Building Beautiful Web Apps

User experience and visual design best practices for PWAs

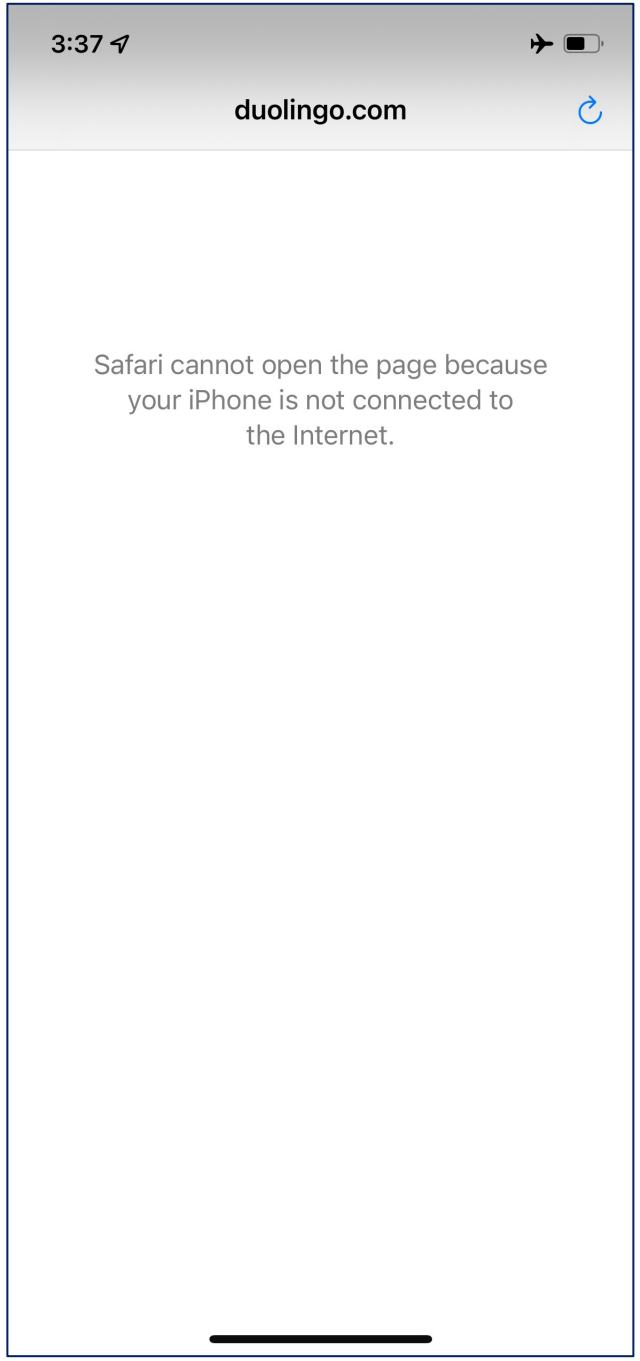
Stephanie Stimac @seaotta
Product Manager Microsoft Edge

All Day Hey! 2022

PWAs

Progressive Web Apps

whomp
whomp

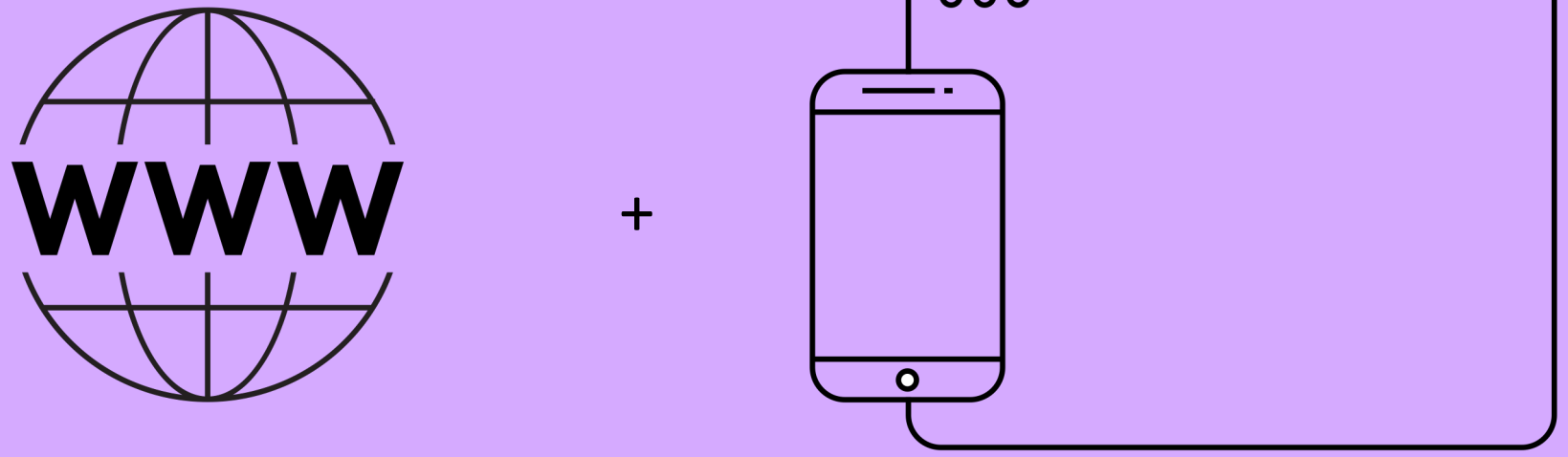

3:37
duolingo.com
Safari cannot open the page because your iPhone is not connected to the Internet.

PWAs

PWAs are progressively enhanced websites that function like installed native apps on supported platforms and function like regular websites on other browsers.

- Microsoft Docs

PWA



Why?

Why should I care about PWAs?

- Cheaper to produce and maintain
- Write one codebase, use across multiple platforms
- More distribution options (app stores)
- Able to control updates more easily
- Offline mode and faster

Addy Osmani

Dec 23, 2017 · 9 min read · Listen

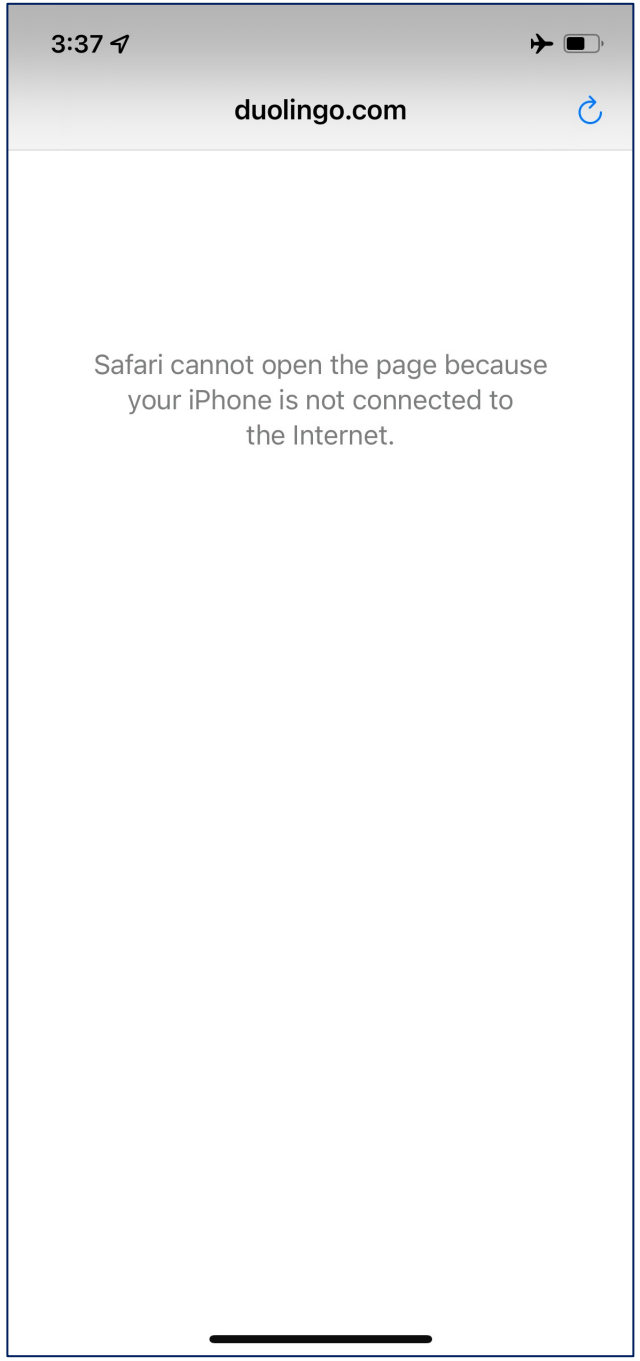
A Tinder Progressive Web App Performance Case Study

Tinder recently swiped right on the web. Their new responsive Progressive Web App — Tinder Online — is available to 100% of users on desktop and mobile, employing techniques for JavaScript performance optimization, Service Workers for network resilience and Push Notifications for chat engagement. Today we'll walk through some of their web perf learnings.

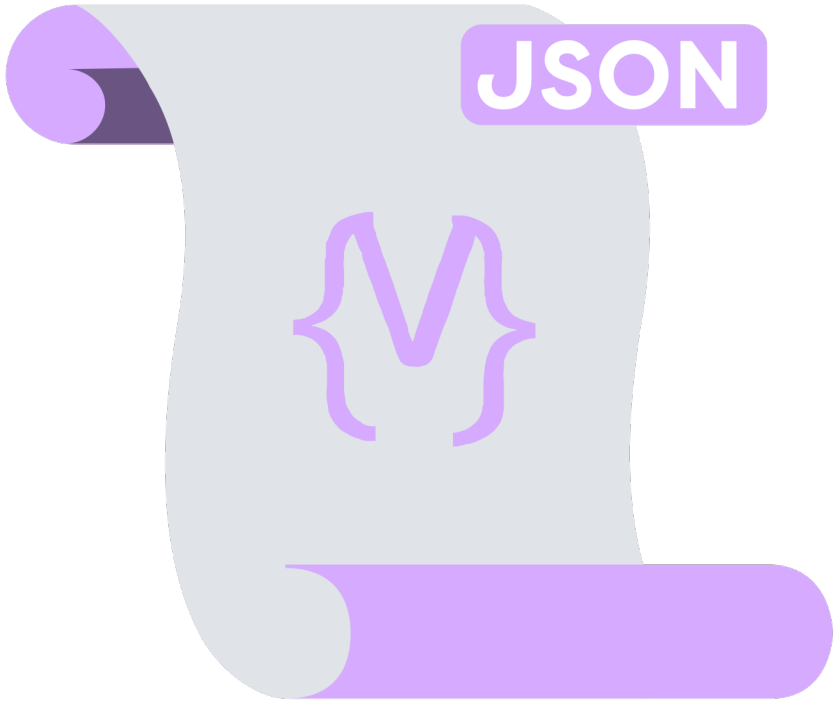
https://aka.ms/pwa-tinder-case-study

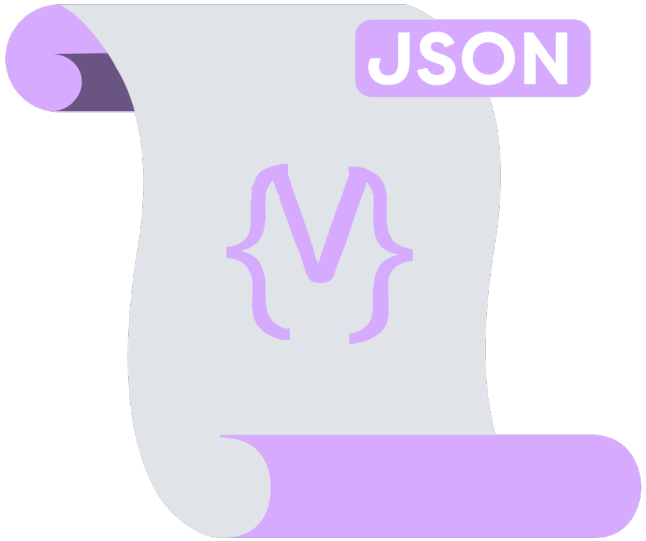
User Experience and Visual Design Best Practices for PWAs

whomp
whomp


3:37
duolingo.com
Safari cannot open the page because your iPhone is not connected to the Internet.

The Web Application Manifest


JSON

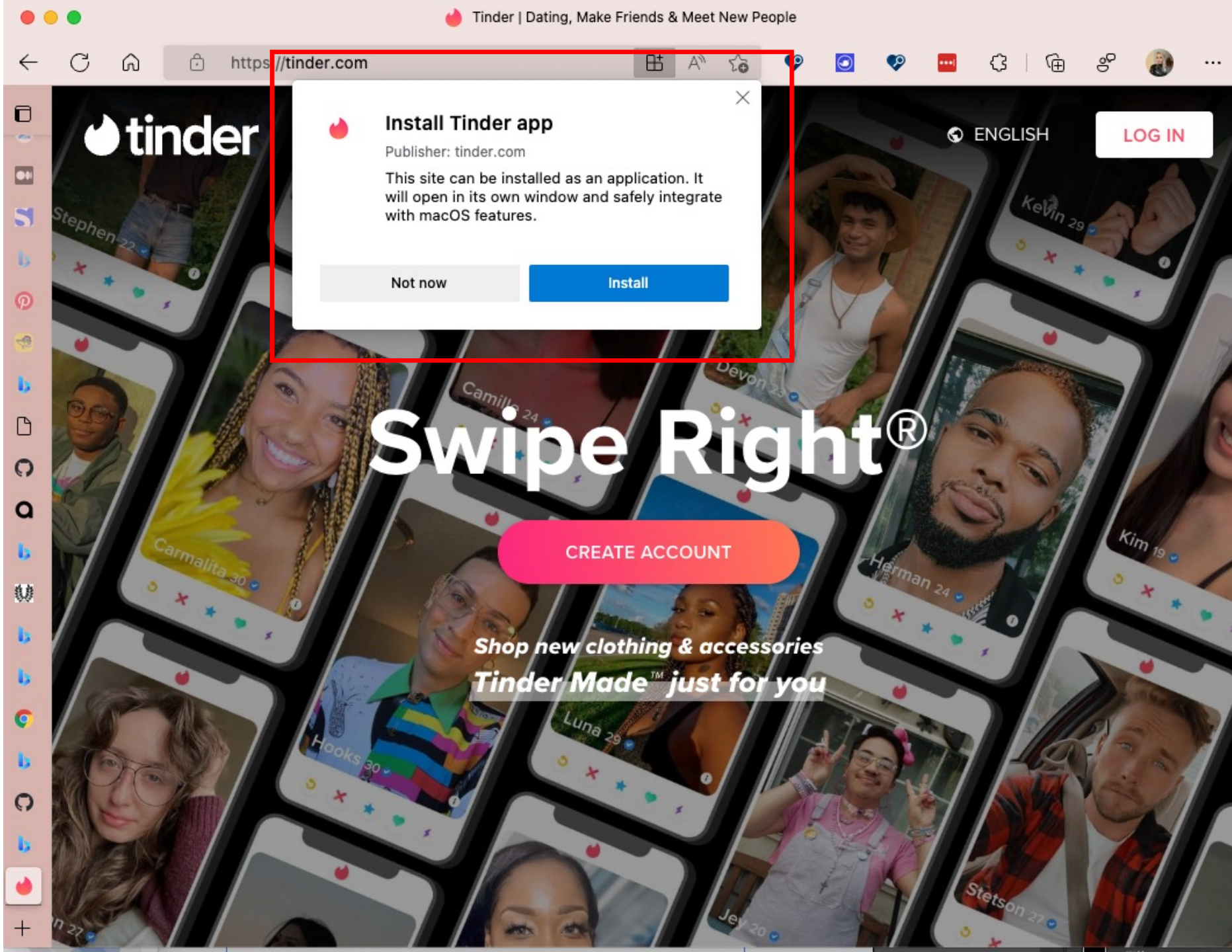

JSON


iOS

01 UI Basics for PWAs

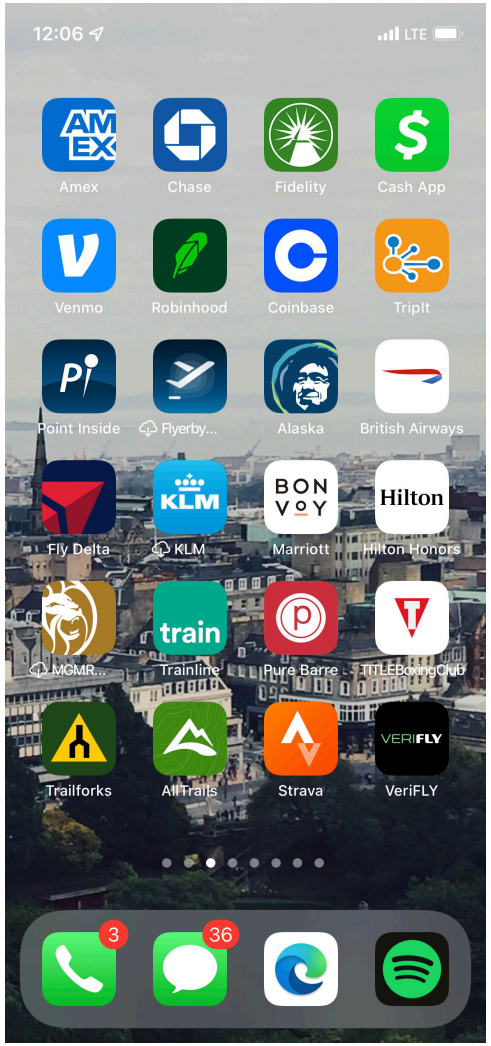
Setting app icons

01 UI Basics for PWAs



Taskbar
Start menu
Desktop icon (Mac, Windows, Chrome OS)
System settings
Android home screen
Add-to-home-screen prompt
Android splash screen
iOS home screen
iOS splash screen

01 UI Basics for PWAs / Icons



Different strokes for different folks

Transparent PWA icons appear inside white circles on Android

Source: https://web.dev/maskable-icon/

Icon purpose

Manifest member

```
{
  …
    "icons": [
      …
        {
          "src": "path/to/regular_icon.png",
          "sizes": "196x196",
          "type": "image/png",
          "purpose": "any"
        },
      …
      ],
```

“purpose”: “any”

Transparent PWA icons appear inside white circles on Android

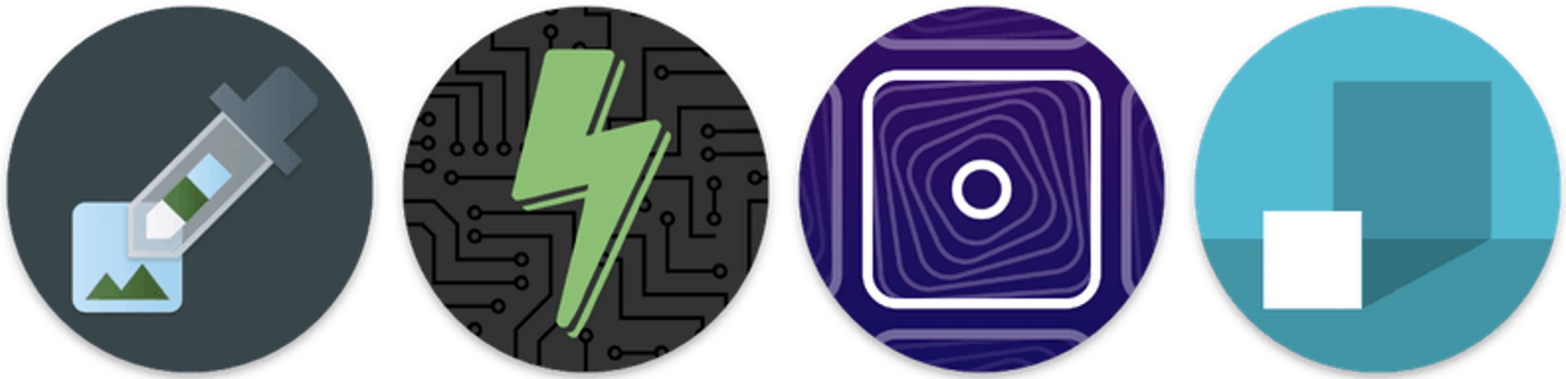
Source: https://web.dev/maskable-icon/

Maskable icons

Manifest member

```
{
  …
    "icons": [
      …
        {
          "src": "path/to/maskable_icon.png",
          "sizes": "196x196",
          "type": "image/png",
          "purpose": "maskable"
        },
      …
     ],
```

“purpose”: “maskable”



Maskable icons cover the entire circle instead

Source: https://web.dev/maskable-icon/

https://web.dev/maskable-icon/

Summary: PWA icons

Required for your web app to be installable and receive a prompt from the browser

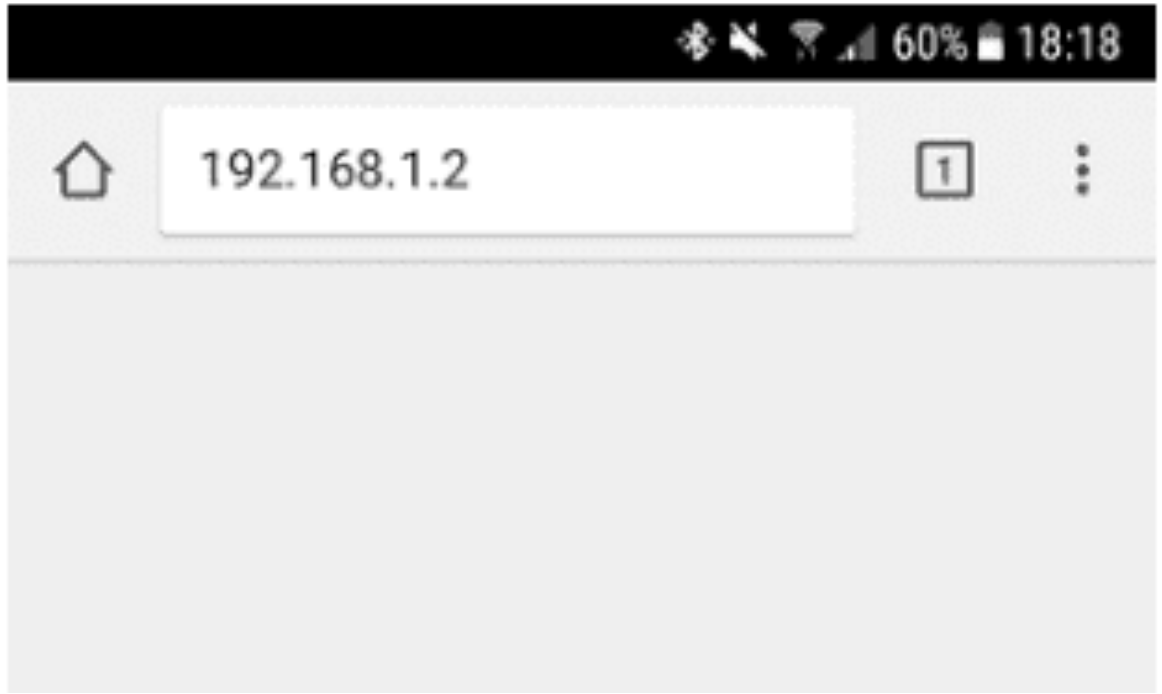
The first touch point with your PWA's interface and should stand out, reflect the brand

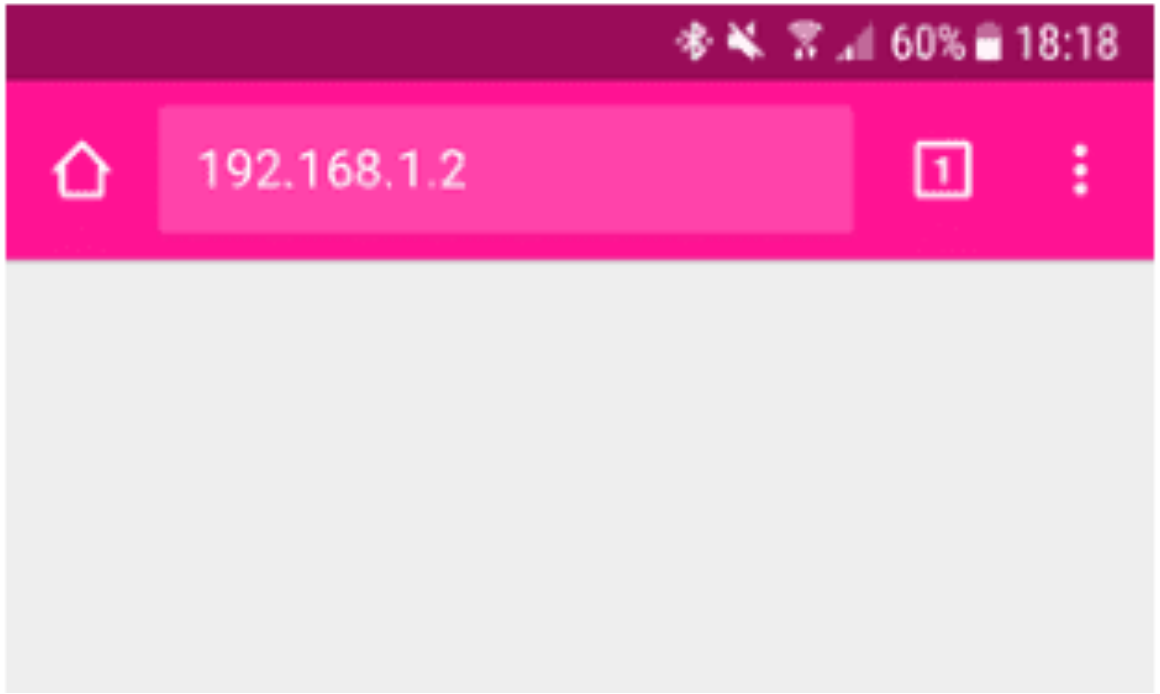
Streamlines the user experience by providing a shortcut from the home screen

Adaptive icons allow your icons to look more design conscious across platforms by filling an icon space rather than being squished into one

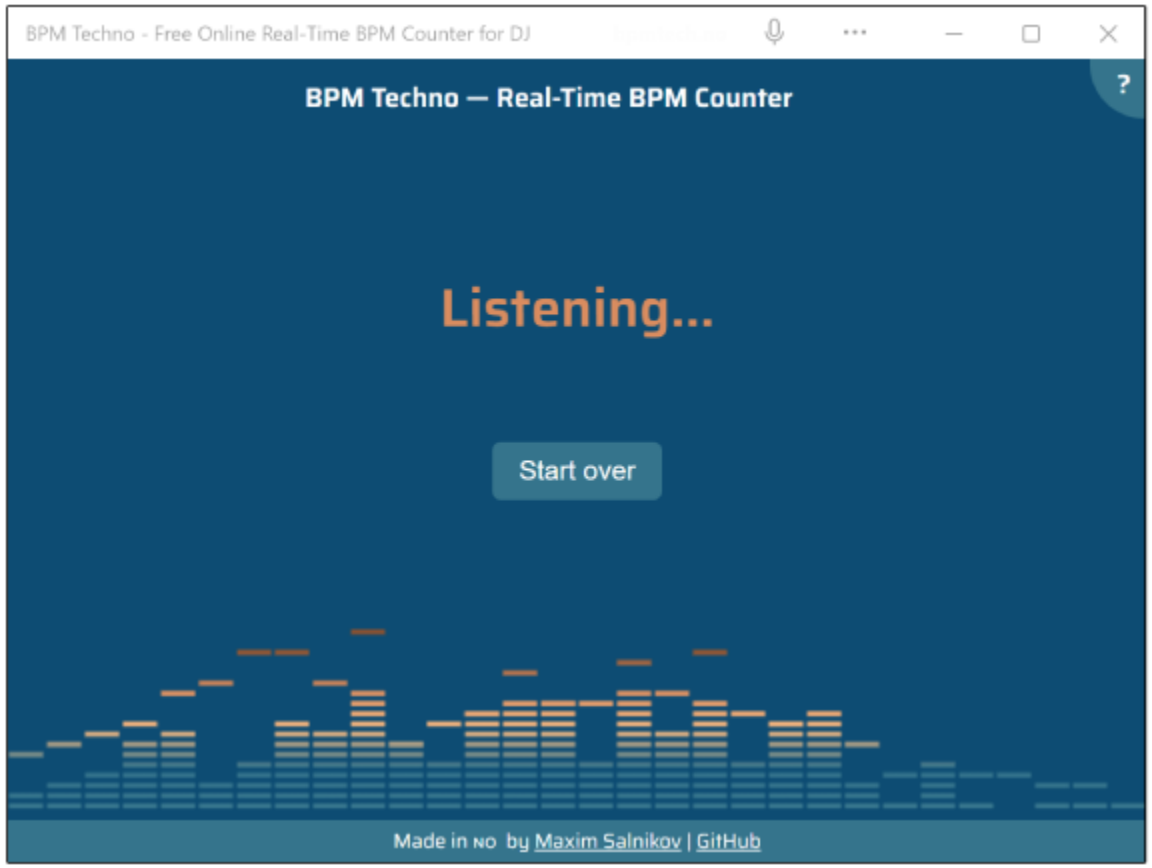
Defining a theme colour

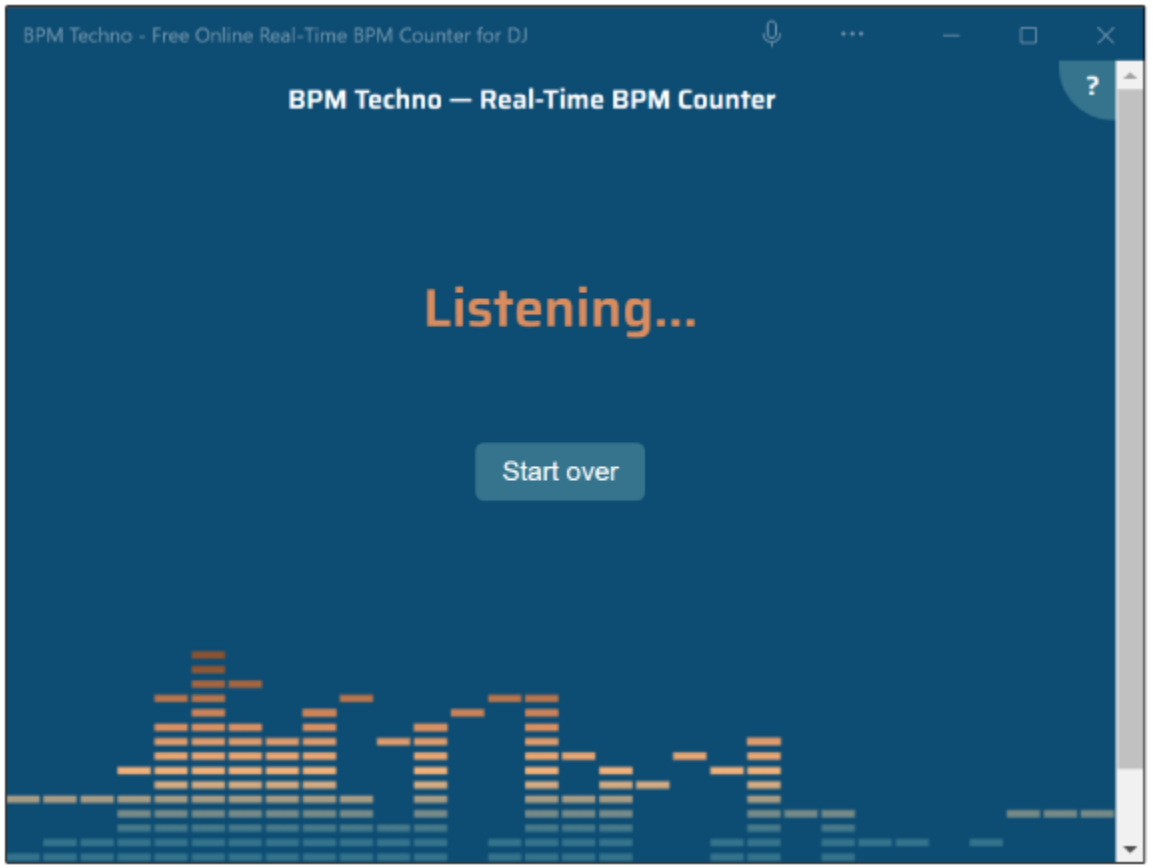
In browser





Standalone mode





Applying a theme color

Manifest member

```
{
    "theme_color": "#7700e7"
     …
}
```

Applying a theme color

HTML Meta tag

```
<meta name="theme-color" content="#7700e7”>
```

Summary: Theme color

Applying a theme color provides a more native app like experience

Opportunity to tie in your primary brand color, creating a sense of cohesion

Define via the web app manifest and a <meta tag> or just the web app manifest depending on the type of experience you want to deliver

Use system fonts

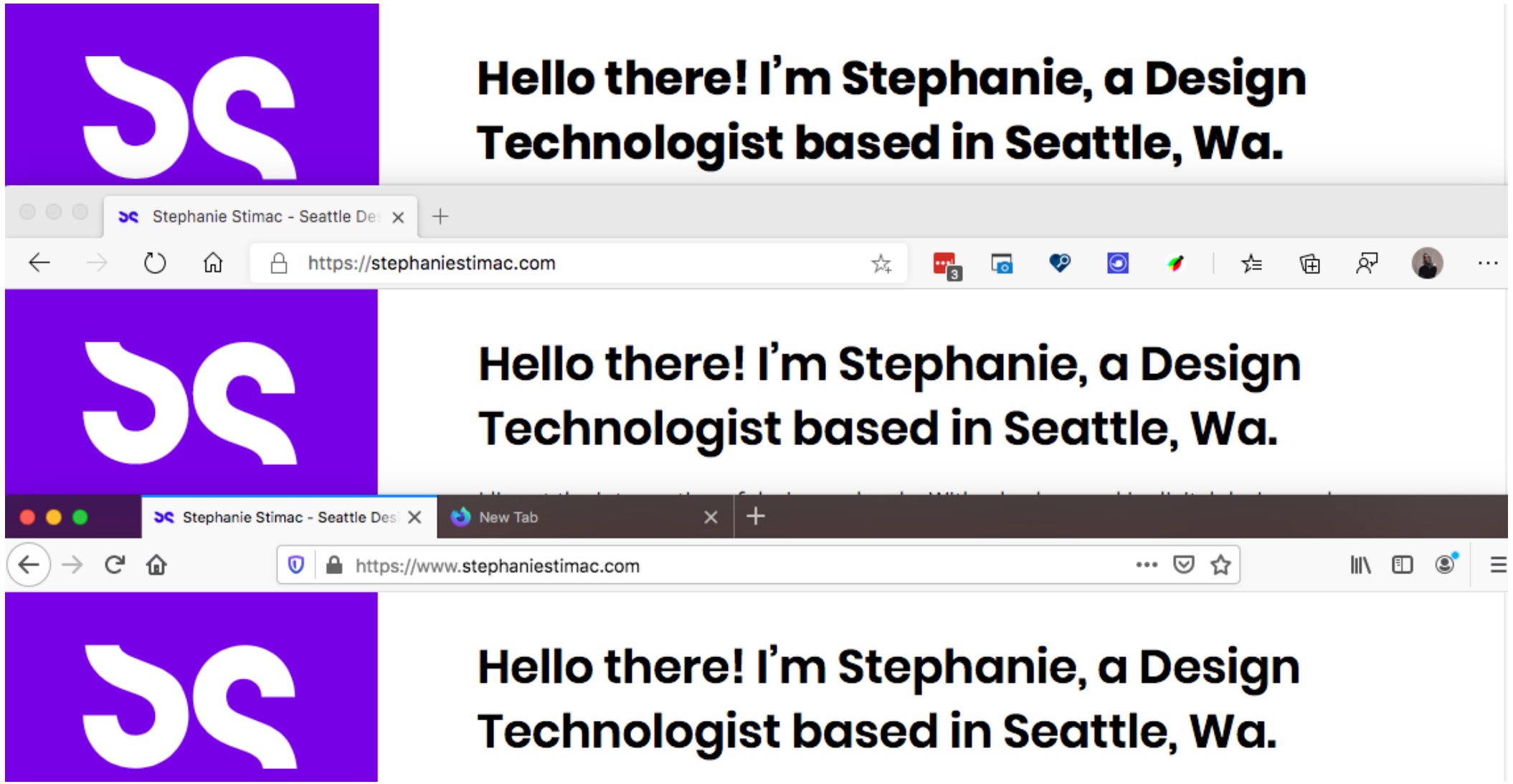
Android: Roboto

macOS, iPad OS and iOS: SF

Windows: Segoe UI

CSS Property + Value

```
font-family: system-ui;
```


Hello there! I'm Stephanie, a Design Technologist based in Seattle, Wa.
Stephanie Stimac - Seattle De
https://stephaniestimac.com
Hello there! I'm Stephanie, a Design Technologist based in Seattle, Wa.
Stephanie Stimac - Seattle Des
New Tab
https://www.stephaniestimac.com
Hello there! I'm Stephanie, a Design Technologist based in Seattle, Wa.

Summary: Use system fonts

Using system fonts has a more platform specific feel

There's also a performance benefit

Reduces overheard of managing a custom font across different platforms

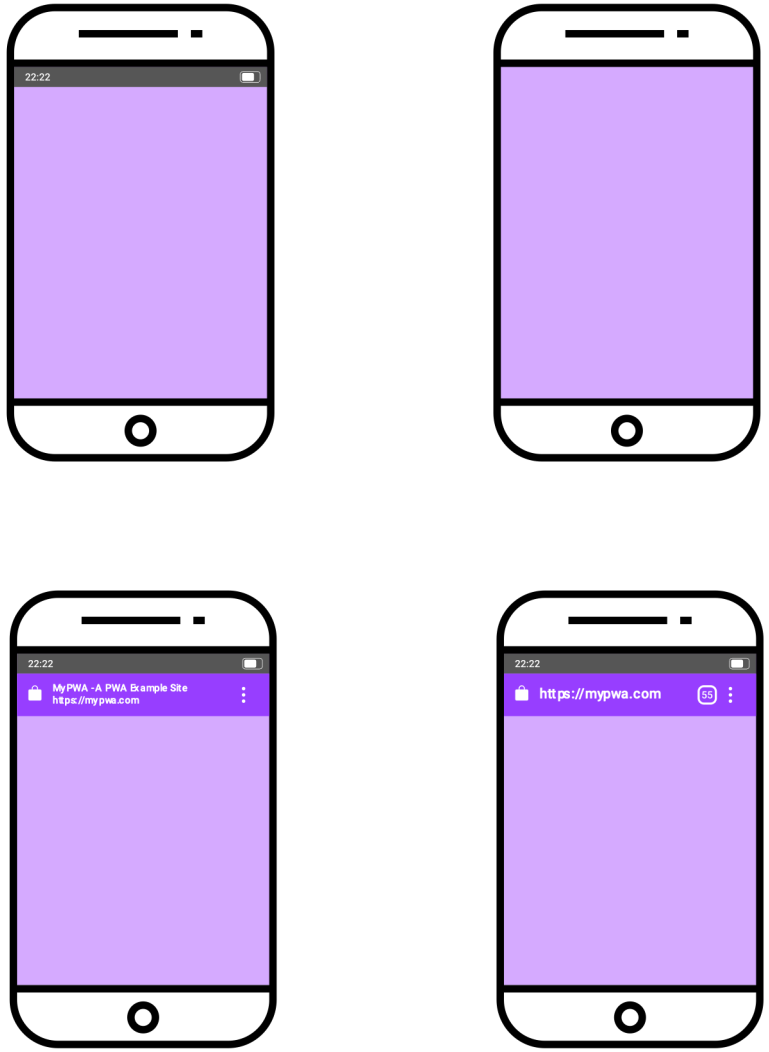
02 Responsive Considerations

Display modes

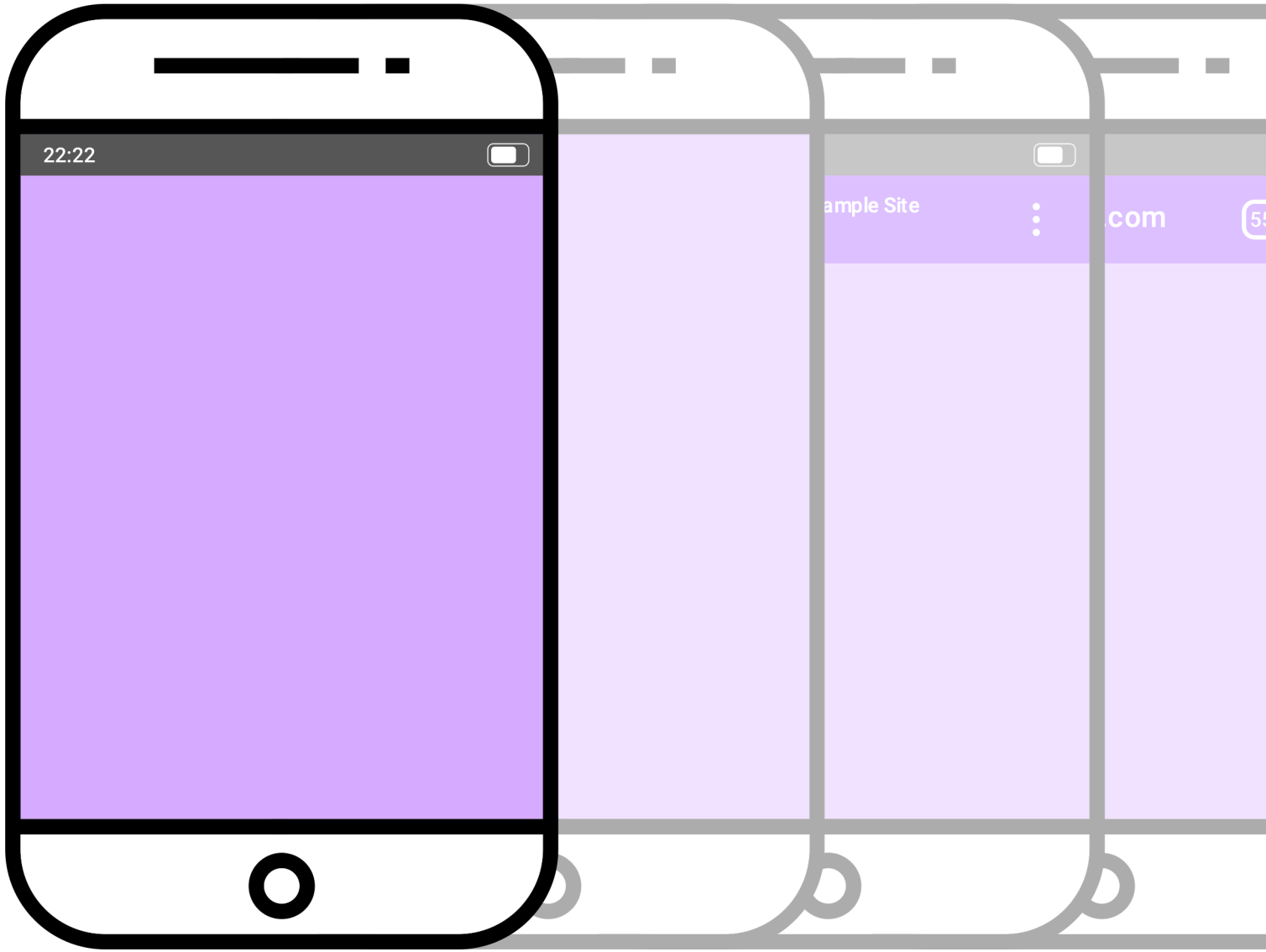
Manifest member

```
{
    “display”: “standalone”
}
```

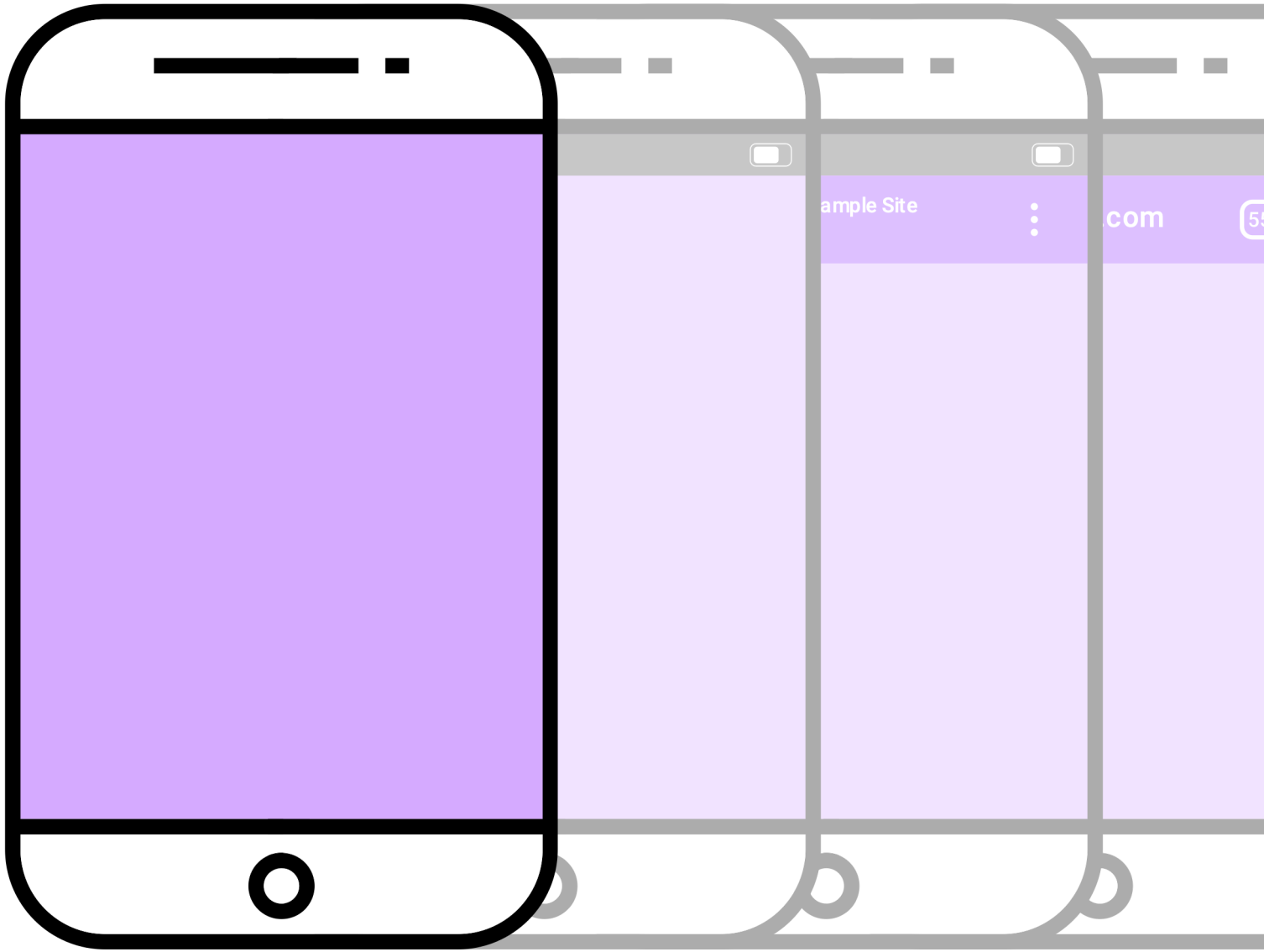
4 display modes



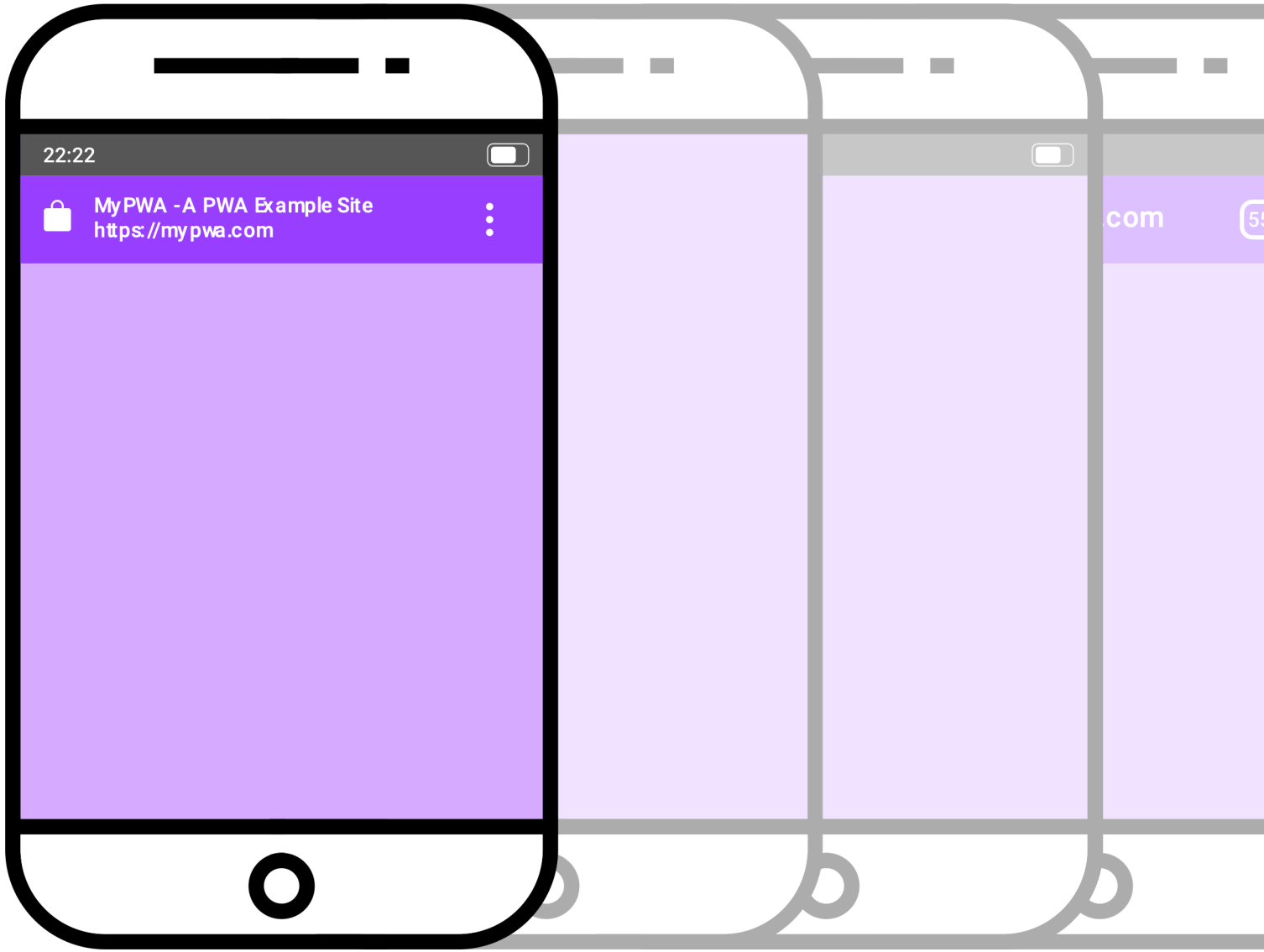
Standalone



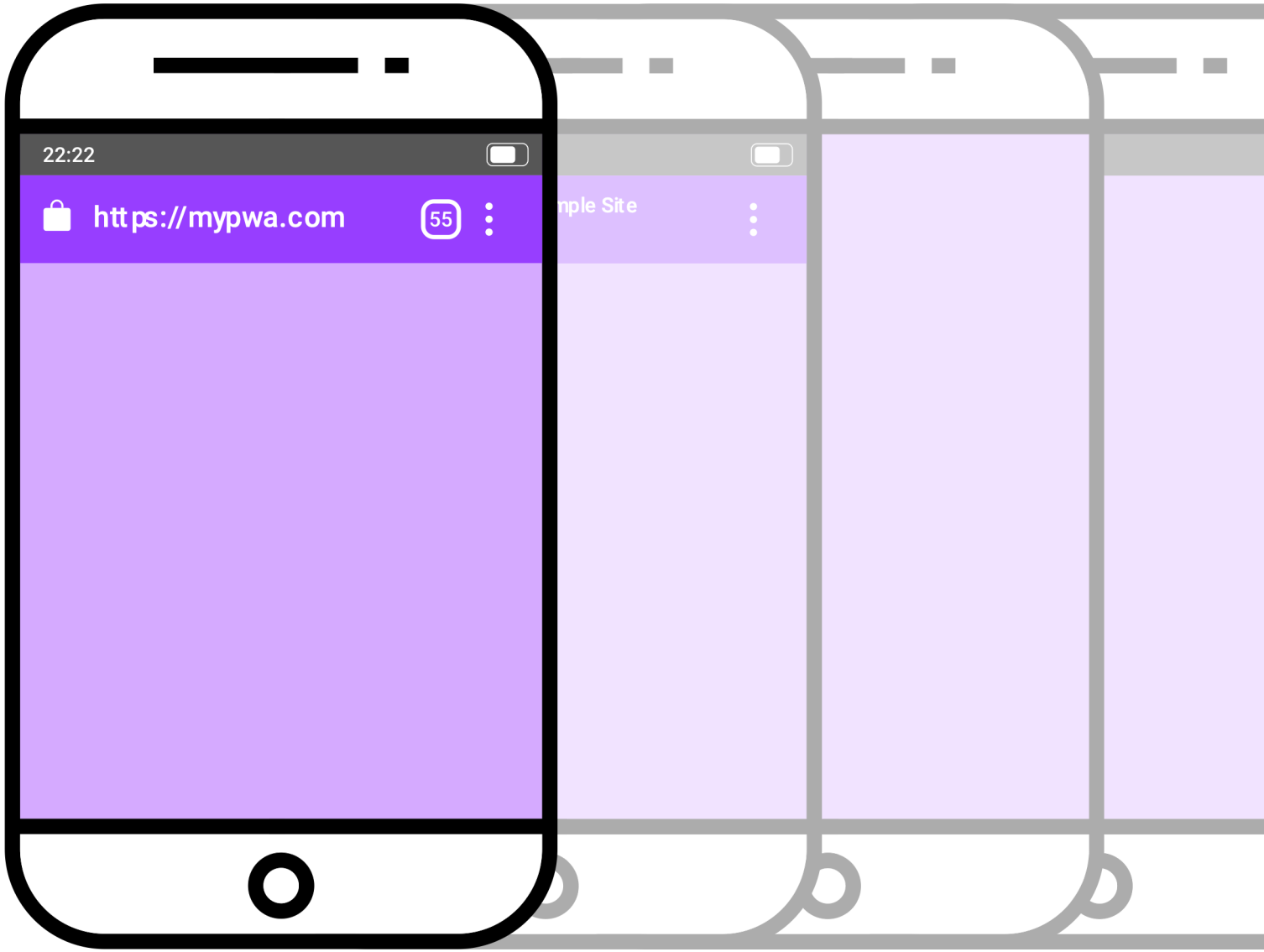
Fullscreen



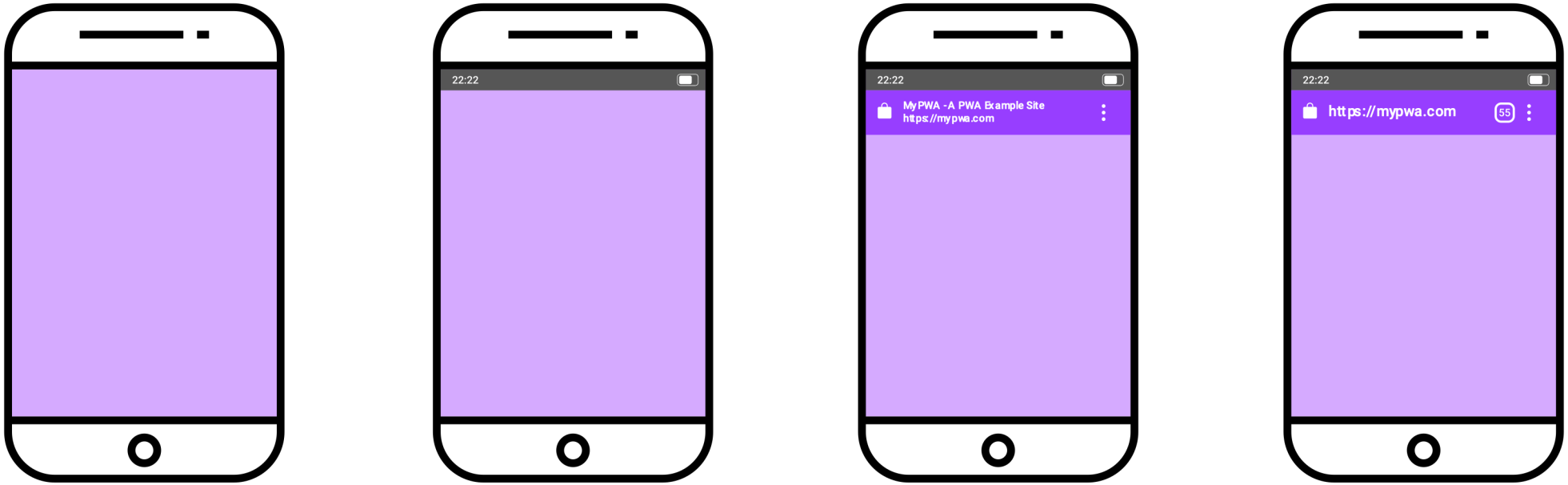
Minimal-UI



Browser

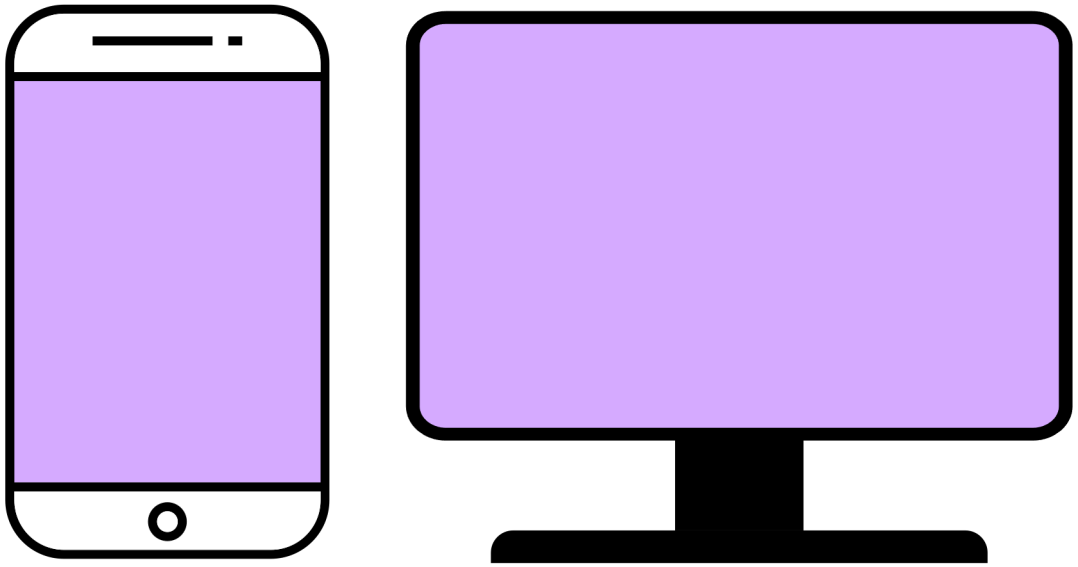


02 Responsive considerations / Display modes

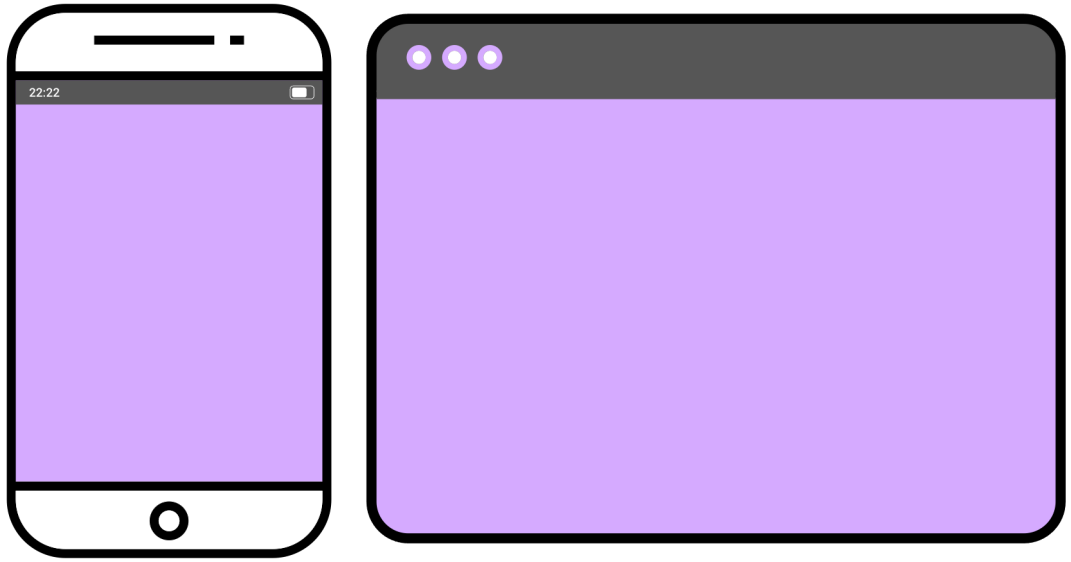


fullscreen > standalone > minimal-ui > browser

02 Responsive considerations / Display modes



fullscreen



standalone

Media queries with display-mode

January 12th, 2022

It's said that the best way to learn about something is to teach it. I certainly found that to be true when I was writing the web.dev course on responsive design.

I felt fairly confident about some of the topics, but I felt somewhat out of my depth when it came to some of the newer modern additions to browsers. The last few modules in particular were unexplored areas for me, with topics like screen configurations and media features. I learned a lot about those topics by writing about them.

Best of all, I got to put my new-found knowledge to use! Here's how…

The Session is a progressive web app. If you add it to the home screen of your mobile device, then when you launch the site by tapping on its icon, it behaves just like a native app.

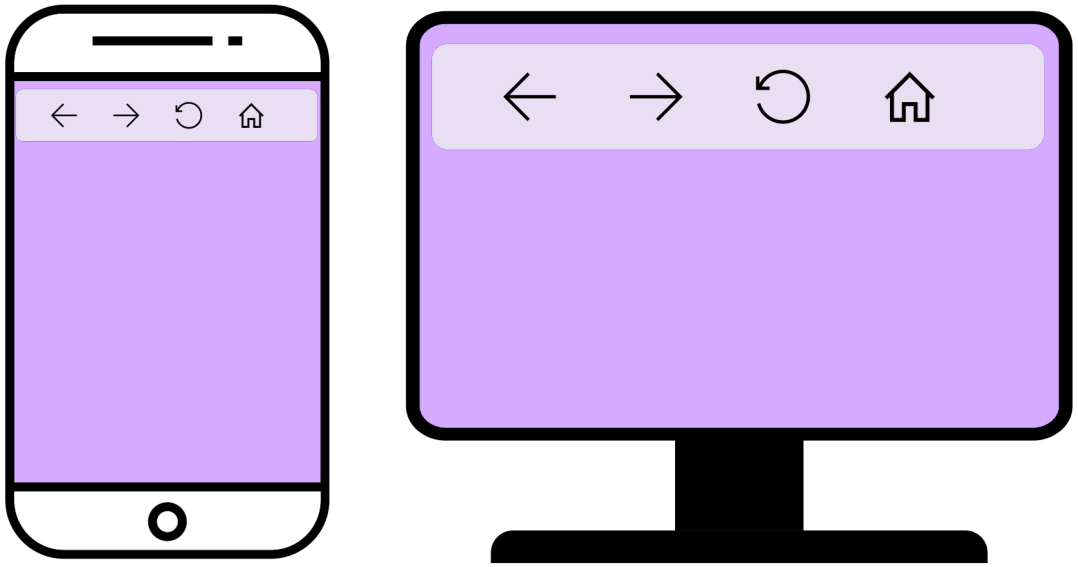
In the web app manifest file for The Session, the `display-mode` property is set to "standalone." That means it will launch without any browser chrome: no address bar and no back button. It's up to me to provide the functionality that the browser usually takes care of.

So I added a back button in the navigation interface. It only appears on small screens.

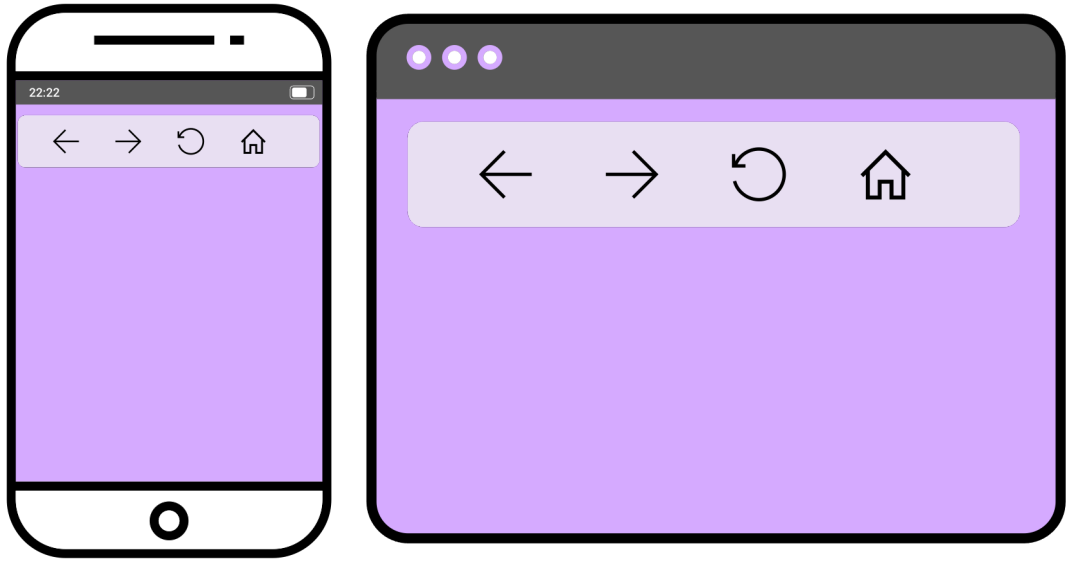
Do you see the assumption I made?

I figured that the back button was most necessary in the situation where the site had been added to the home screen

Ensure there's clear navigation

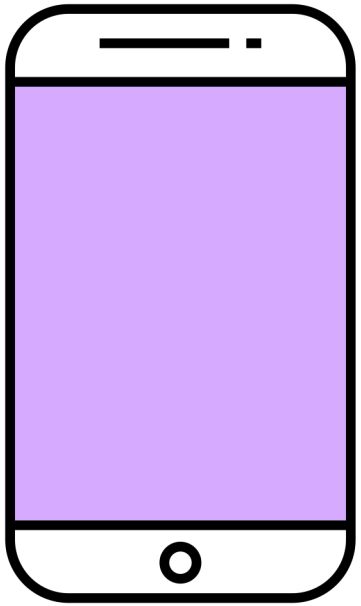


fullscreen

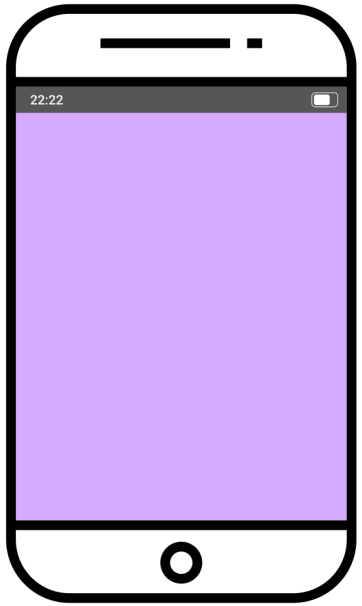


standalone

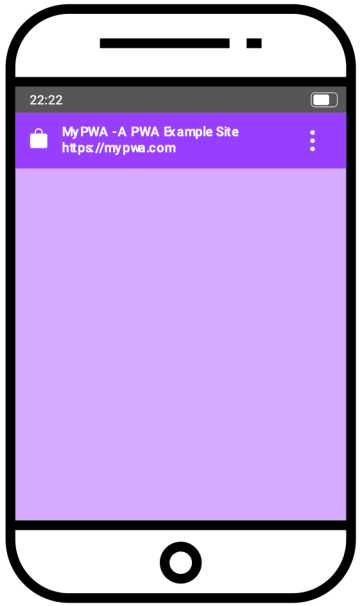
Layout considerations



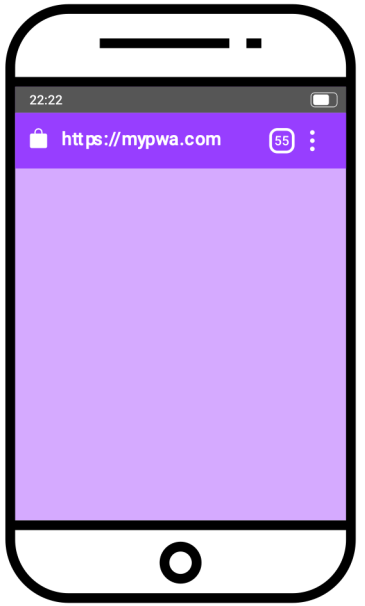
fullscreen



standalone



minimal-ui



browser

Target display modes with CSS

Media queries

```
/* It targets only the app used within the browser */
@media (display-mode: browser) {
}

/* It targets only the app used with a system icon in standalone mode */
@media (display-mode: standalone) {
}

/* It targets only the app used with a system icon in all mode */
@media (display-mode: standalone), (display-mode: fullscreen), (display-
mode: minimal-ui) {
}
```

Summary: Display modes

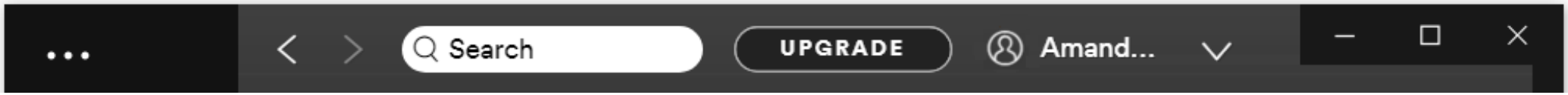
There are 4 display modes for your PWA that give experiences that are progressively more app like

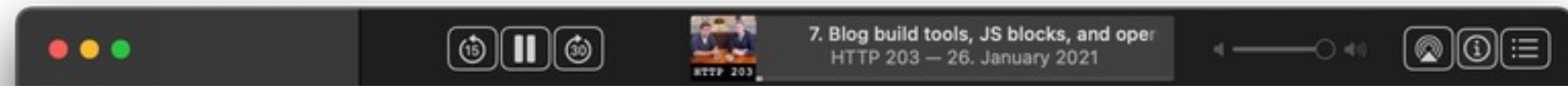
Not all display modes are supported by all browser so there is a fallback cascade

Some display modes have no browser UI, you need to ensure you have a way for users to navigate through your PWA easily without that UI

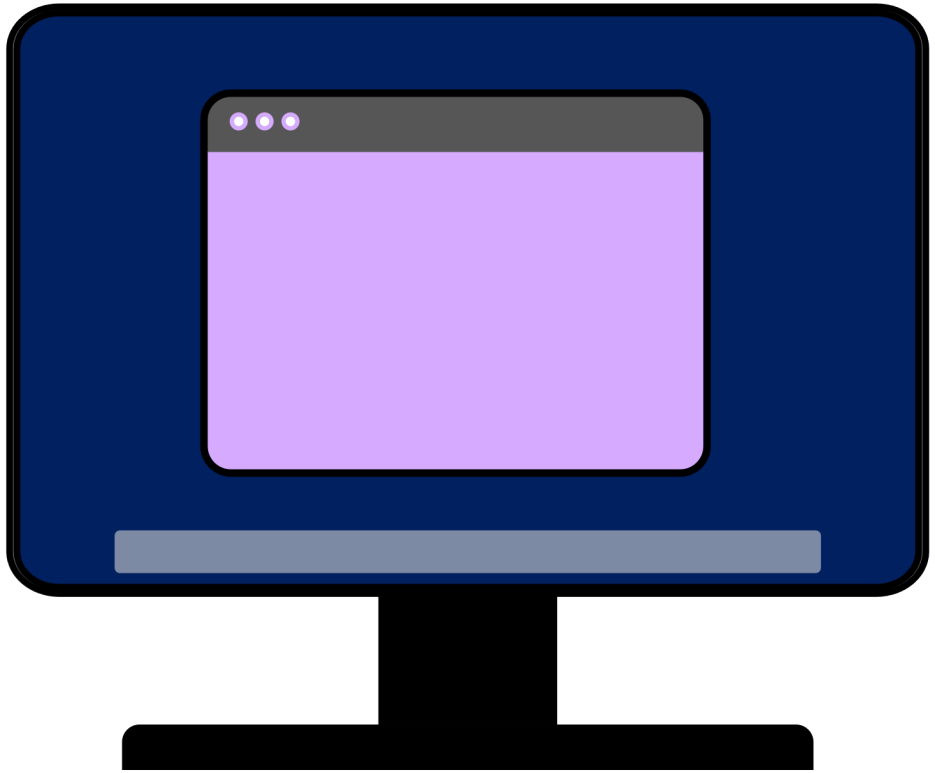
Window Controls Overlay

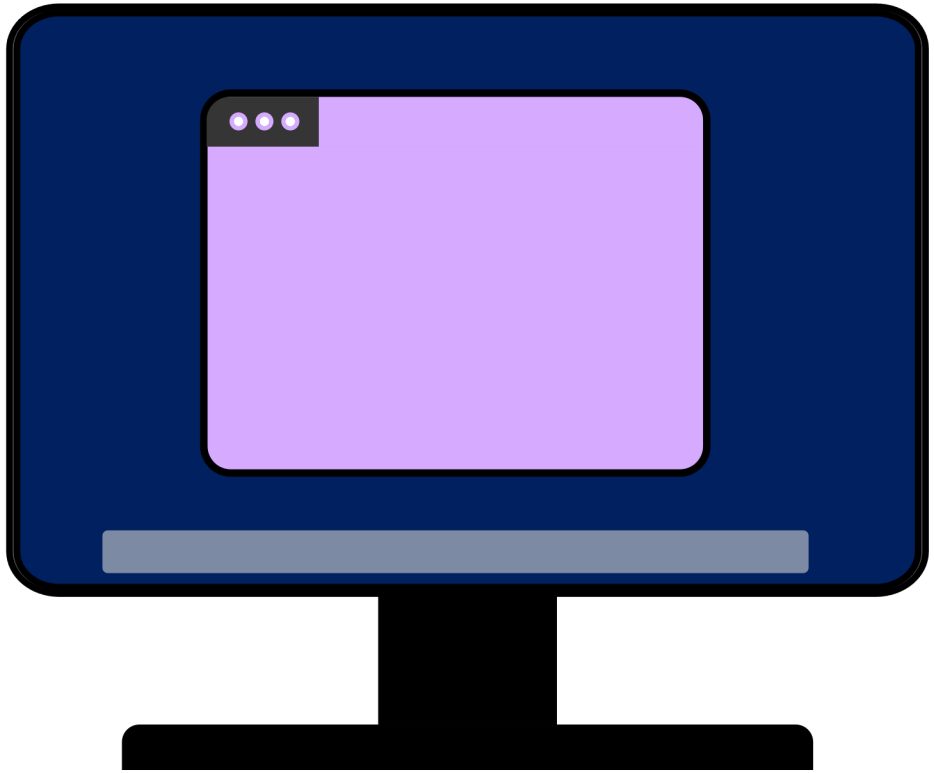
02 Responsive considerations / Window Controls Overlay





02 Responsive considerations / Window Controls Overlay

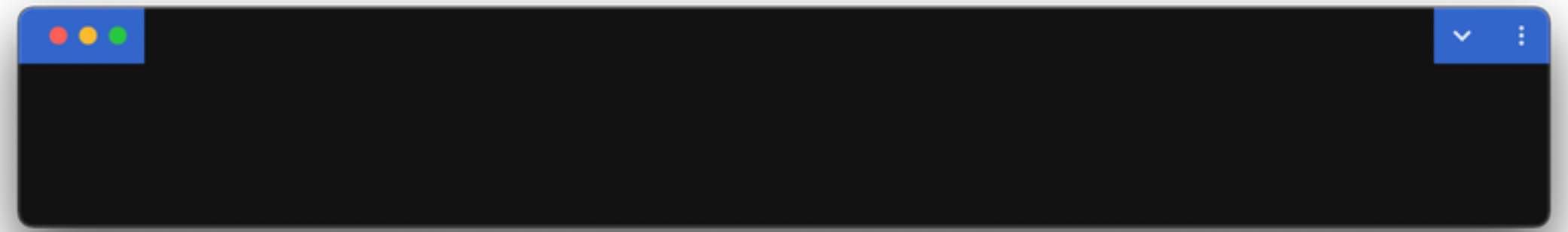




Manifest member

```
{
      “display”: “standalone”,
      "display_override": ["window-controls-overlay"],
}
```

02 Responsive considerations / Window Controls Overlay



1. The app is not opened in the browser, but in a separate PWA window.
2. The manifest includes "display_override": ["window-controls-overlay"].
3. The PWA is running on a desktop operating system.
4. The current origin matches the origin for which the PWA was installed.

CSS environment variables

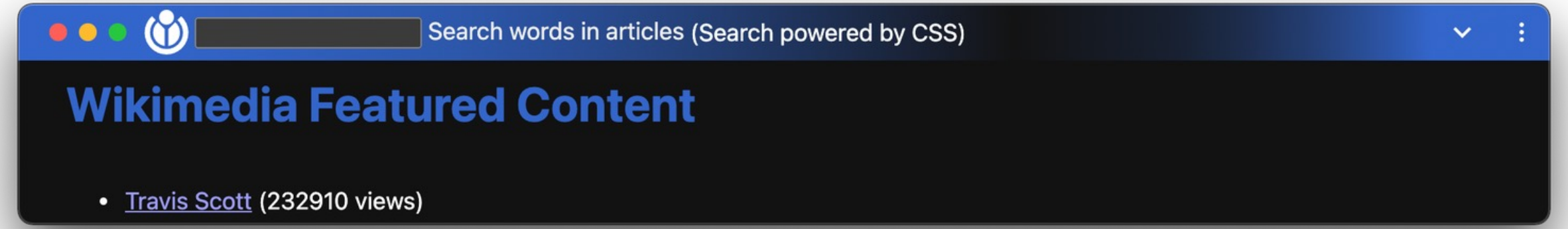
```
titlebar-area-x
titlebar-area-y
titlebar-area-width
titlebar-area-height
```

https://wicg.github.io/window-controls-overlay/

CSS environment variables

```
#title-bar {
        position: fixed;
        left: env(titlebar-area-x, 0);
        top: env(titlebar-area-y);
        height: env(titlebar-area-height);
        width: env(titlebar-area-width, 100%);
}
```

https://aka.ms/WCO-Design



https://web.dev/window-controls-overlay/

Ensure titlebar parts are still draggable

```
#title-bar {
        -webkit-app-region: drag;
        app-region: drag;
}

input {
        -webkit-app-region: no-drag;
        app-region: no-drag;
}
```

https://web.dev/window-controls-overlay/

Summary: Window controls overlay

Gives you more control over the design and the content within the titlebar to be even more app like

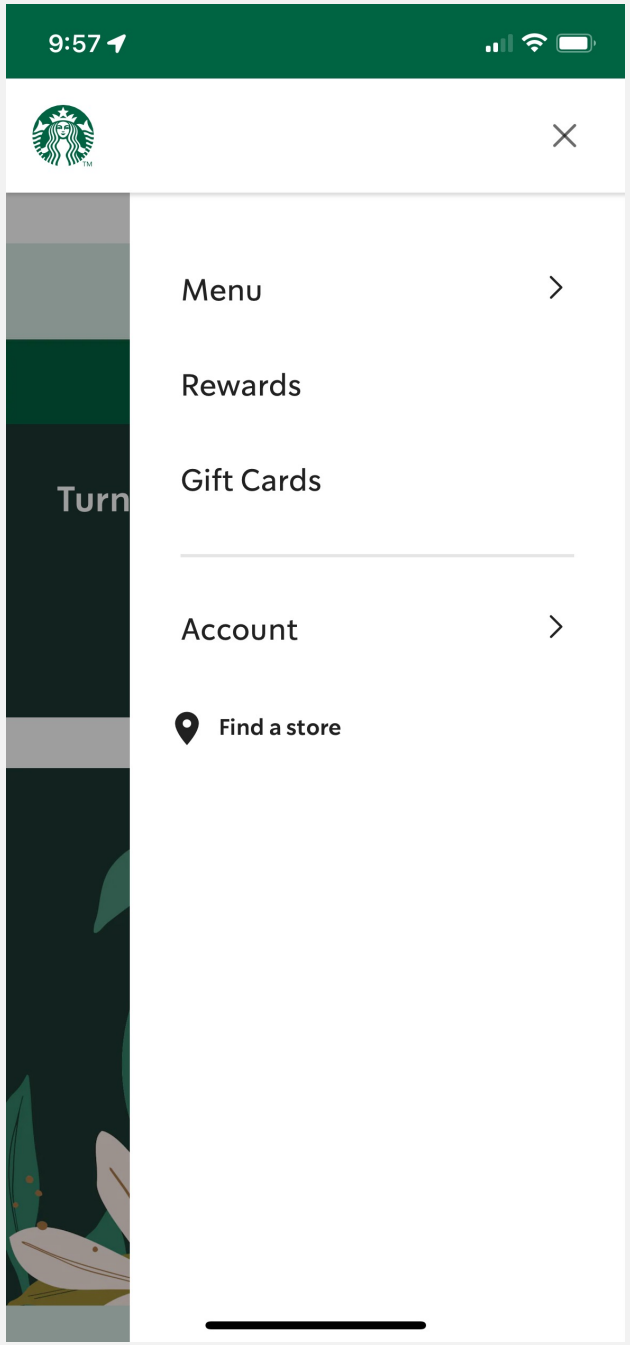
Available on desktop

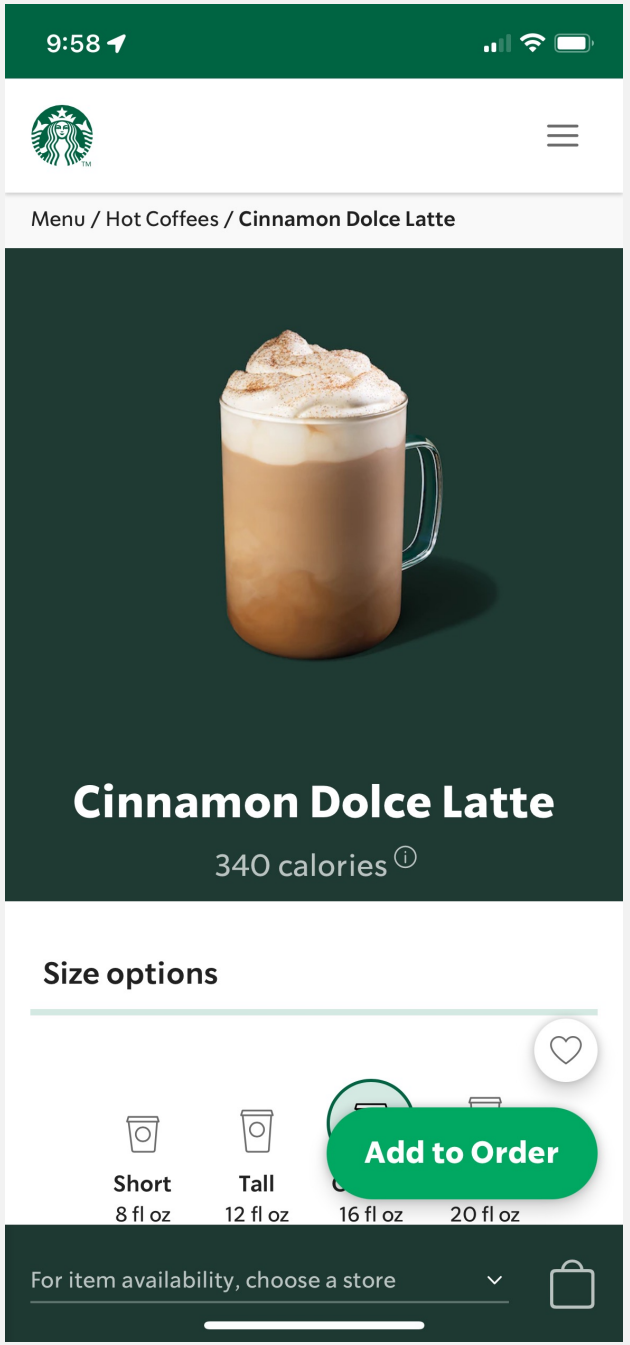
If on a platform that’s not supported, will fallback to the next supported mode, like “standalone”

03 User Experience Considerations

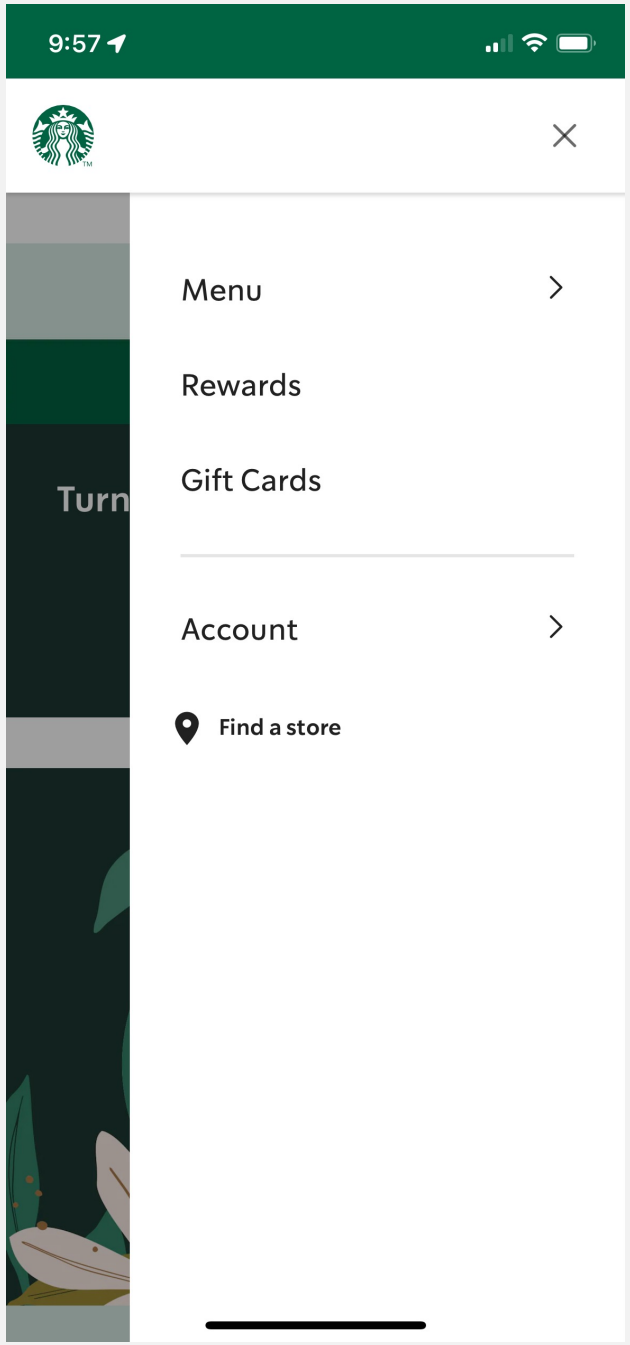
Go lite and prioritize content

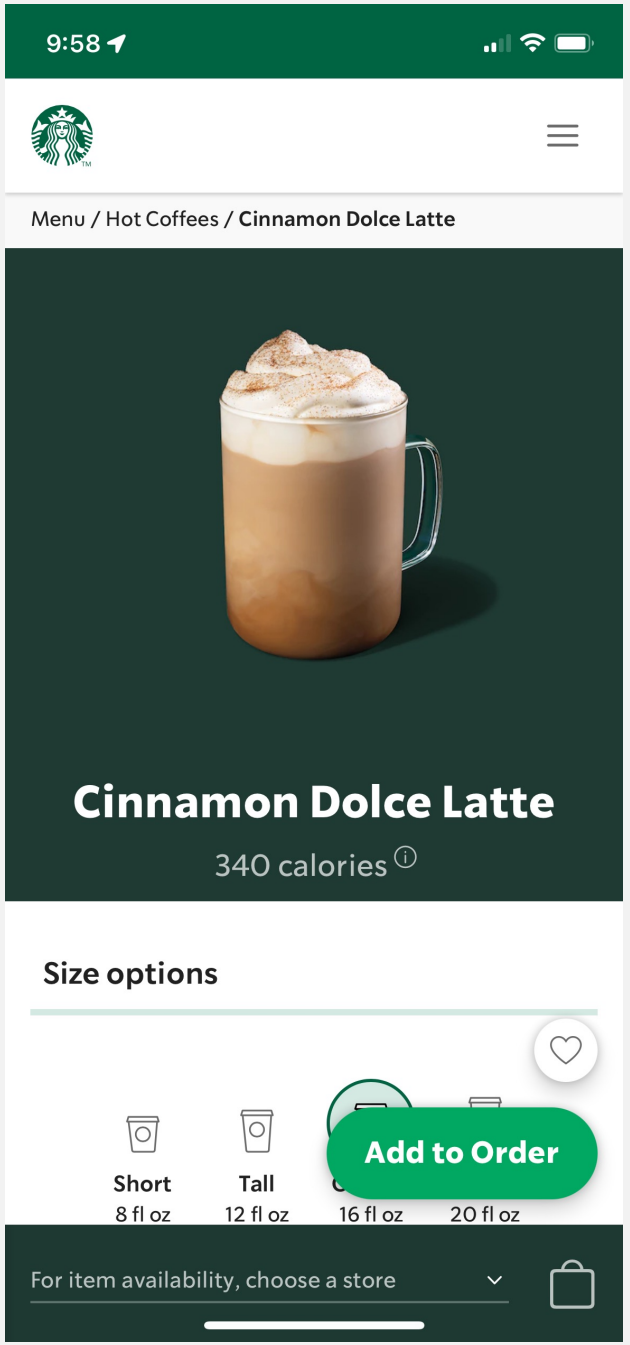
Less is more


9:57
Menu
Rewards
Gift Cards
Account
Find a store

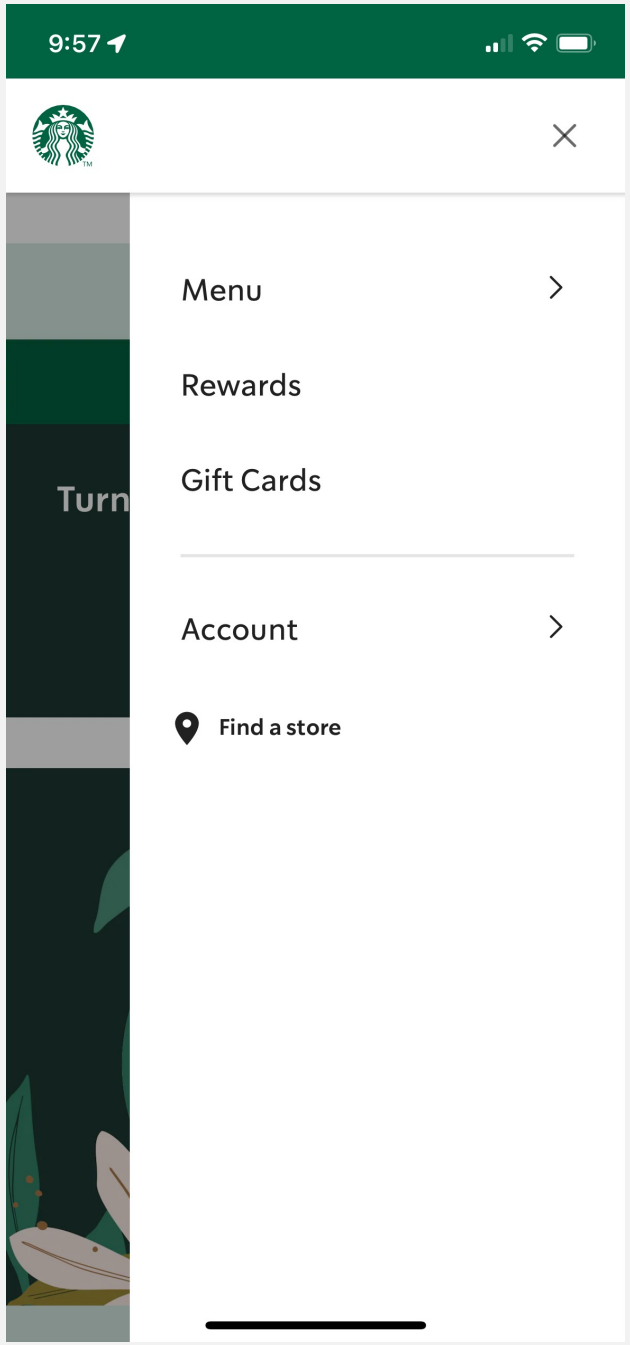

9:58
Menu / Hot Coffees / Cinnamon Dolce Latte
Cinnamon Dolce Latte
340 calories
Size options
Short
8 fl oz
Tall
12 fl oz
16 fl oz
20 fl oz
Add to Order
For item availability, choose a store

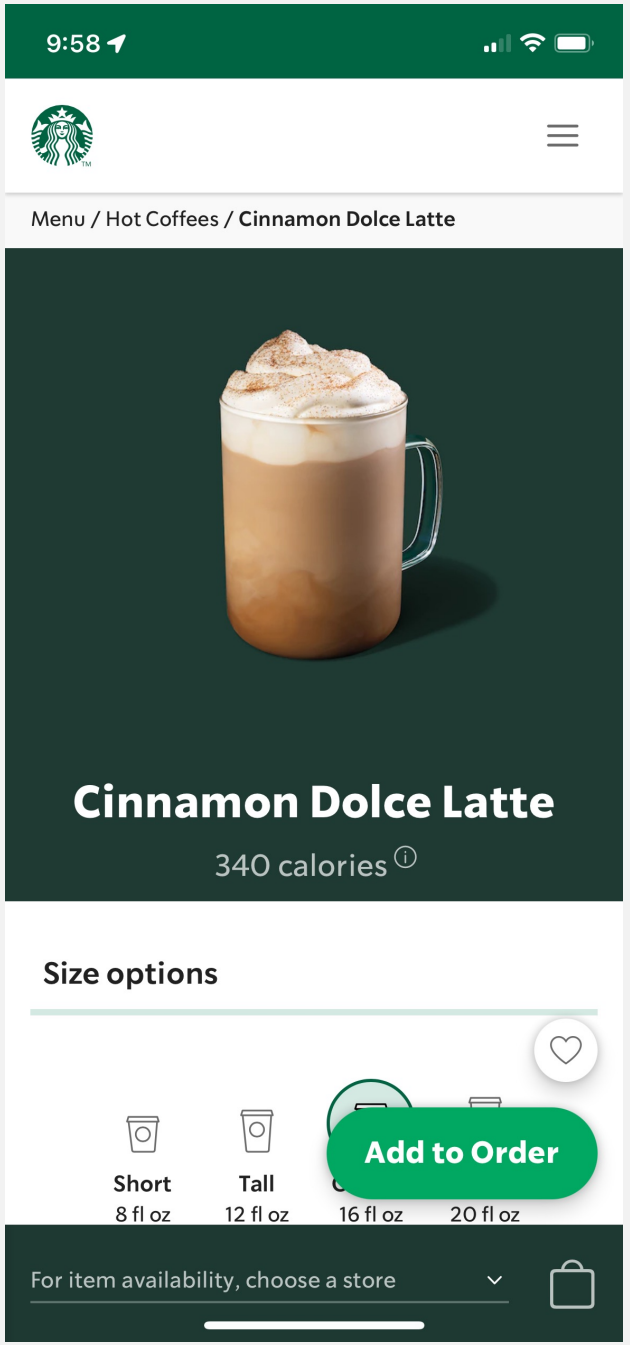
Identify the top tasks



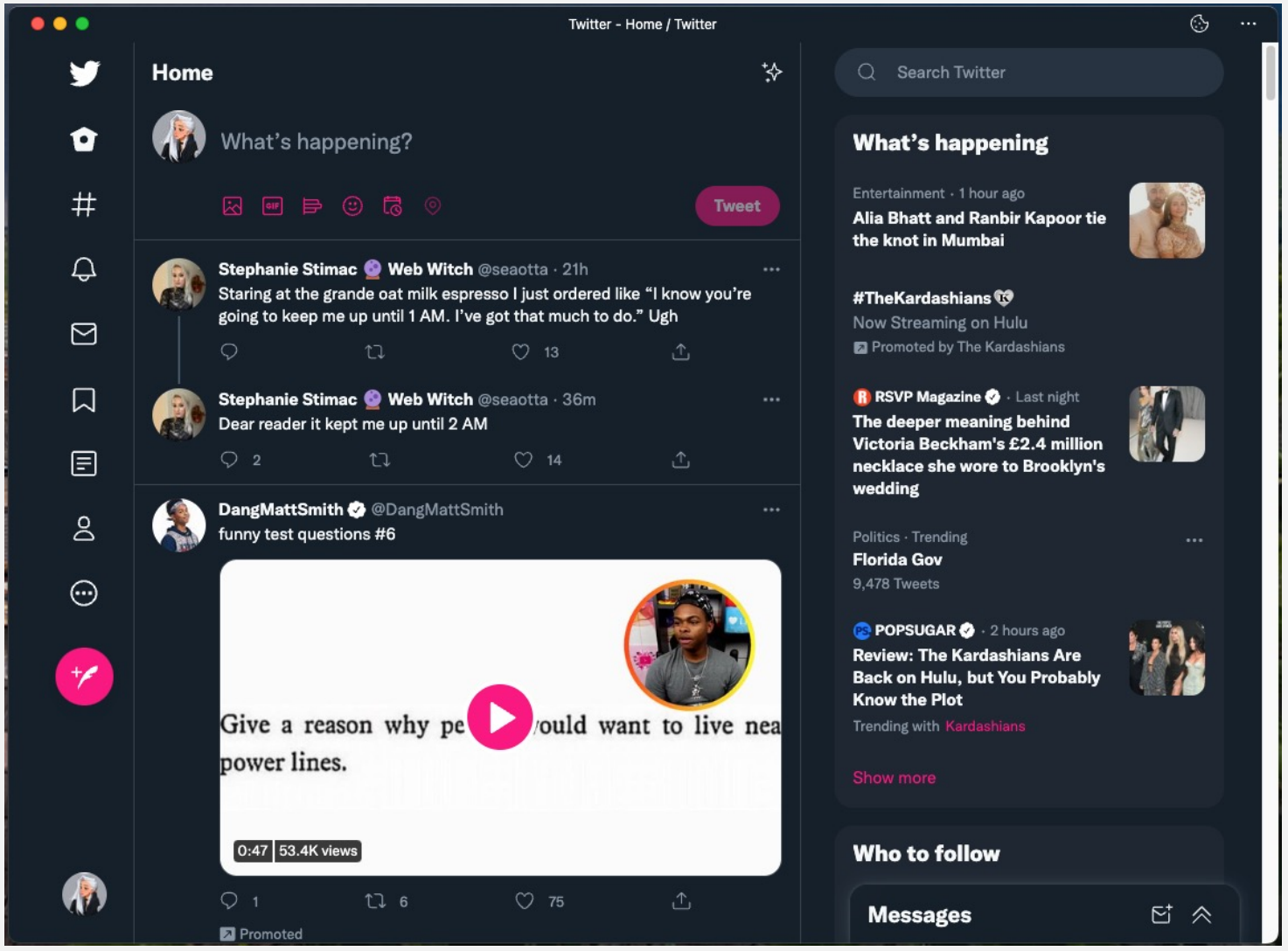


How can you make it faster?

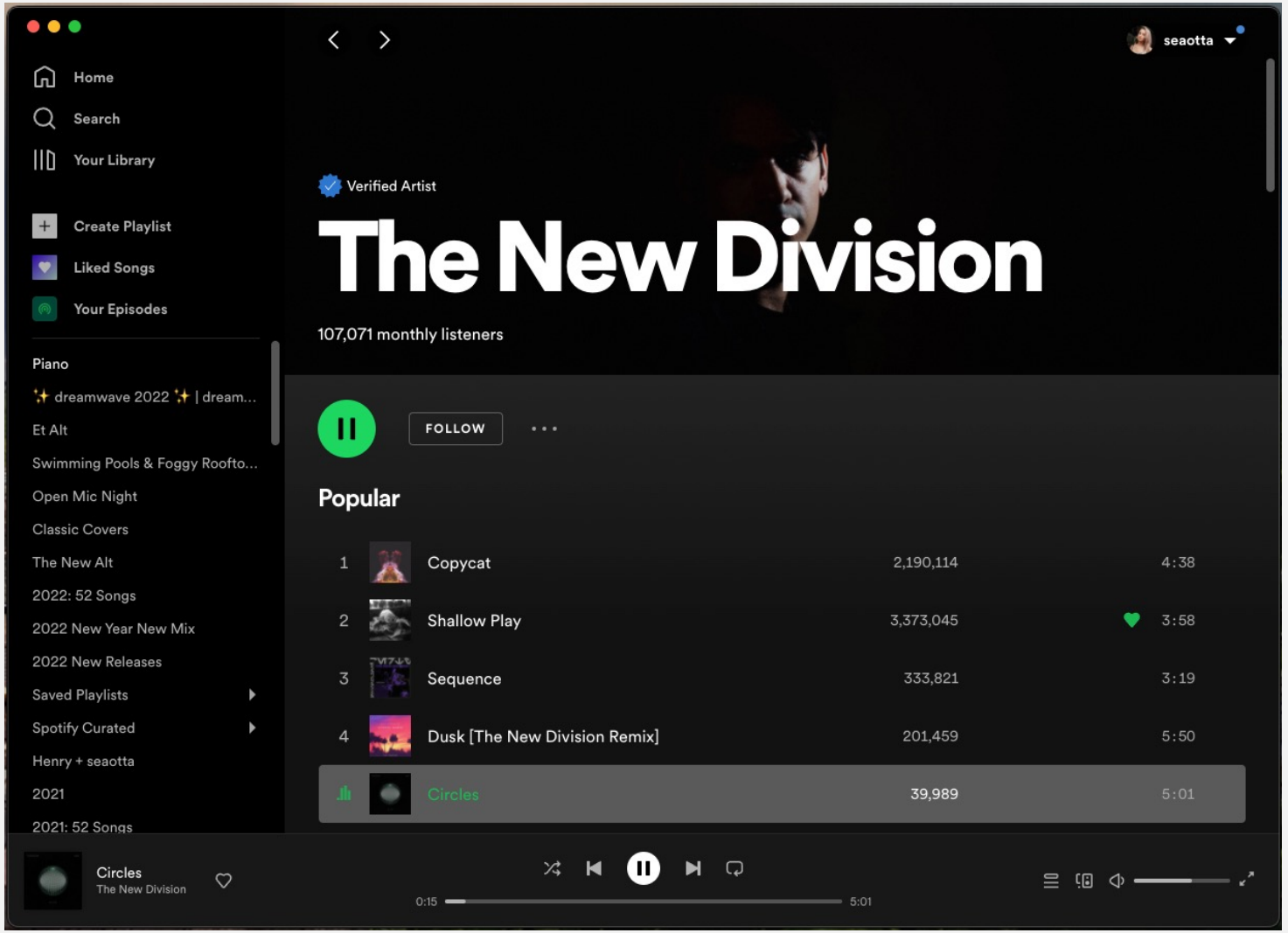




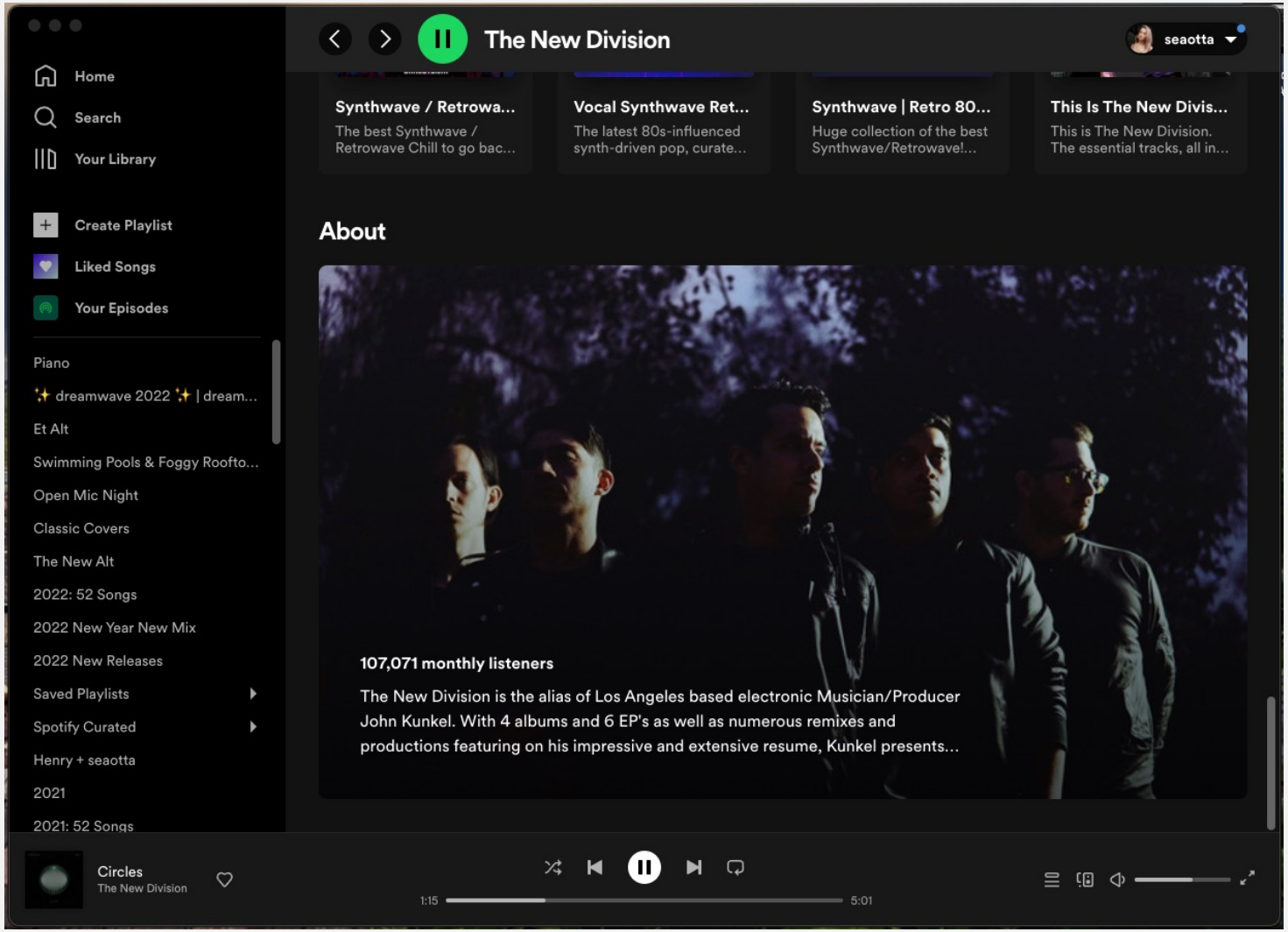
Reduce clutter: lose the footer



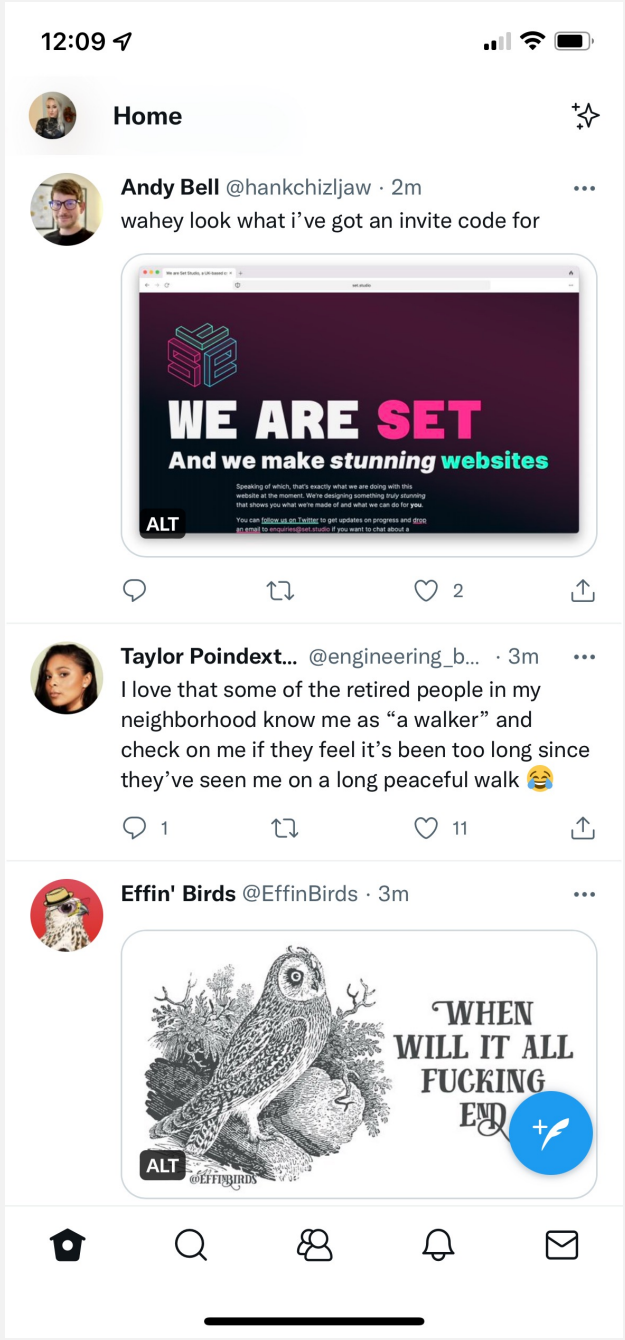
Reduce clutter: lose the footer

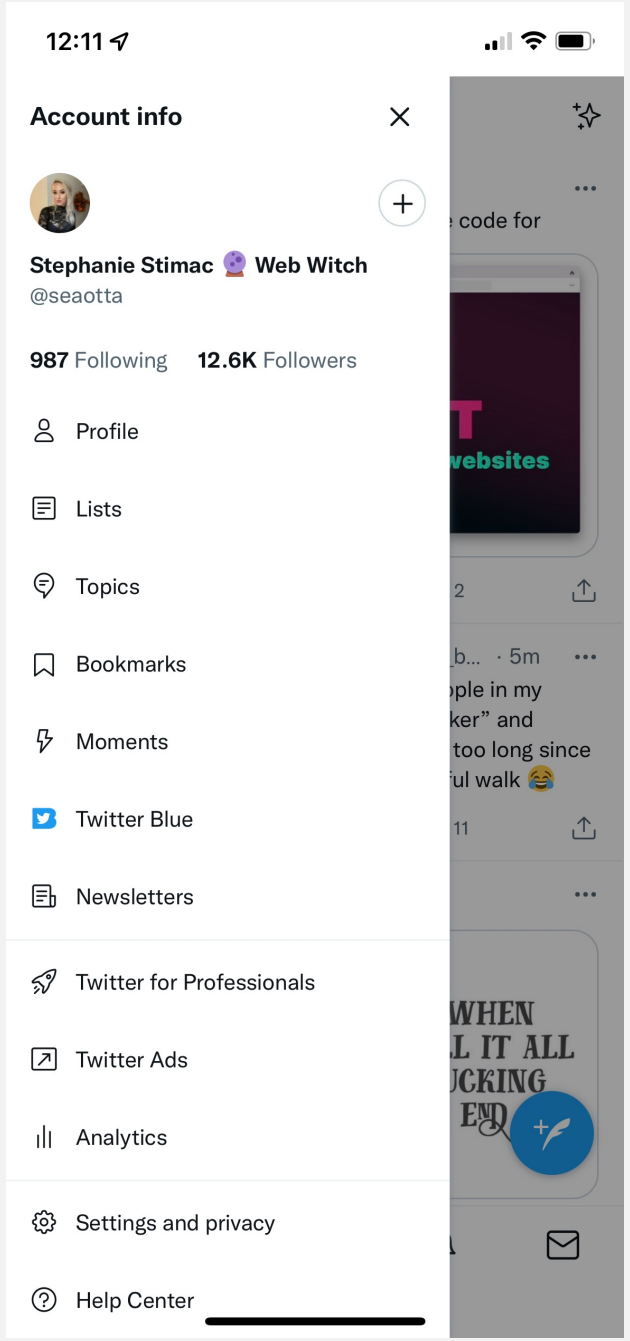


Reduce clutter: lose the footer

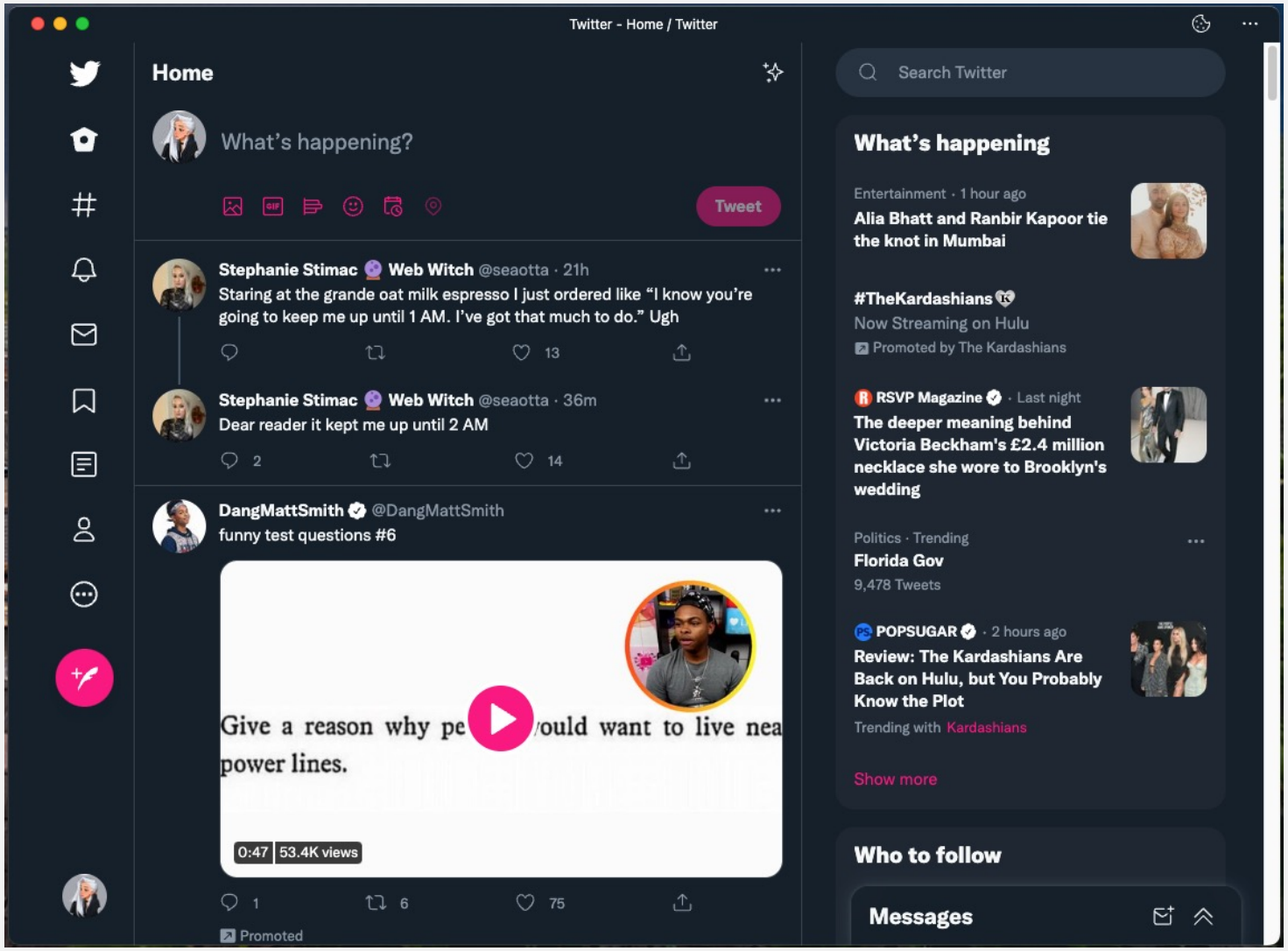


Reduce clutter: mobile UI

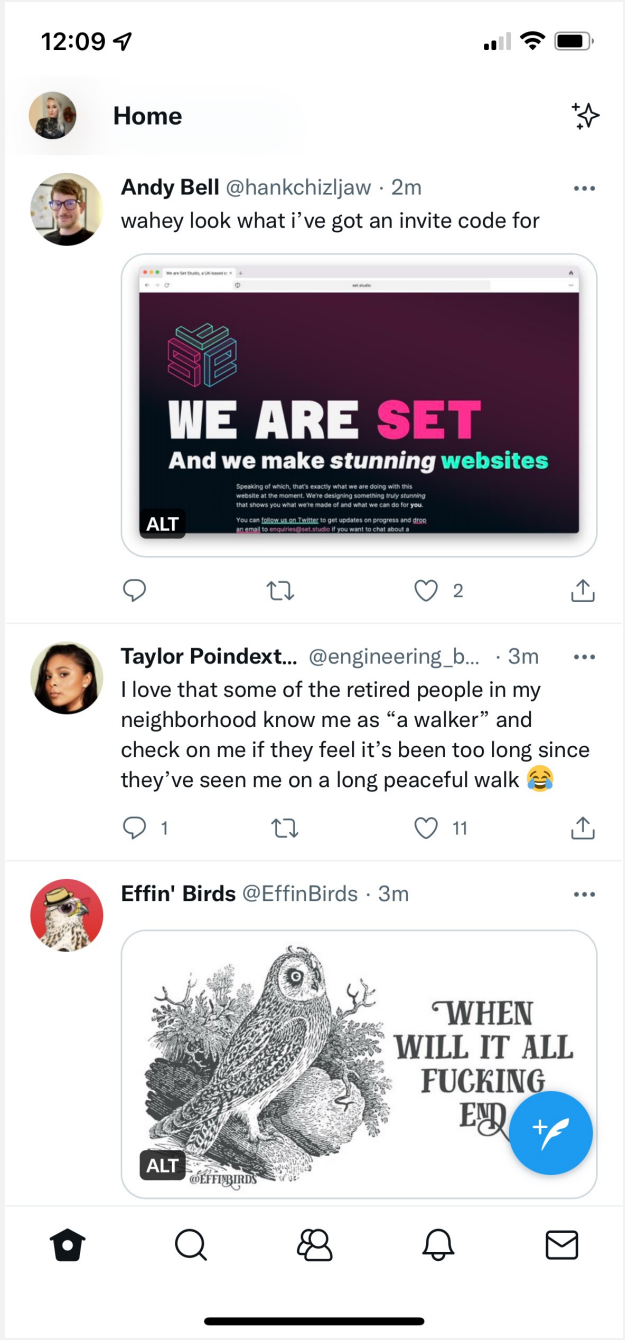


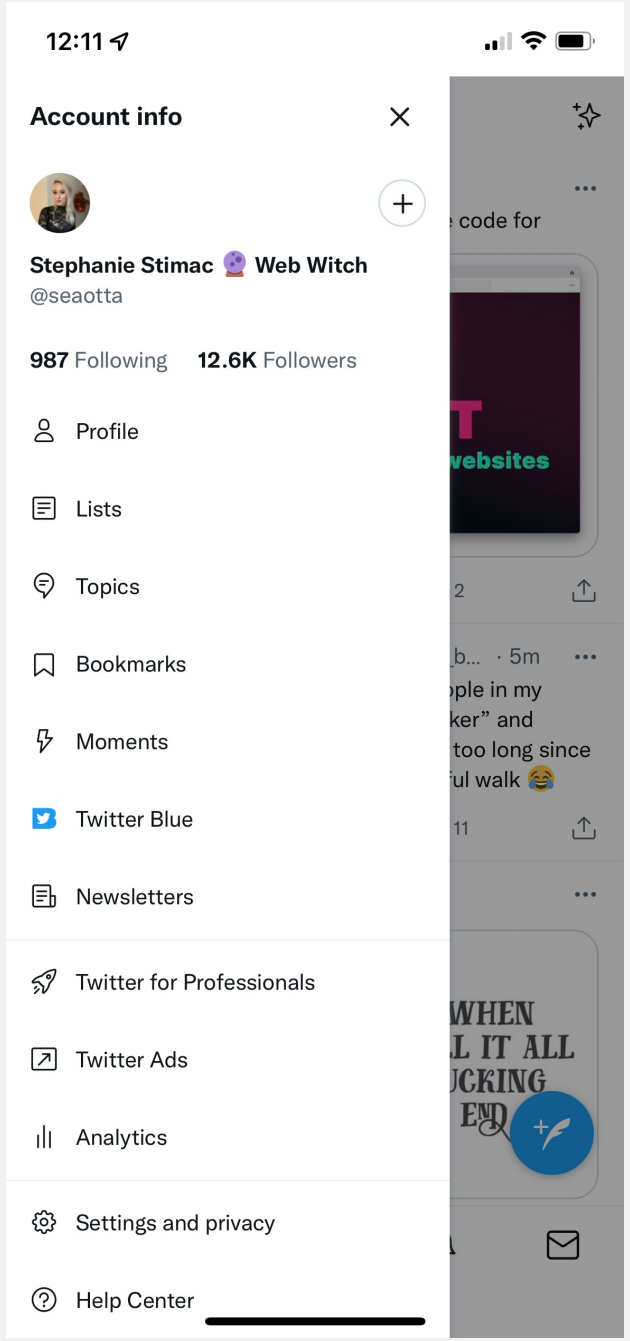


Reduce clutter: mobile UI



Reduce clutter: mobile UI





Summary: Go lite and prioritize content

Less is more, focus on the main tasks users are trying to complete and prioritize those experiences and content

For more app like experiences, lose the footer to reduce clutter

Different experiences on desktop vs mobile are fine as you take advantage of more real estate space.

On mobile, focus on bubbling up the most important UI elements that help users complete their main tasks

Make interactions feel seamless + fast

Skeleton UI for loading

Skeleton UI for loading

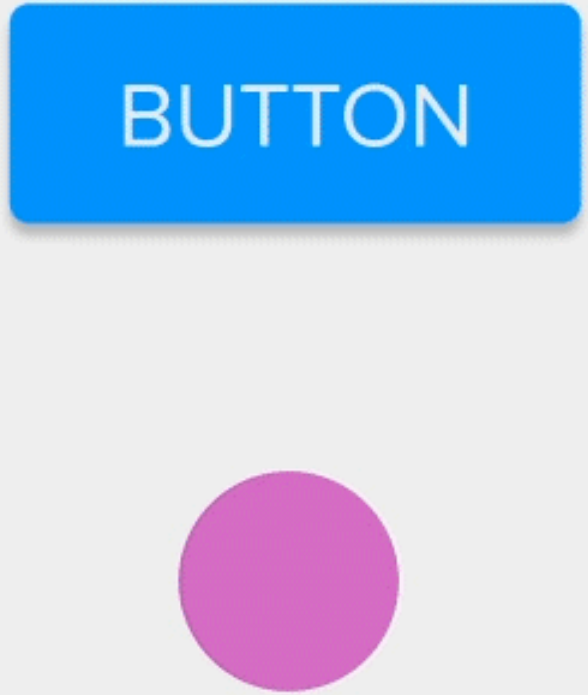
Skeleton UI for loading

“If you can display content right away, by all means, do that instead.”

Tim Kadlec
Director of Engineering
WebPageTest

https://aka.ms/Skeleton-UI

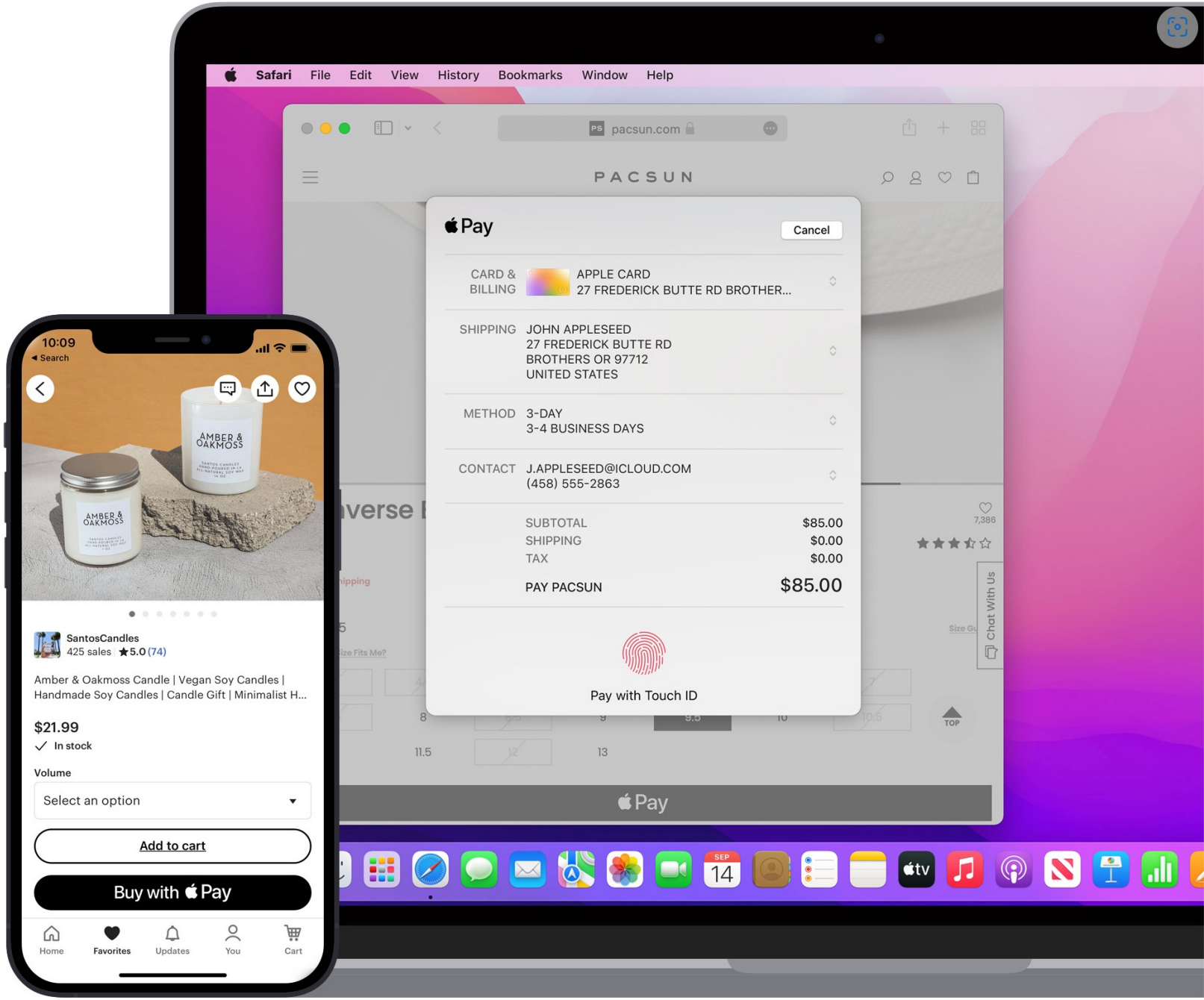
Provide interaction feedback



Source: Shopify

Integrate with features to make tasks easier and faster to complete

03 User experience considerations



03 User experience considerations / Seamless & fast interactions

PWA

Features

Media capture

Media capture allows apps to use the camera and microphone of a device

Contact picker

The Contact Picker API allows apps to select the user's contacts after permission has been granted.

File System

Access to the file system of the user's device

Audio

The Media Session API allows an app to display controls for media playback on a device's lock screen.

AR/VR

Augmented reality enables apps to place virtual objects in reality.

Geolocation

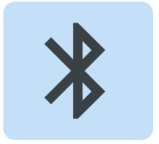
The Geolocation API enables users to share their location with a web app.

Web share

The Web Share API invokes the native share mechanism of the device and allows users to share text, URLs or files.

Vibration

The Vibration API enables web apps to make a mobile device vibrate.



Bluetooth

The Web Bluetooth API enables apps to connect to Bluetooth Low Energy (BLE) devices and read values from or write values to it.

Payment

The Payment Request API provides a browser-based method to enable users to make payments on the web, using a credit card, Apple Pay or Google Pay.

Notifications

The Notifications API enables web apps to display notifications, even when the app is not active.

Authentication

Web Authentication API (WebAuthn) enables passwordless authentication through your device's fingerprint reader or an external USB Security Key.

Audio recording

Record audio using MediaRecorder and visualize audio using Web Audio API.

NFC

The Web NFC API enables web apps to read and write to NFC tags.

Wake lock

The Screen Wake Lock API enables web apps to prevent devices from dimming or locking the screen when the app needs to keep running.

03 User experience considerations / Seamless & fast interactions

PWA

Features

Media capture

Media capture allows apps to use the camera and microphone of a device

Contact picker

The Contact Picker API allows apps to select the user's contacts after permission has been granted.

File System

Access to the file system of the user's device

Audio

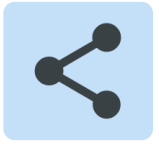
The Media Session API allows an app to display controls for media playback on a device's lock screen.

AR/VR

Augmented reality enables apps to place virtual objects in reality.

Geolocation

The Geolocation API enables users to share their location with a web app.

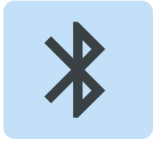


Web share

The Web Share API invokes the native share mechanism of the device and allows users to share text, URLs or files.

Vibration

The Vibration API enables web apps to make a mobile device vibrate.



Bluetooth

The Web Bluetooth API enables apps to connect to Bluetooth Low Energy (BLE) devices and read values from or write values to it.

Payment

The Payment Request API provides a browser-based method to enable users to make payments on the web, using a credit card, Apple Pay or Google Pay.

Notifications

The Notifications API enables web apps to display notifications, even when the app is not active.

Authentication

Web Authentication API (WebAuthn) enables passwordless authentication through your device's fingerprint reader or an external USB Security Key.

Audio recording

Record audio using MediaRecorder and visualize audio using Web Audio API.

NFC

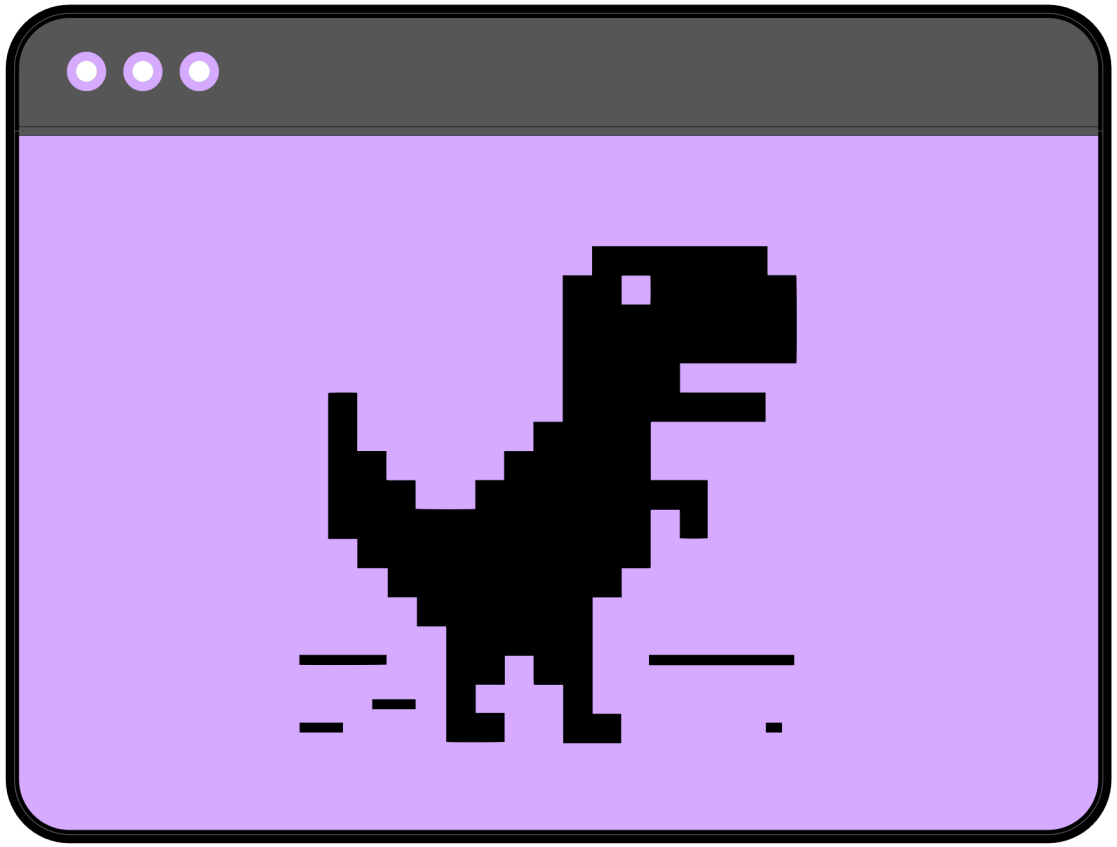
The Web NFC API enables web apps to read and write to NFC tags.

Wake lock

The Screen Wake Lock API enables web apps to prevent devices from dimming or locking the screen when the app needs to keep running.

https://whatpwacando.today/

Don’t forget offline

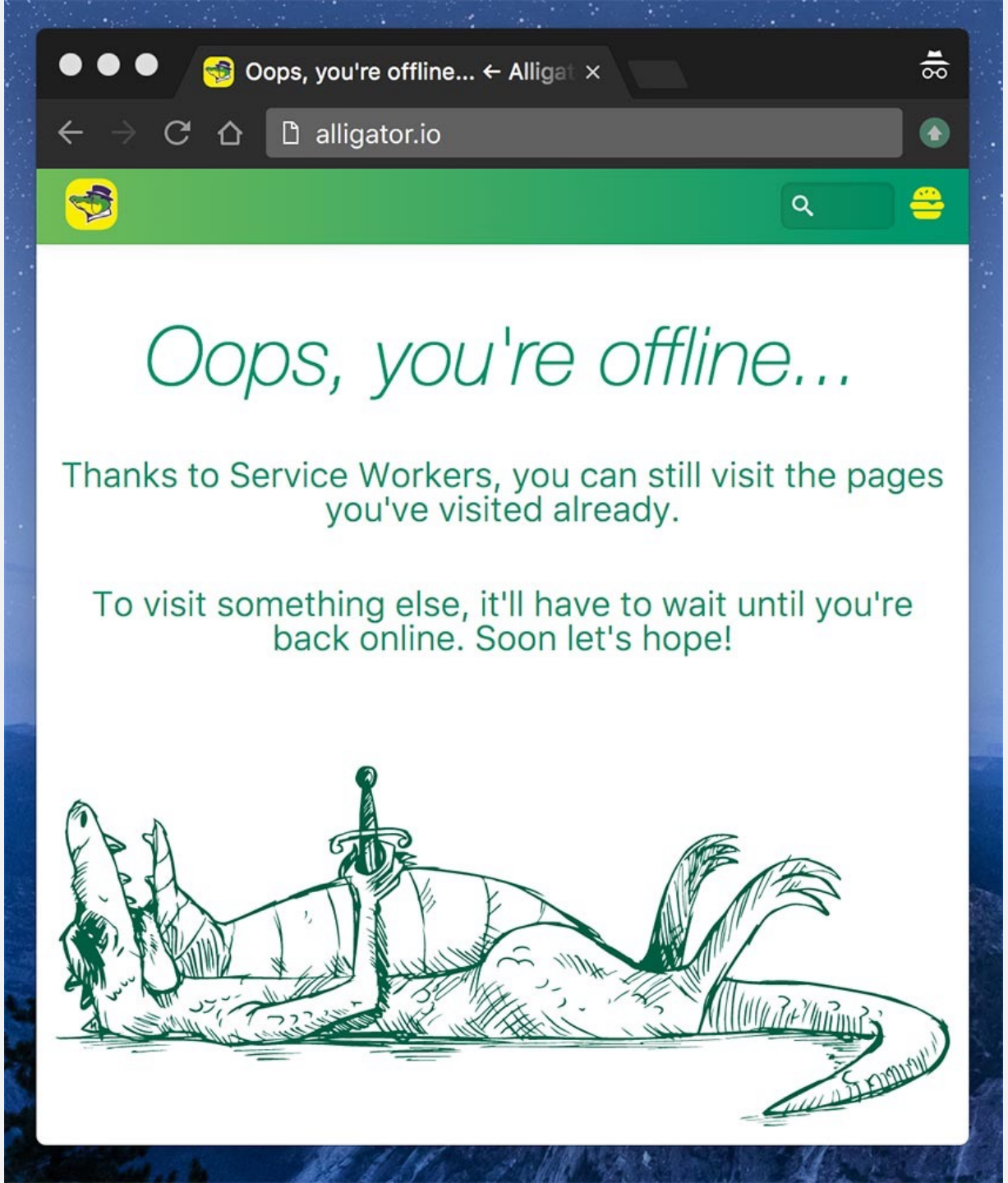


In 2017, Trivago saw 67% of users come back online to their PWA

https://aka.ms/trivago-pwa

Provide a custom offline page

https://aka.ms/Offline-PWA



Proactively cache

https://aka.ms/Offline-PWA

Content that remains unchanged for long intervals:

- Banner images
- Authentication state
- Media for playback

Service workers

Summary: Make interactions seamless and fast

Use skeleton UI for transition states when pages or screens are loading

Where possible, integrate with platform APIs and features to speed processes up e.g. web payments and autofill in a checkout user flow

Create an offline experience to keeps users engaged until their network connection is restored

https://github.com/ststimac/pwa-design-checklist

03 User experience considerations

“Of a very high standard; excellent.”

Thank you

https://github.com/ststimac/pwa-design-checklist

https://noti.st/seaotta

@seaotta

Resources

30 Days of PWA:
https://microsoft.github.io/win-student-devs/#/

Microsoft PWA Docs:
https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/

Google Getting started with Progressive Web Apps
https://developer.chrome.com/blog/getting-started-pwa/

What Web Can Do Today?
https://whatwebcando.today/

What PWA Can Do Today?
https://whatpwacando.today/