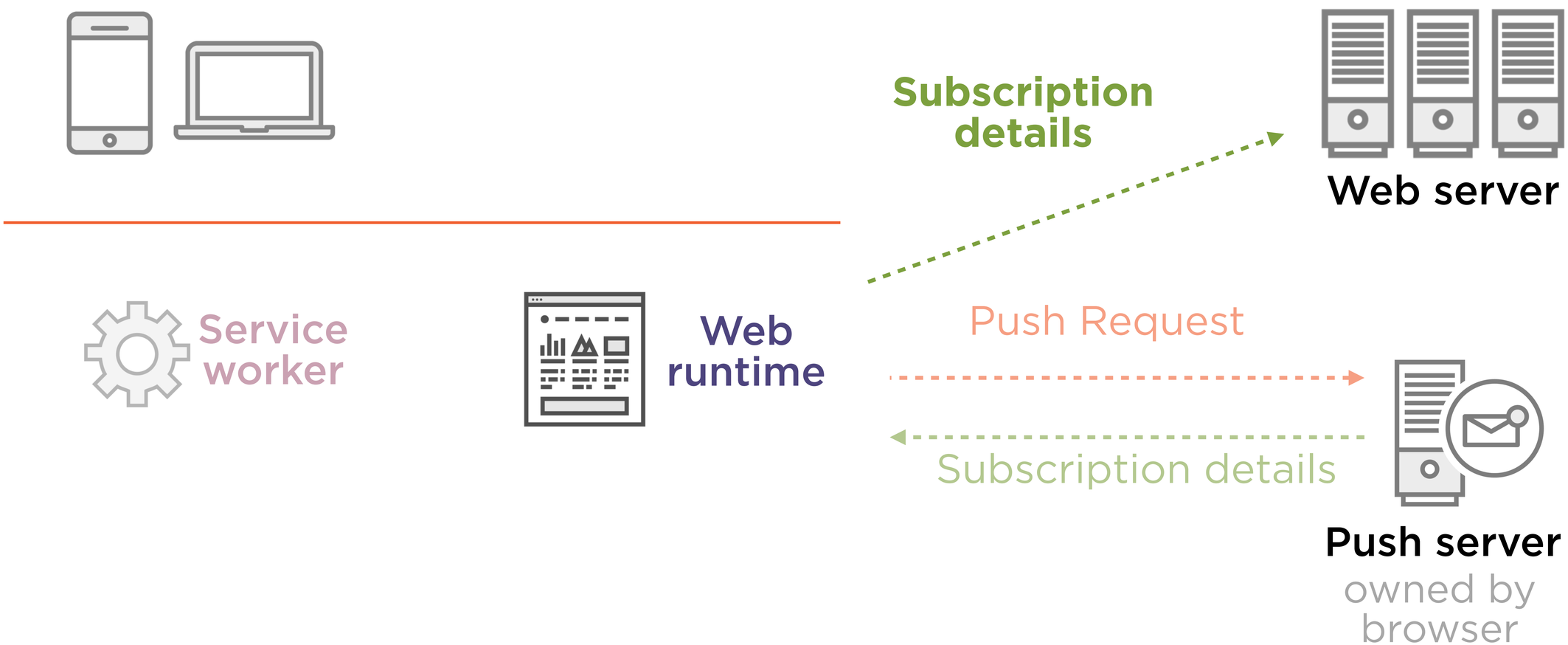
Web Push Subscription Architecture



Web Push Subscription Architecture



Web Push Subscription Architecture



Step 2: Register a Push Subscription

script.js

Step 2: Register a Push Subscription

Check course's source code for instructions for get a key

script.js

```
if ('PushManager' in window) {
  const registration = await navigator.serviceWorker.ready;

}
```

Step 2: Register a Push Subscription

Check course's source code for instructions for get a key

script.js

```
if ('PushManager' in window) {
  const registration = await navigator.serviceWorker.ready;
  const pushData = await registration.pushManager
        .subscribe(
            { userVisibleOnly: true, applicationServerKey: 'KEY' }
        );




}
```

Step 2: Register a Push Subscription

Check course's source code for instructions for get a key

script.js

```
if ('PushManager' in window) {
  const registration = await navigator.serviceWorker.ready;
  const pushData = await registration.pushManager
        .subscribe(
            { userVisibleOnly: true, applicationServerKey: 'KEY' }
        );
  if (pushData) {
     // We have the Push Subscription details to save in our server
  }
}
```

```
{
  "endpoint": “https://pushserver.com/unique-id",
  "keys": {
    "p256dh" :
       "BNcRdreALRFXTkOOUHK1EtK2wtaz5Ry4YfYCA_0QTpQtUbVlUls0VJXg7A8u-
        Ts1XbjhazAkj7I99e8QcYP7DkM=",
    "auth"   : "tBHItJI5svbpez7KI4CCXg==”
  }
}
```

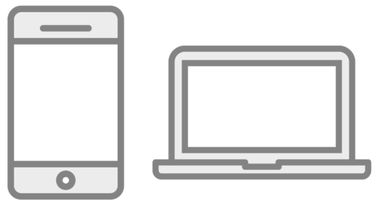
Subscription details from the Push server

It includes an end-point, a unique ID for that user in that browser and public keys we
have to use later to encrypt messages to that user

We need to store this data safely in our web server and assign it to the user

If you want to send a personalized message to each user you need to assign these details to the right user

Web Push Message Architecture



Web server

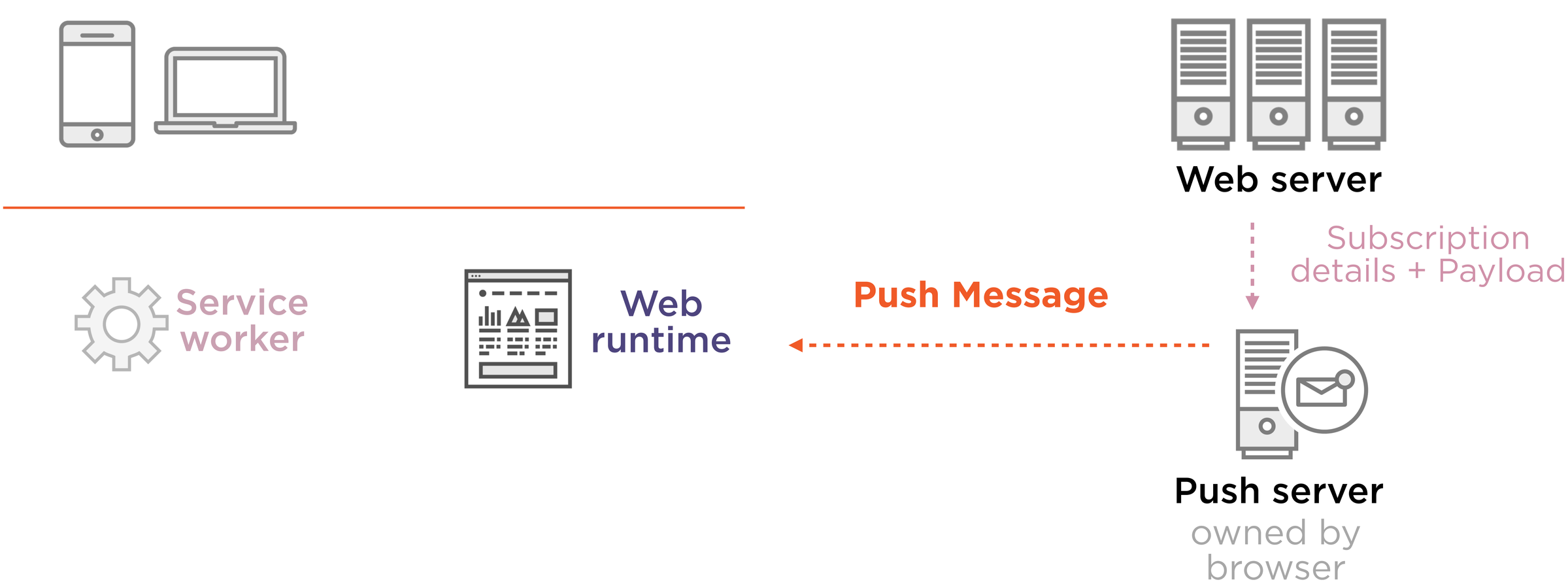
Web
runtime

Push server

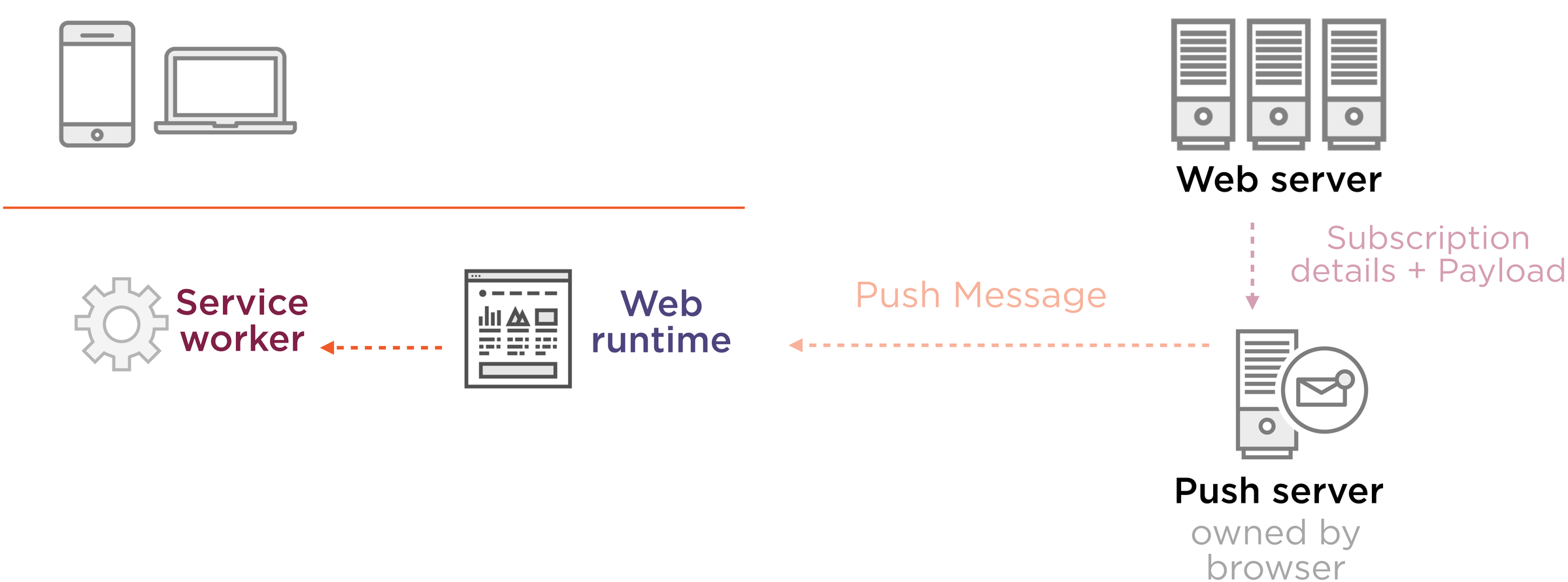
owned by
browser

Web Push Message Architecture

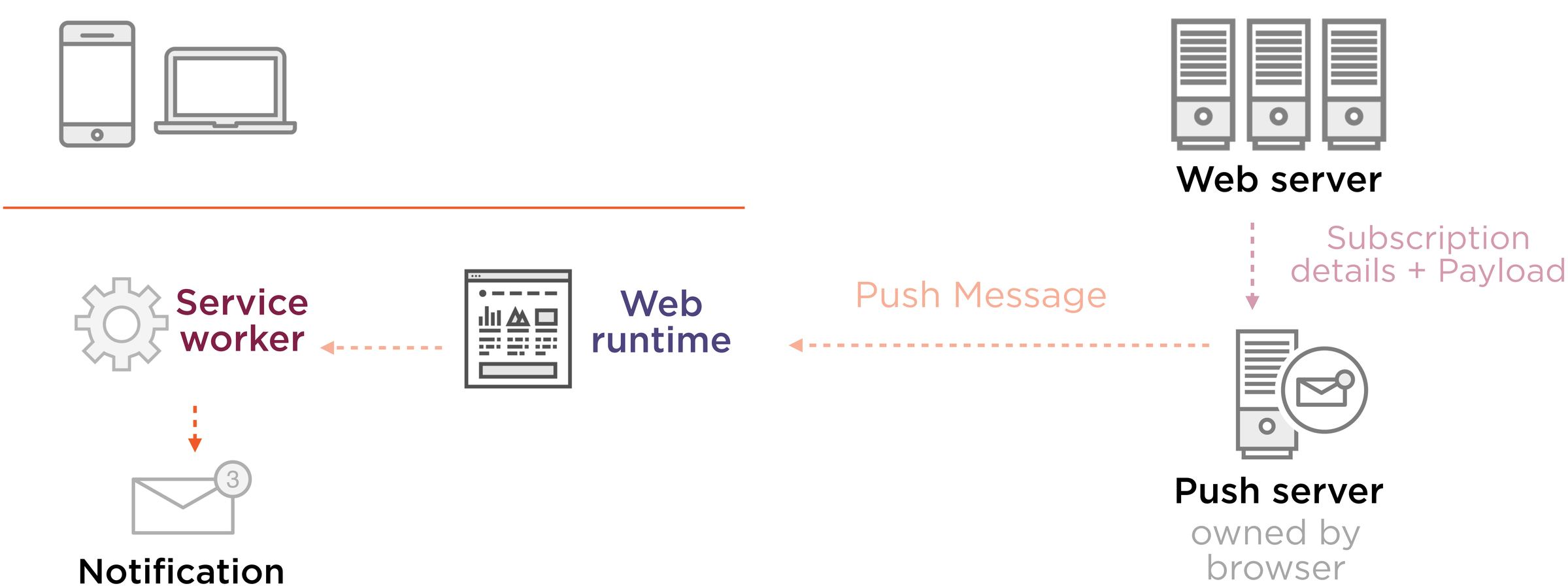
Web Push Message Architecture



Web Push Message Architecture



Web Push Message Architecture



Step 3: Receive the push and create notification

serviceworker.js

Step 3: Receive the push and create notification

serviceworker.js

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

Step 3: Receive the push and create notification

serviceworker.js

```
self.addEventListener('push', event => {
  event.waitUntil(
    self.registration.showNotification('Title',
     {
       body: event.data.text(),
        icon: "/icon.png"
     }
    )
  );
});
```

Push messages can arrive when a web app is active or when it's closed

You are not forced to notify the user with the payload's body, but you are forced to notify something

Workshop time

What we've covered today

- Web App Lifecycle
- Background's Life
- Detection
- Notifications
- Page Visibility API
- Service Workers
- Execution in the Background
- Web Push

hi@firt.dev

@firt

Fot