# Progressive Web Applications

# PWA

Progressive Web Apps (PWA) are web applications built and enhanced with modern APIs to deliver native-like capabilities, reliability, and installability while reaching *anyone, anywhere, on any device* with a single codebase.

# Famous PWAs

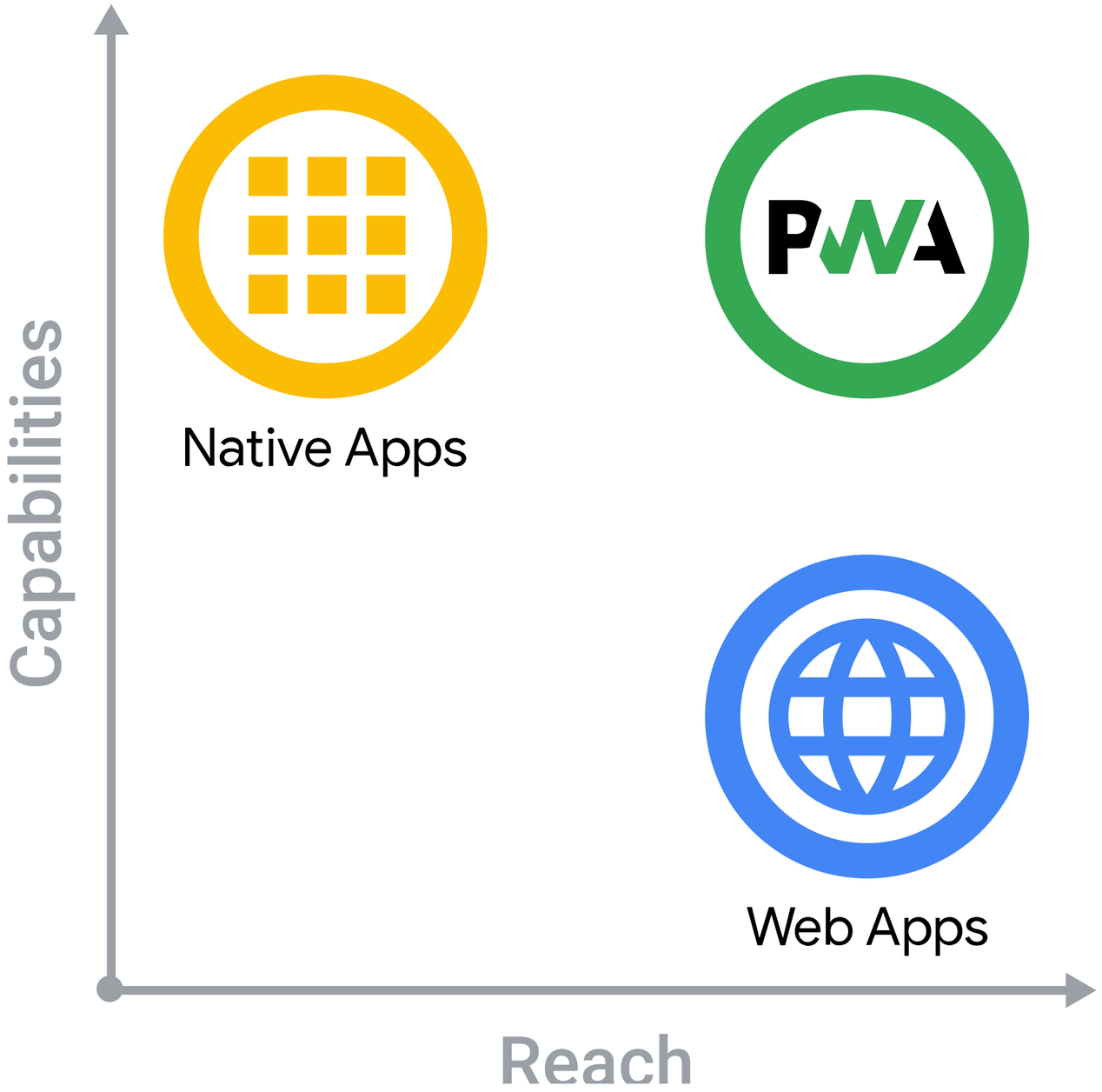
- Twitter
- Starbucks (x2 active users and on desktop)
- Flipboard
- Pinterest
- Instagram (only on Windows)
  - Developing countries with camera filters and offline support

# Web Applications

— Immersive experience

— Speed and fluidity

— Polish and personality


Capabilities
Native Apps
PWA
Web Apps
Reach

- Introduced in 2015
- 3 main pillars
    1. Capable
    2. Reliable
    3. Installable

# 1. Capable

— WebRTC, Geolocation, Push Notifications
— WebAssembly
— Filesystem, Camera API etc.
— Secure, permission model

## 2. Reliable

— Speed is essential
— Work regardless of network connection

## 3. Installable

— Launchable
— Handle links from other applications
— Default application for filetype

## Key features

1. Responsive
2. Connectivity-independent
3. Feel like a native app
4. Always up-to-date
5. Safe (HTTPS)
6. Discoverable
7. Linkable

# PWAs vs native apps

— Better experience on the website itself
— Increase chance of downloading the native app
— People are installing less applications
— Native capabilities

# Pro

— Lightweight compared to native apps
— More accessible via web URL
— No app/store or gatekeeper
— Updates OTA

# First movers

- Developing markets
  - Limited storage
  - Slow and unstable connection

# Use cases - Twitter

— + 65% pages per session

— + 75% tweets

— - 20% bounce rate

— - 97% size

# Use cases - Pinterest

Comparing old mobile web to new mobile web

| Time Spent > 5 minutes | User-generated Ad $ | Ad Clickthroughs | Core Engagements |
|---|---|---|---|
| +40% | +44% | +50% | +60% |

Comparing across web/native

| Time Spent > 5 minutes | User-generated Ad $ | Ad Clickthroughs | Core Engagements |
|---|---|---|---|
| +5% | +2% | +0% | +2-3% |

— PWA Stats

# Compared to other web solutions

PhoneGap
Electron
React Native

— Unibiquity

— Linkability

# Progressive enhancement

Create a baseline that works everywhere. Then enhance the experience on more capable devices.

# Writing our first PWA

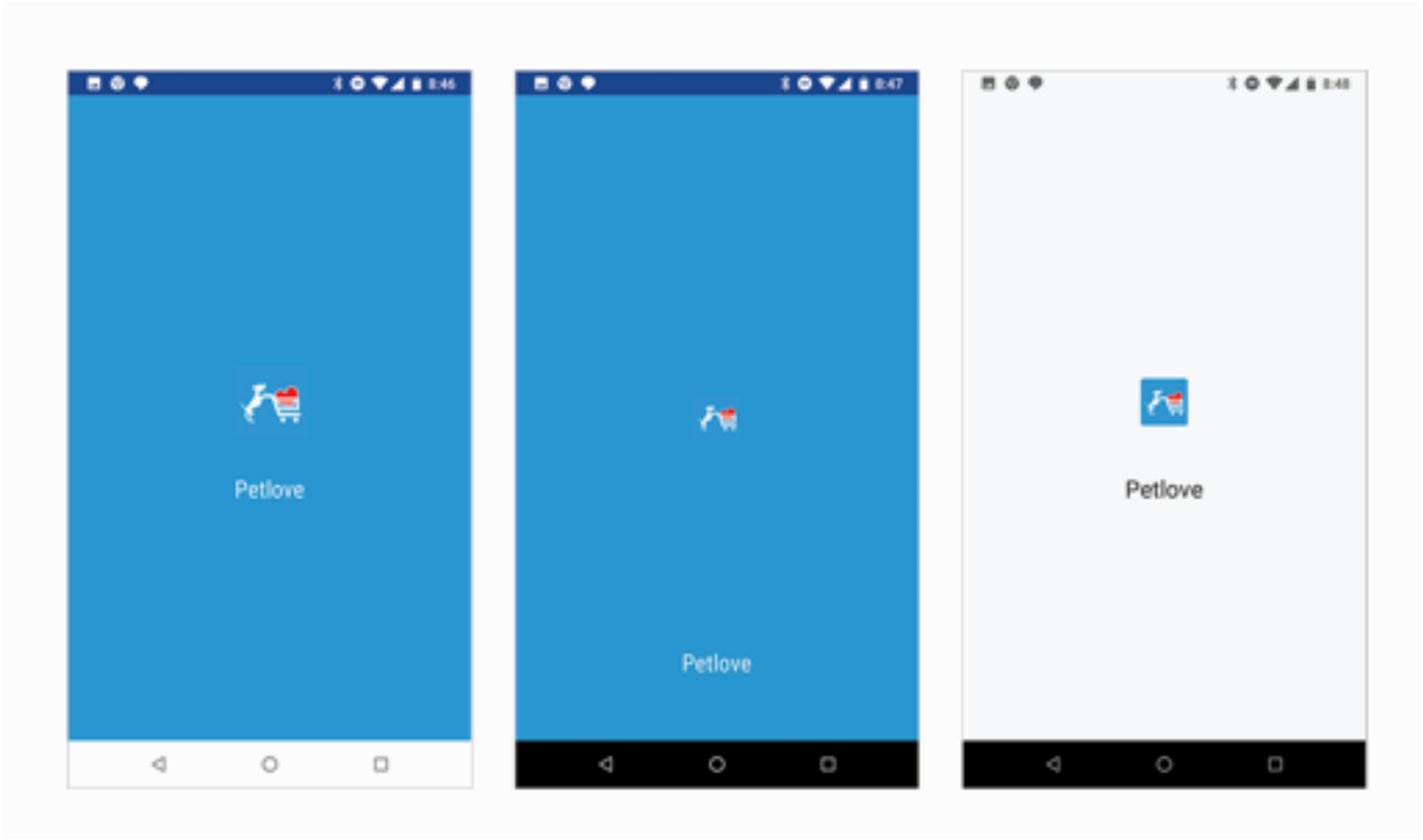
## Technically

1. HTTPS
   - Required however for APIs like Geolocation, Payment
2. Service worker
3. A manifest
   - app-manifest.firebaseapp.com/

# HTTPS

— Safety
— APIs like Request Payment, Geolocation
— Bult-in in GH Pages, Cloudflare, Netlify etc.

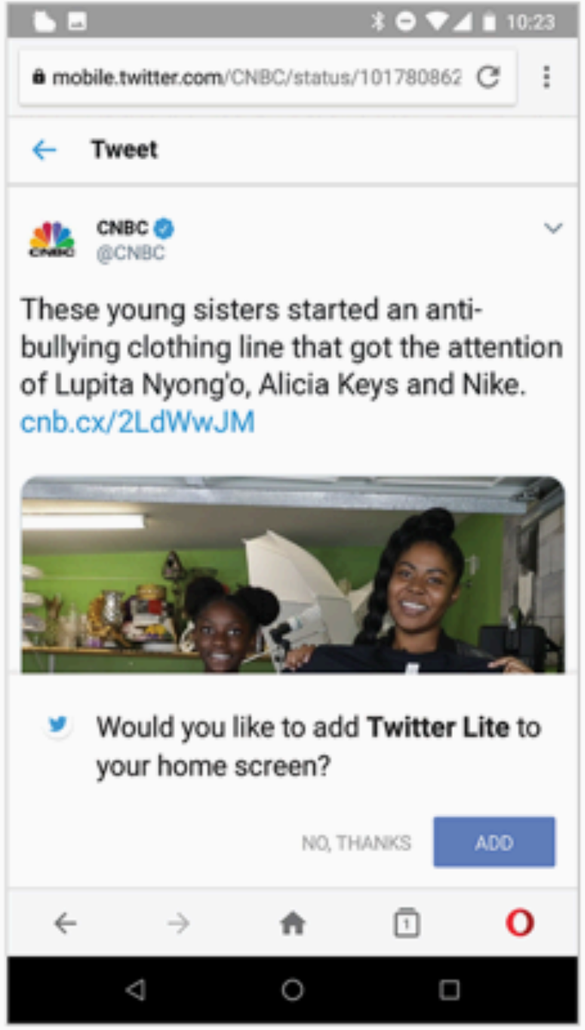
# background_color

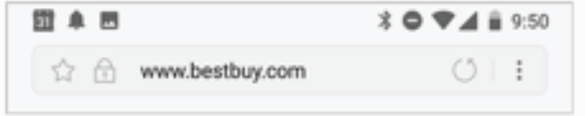


# background_color

# theme_color

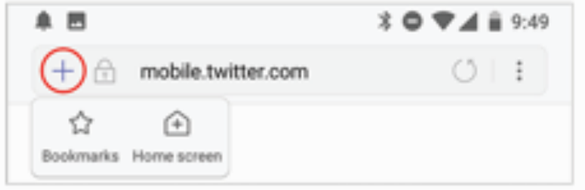

9:18
https://www.amazon.com
5
9:18
https://ride.lyft.com/?center=45.4681
4


Samsung's standard address bar


Firefox's standard address bar

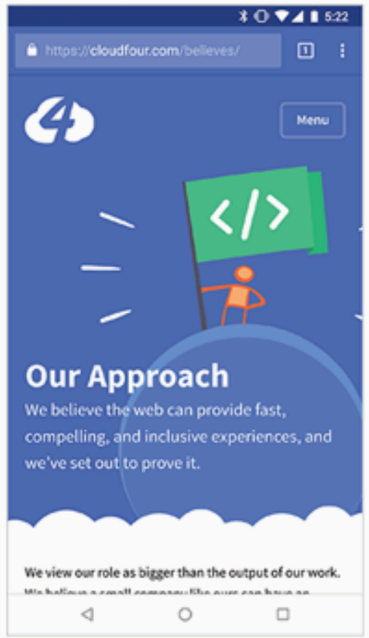



Samsung's add-to-homescreen badge

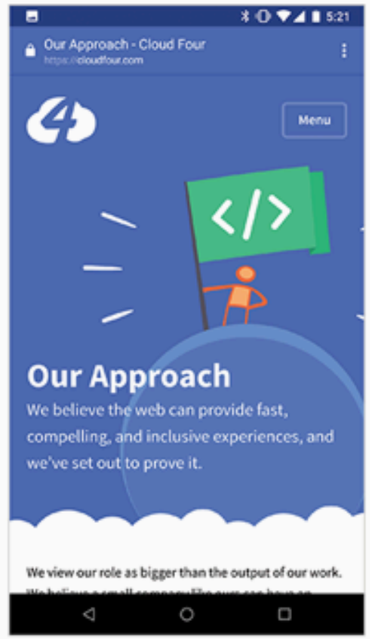

Firefox's add-to-homescreen badge

# display-mode

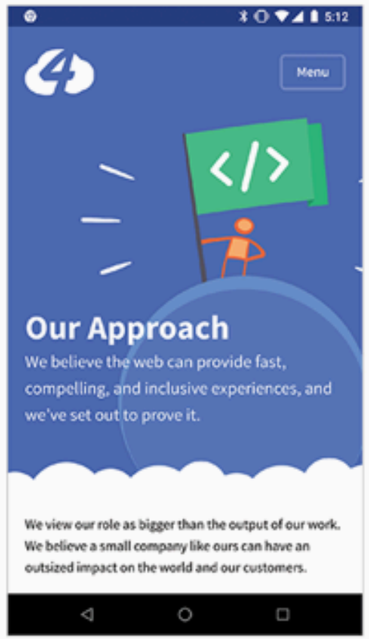
- browser
- minimal-ui
- standalone (!)
  - URL address bar
  - Share options
- fullscreen (!!)
  - Navigation bar



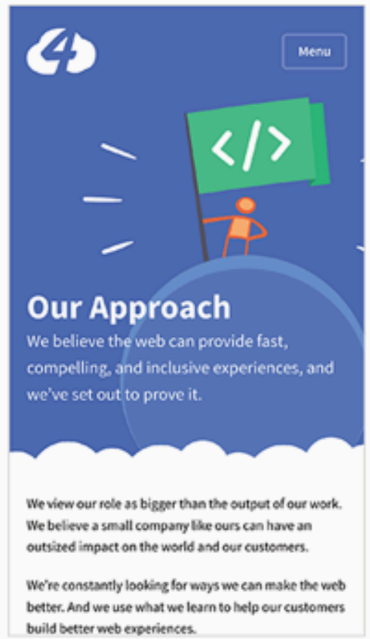

display-mode: browser




display-mode: minimal-ui

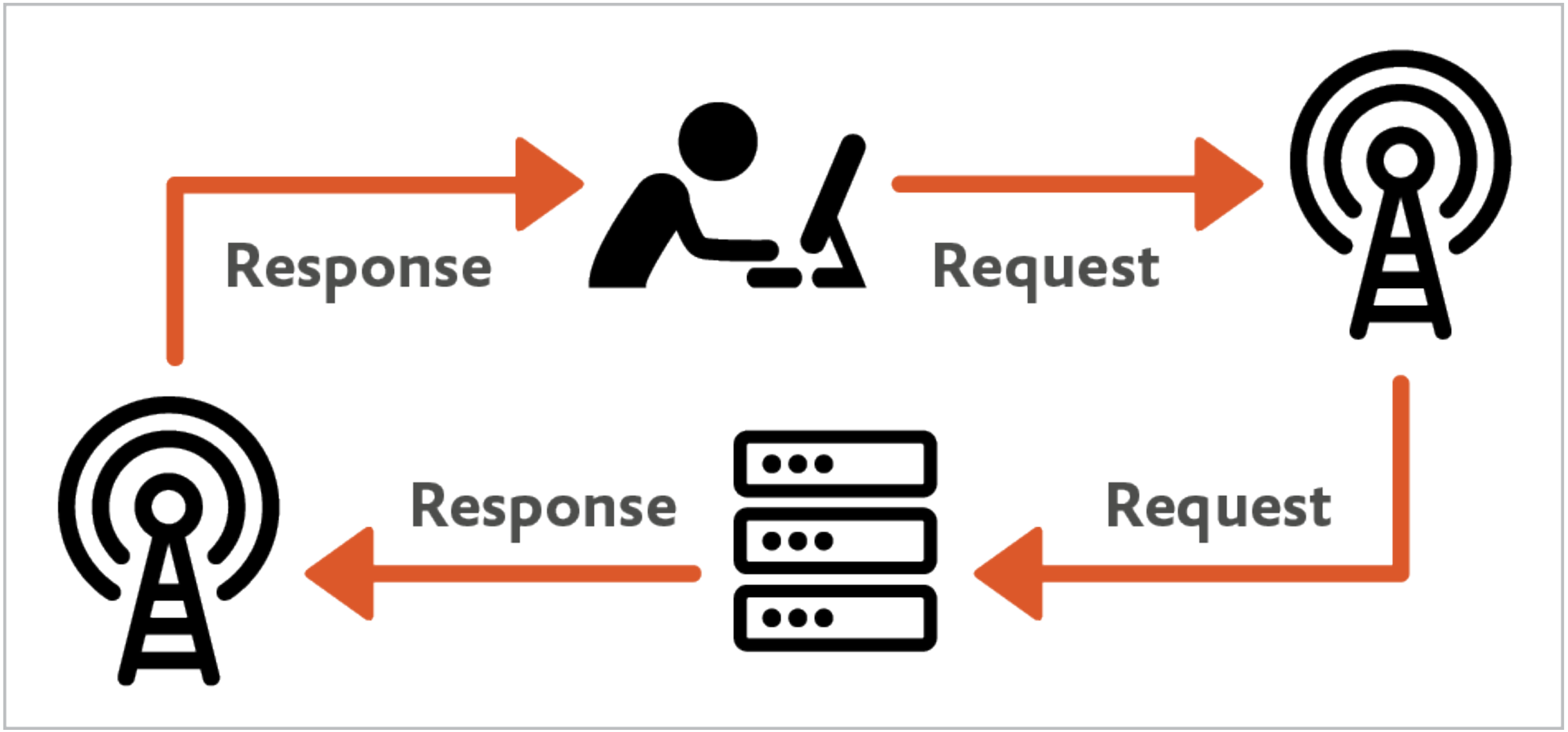



display-mode: standalone



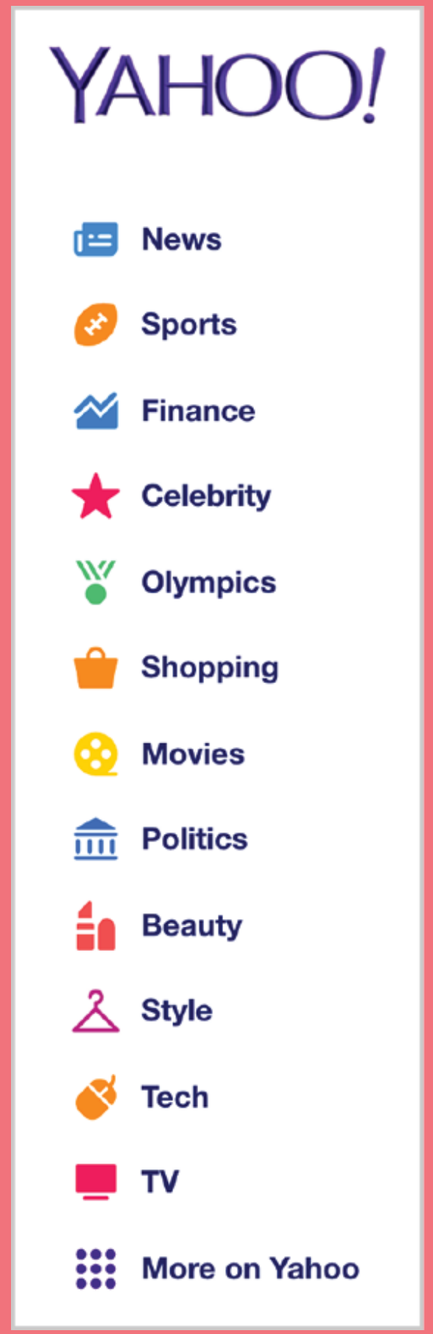

display-mode: fullscreen

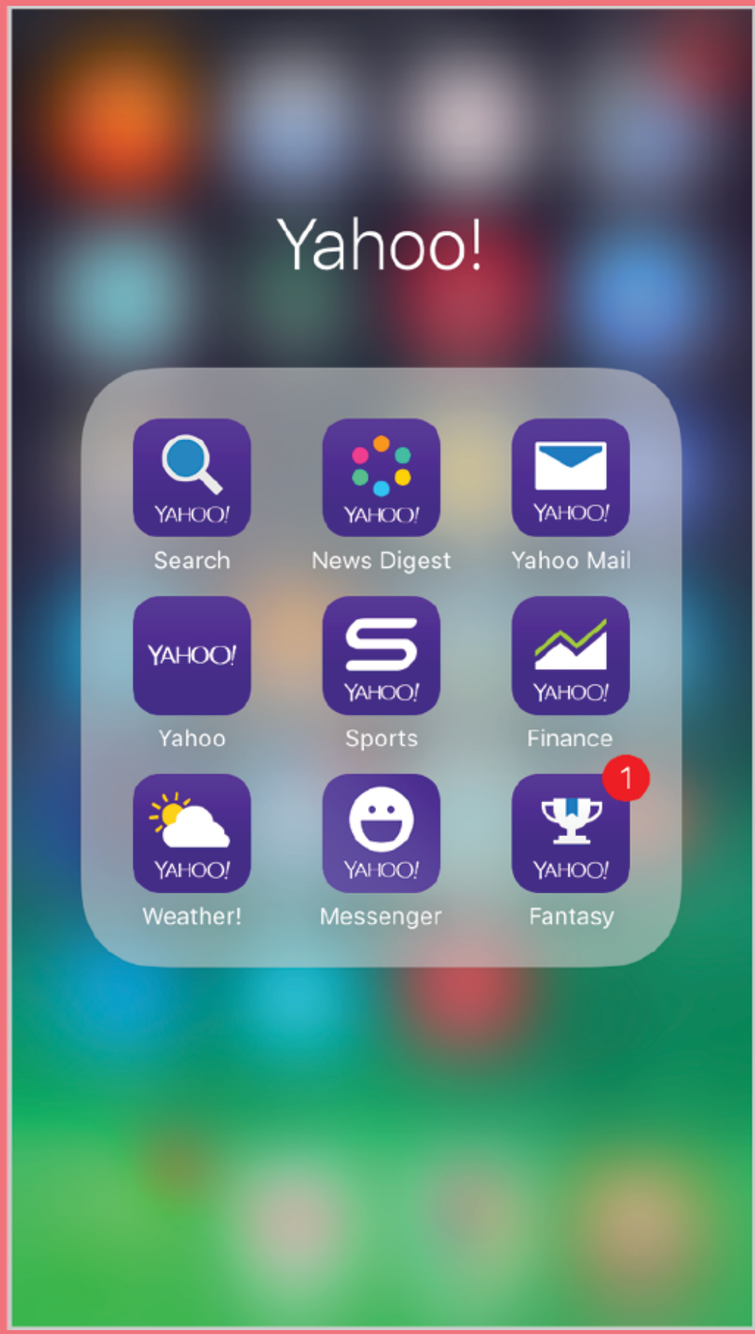
# Service worker


Response
Request
Response
Request

```
if ('serviceWorker' in navigator) {
  const swUrl = `/sw.js`;

  window.addEventListener('load', () => {
    navigator.serviceWorker
      .register(swUrl)
      .then(registration => {
        console.log('[ServiceWorker] registered', registration);

        registration.addEventListener('updatefound', () => {
          console.log('[ServiceWorker] update found');
        });
      })
      .catch(err => console.log('[ServiceWorker] registration failed: ', err));
  });
}
```

# SW registration scope



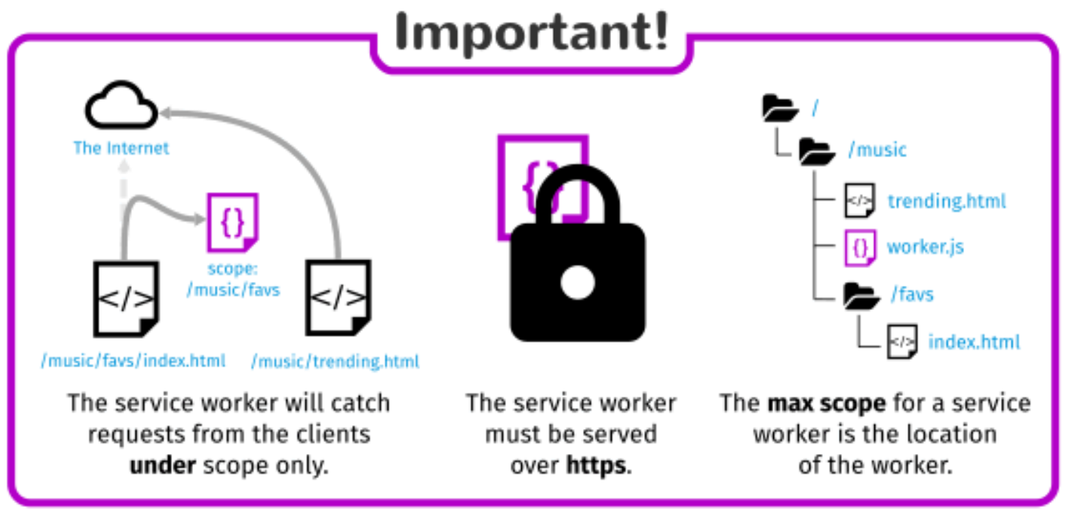
```
self.addEventListener('install', () => {
  console.log('install');
});

self.addEventListener('activate', () => {
  console.log('activate');
});
```

# Service Worker limitations

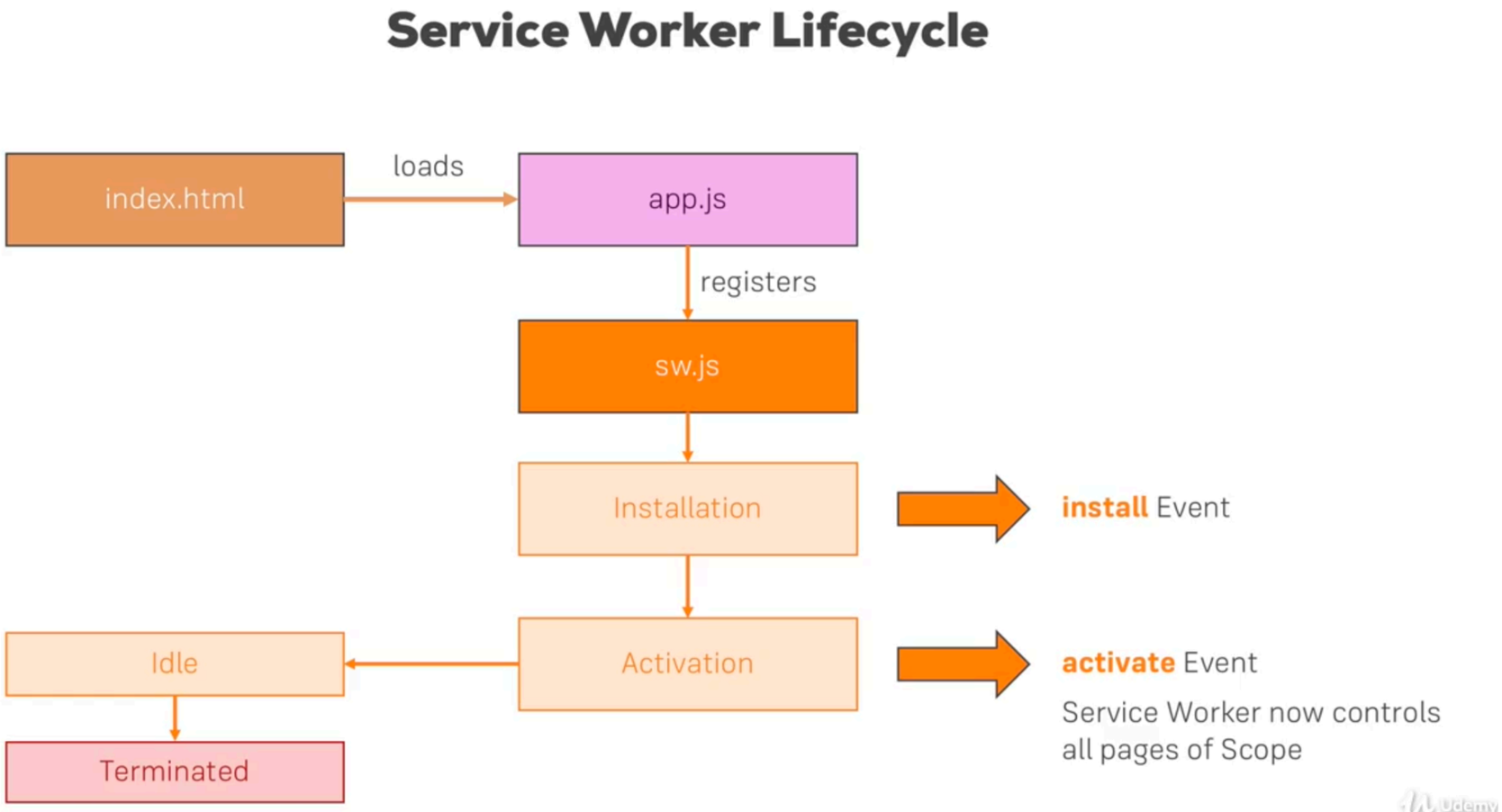
— No access to DOM

— Similar to `WebWorker`

— Must live on the same domain

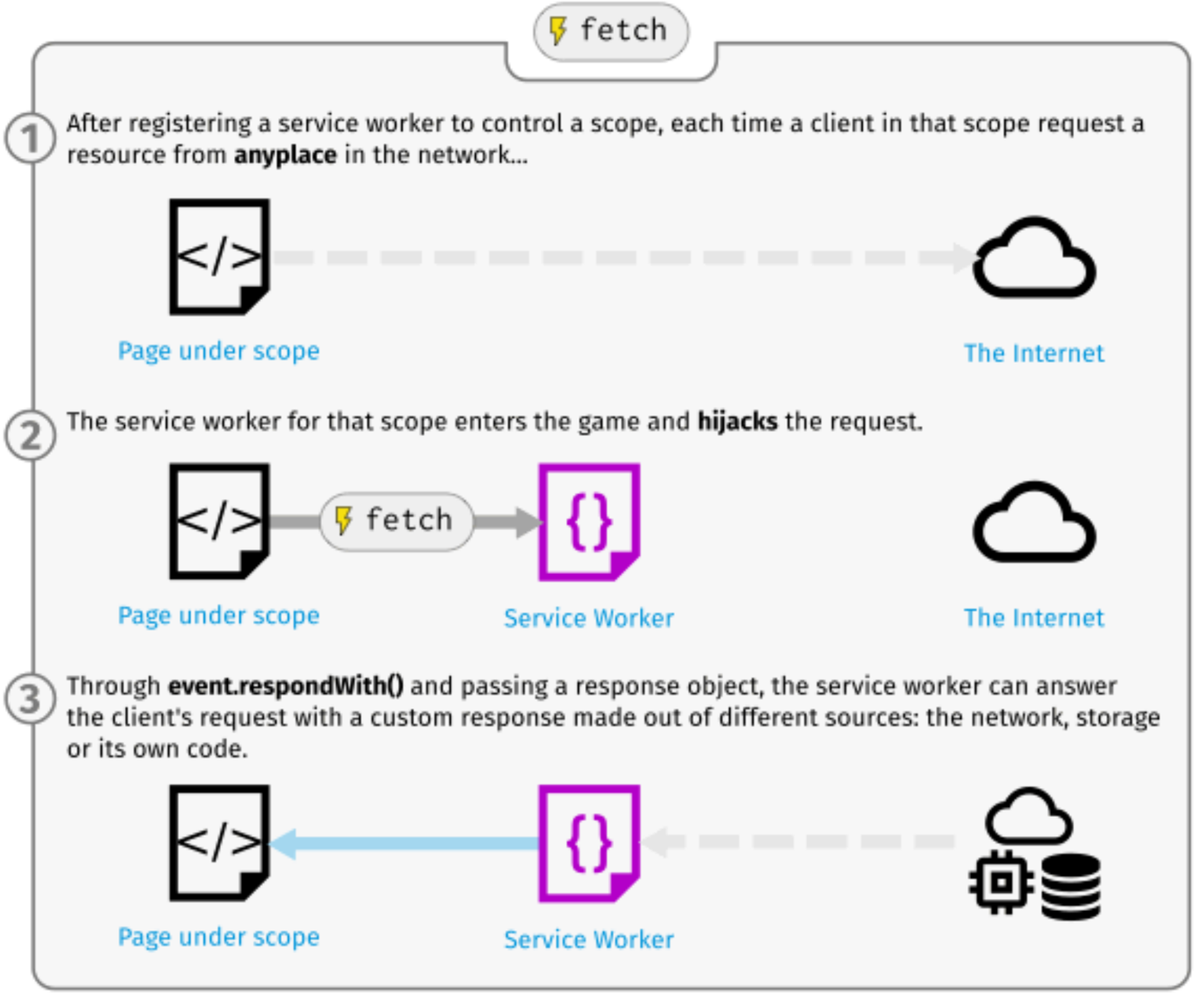


# SW lifecycle



# Intercepting fetch

```
self.addEventListener('fetch', event => {
  const request = event.request;

  event.respondWith(new Response('Hello world'));
});
```


fetch
1
After registering a service worker to control a scope, each time a client in that scope request a resource from **anyplace** in the network...
Page under scope
The Internet
2
The service worker for that scope enters the game and **hijacks** the request.
fetch
Page under scope
Service Worker
The Internet
3
Through **event.respondWith()** and passing a response object, the service worker can answer the client's request with a custom response made out of different sources: the network, storage or its own code.
Page under scope
Service Worker

# Add static files to cache

```
const CACHE_NAME = 'daisyhub-cache';

self.addEventListener('install', event => {
  console.log('install');
  event.waitUntil(
    caches.open(CACHE_NAME).then(cache => {
      return cache.addAll(['/static/media/image.png']);
    }),
  );
});
```

# Cache handling

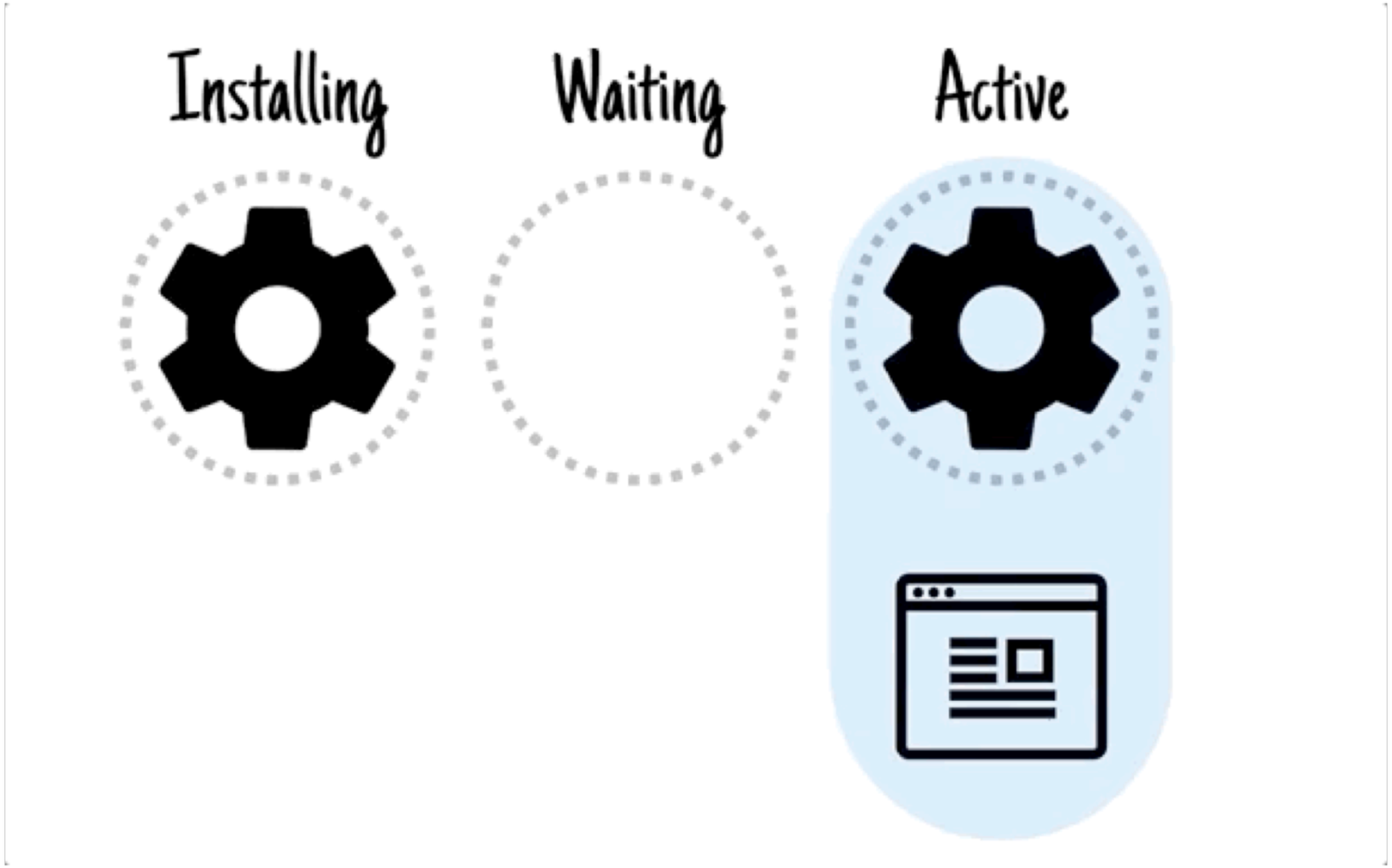
*There are only two hard things in Computer Science: cache invalidation and naming things.*

— Phil Karlton

# Cache falling back to network

```
self.addEventListener('fetch', event => {
  const request = event.request;

  if (request.method !== 'GET') return;

  event.respondWith(
    caches.match(request).then(response => {
      return response || fetch(request);
    }),
  );
});
```

# Service worker wait



# Avoid waiting for clients to close

```
self.skipWaiting();
self.clients.claim()
```

```
navigator.serviceWorker.register('/sw.js').then(reg => {
  reg.installing; // the installing worker, or undefined
  reg.waiting; // the waiting worker, or undefined
  reg.active; // the active worker, or undefined

  reg.addEventListener('updatefound', () => {
    const newWorker = reg.installing;

    newWorker.state;
    // "installing" - the install event has fired, but not yet complete
    // "installed"  - install complete
    // "activating" - the activate event has fired, but not yet complete
    // "activated"  - fully active
    // "redundant"  - discarded. Either failed install, or it's been
    //                replaced by a newer version

    newWorker.addEventListener('statechange', () => {
      // newWorker.state has changed
    });
  });
});

navigator.serviceWorker.addEventListener('controllerchange', () => {
  // A new worker has become the new active worker.
});
```

# Delete old cache

```
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(keyList => {
      return Promise.all(
        keyList.map(cacheKey => {
          if (cacheKey !== CACHE_NAME) return caches.delete(cacheKey);

          return Promise.resolve(null);
        }),
      );
    }),
  );
});
```

# Network with generic fallback

```
const urlsToCache = ['/offline.html'];

self.addEventListener('fetch', event => {
  const request = event.request;
  const url = new URL(request.url);

  if (url.pathname === '/') {
    return event.respondWith(
      fetch(request).catch(error => {
        return caches.match('/offline.html');
      }),
    );
  }
});
```

# Cache on network response

```
const request = event.request;
const accept = request.headers.get('Accept');

if (accept.includes('image')) {
  return event.respondWith(
    caches.match(request).then(response => {
      if (response) return response;

      return caches.open(CACHE_NAME).then(cache => {
        return fetch(request).then(response => {
          cache.put(request, response.clone());

          return response;
        });
      });
    }),
  );
}
```

# Stale-while-revalidate

```
const imagesRegxp = /(\.(png|jpeg|svg|ico))$/;
if (imagesRegxp.test(request.url)) {
  return event.respondWith(
    caches.match(request).then(response => {
      if (response) {
        event.waitUntil(
          caches.open(CACHE_NAME).then(cache => {
            return fetch(request).then(response => {
              return cache.put(request, response.clone());
            });
          }),
        );

        return response;
      }

      return caches.open(CACHE_NAME).then(cache => {
        return fetch(request).then(response => {
          cache.put(request, response.clone());

          return response;
        });
      });
    }),
  );
}
```

# Stale-while-revalidate

```
function staleWhileRevalidate(cacheName, request) {
  return caches.open(cacheName).then((cache) => {
    return cache.match(request).then((response) => {
      const fetchRequest = fetch(request).then((fetchResponse) => {
        cache.put(request, fetchResponse.clone());

        return fetchResponse;
      });

      return response || fetchRequest;
    });
  });
}
```

```
const CACHE_STATIC = 'daisyhub-static-v1';
const CACHE_IMAGES = 'daisyhub-images-v1';
const CACHE_NAMES = [CACHE_STATIC, CACHE_IMAGES];

const staticUrlsToCache = ['/offline.html'];
```

# Network falling back to the cache

```
const CACHE_PAGES = 'daisyhub-pages-v1'
const CACHE_NAMES = [CACHE_STATIC, CACHE_IMAGES, CACHE_PAGES];

const isHome = url.pathname === "/";
if (isHome) {
  return event.respondWith(
    caches.open(CACHE_PAGES).then((cache) => {
      return fetch(request)
        .then((response) => {
          event.waitUntil(cache.put(request, response.clone()));

          return response;
        })
        .catch(() => {
          return cache
            .match(request)
            .then((response) => response || caches.match("/offline.html"));
        });
    })
  );
}
```

# networkFallbackCache

```
function networkFallbackCache(cacheName, request) {
  return caches.open(cacheName).then((cache) => {
    return fetch(request)
      .then((response) => {
        cache.put(request, response.clone());

        return response;
      })
      .catch((error) => {
        console.log("Failed to fetch", request.url, error);

        return cache.match(request).then((response) => {
          if (!response) return Promise.reject(null);

          return response;
        });
      });
  });
}
```

# Other HTML pages: Stale-while-revalidate

```
const isHTML = request.mode === 'navigate'

if (isHTML) {
  return event.respondWith(
    staleWhileRevalidate(CACHE_PAGES, request).catch((error) => {
      console.log("Failed to fetch", request.url, error);
      return caches.match("/offline.html");
    })
  );
}
```

# The building blocks of Caching

1. Filter the request
2. Look for the file in the cache
3. Fetch from the network
4. Put copies into the cache

# More caching strategies

# Cache-only

```
self.addEventListener('fetch', (event) => {
  event.respondWith(caches.match(event.request));
});
```

— Good for files in cache at install

# Network-only

```
self.addEventListener('fetch', (event) => {
  event.respondWith(fetch(event.request));
});
```

— For not GET requests, analytics etc.

# Cache then network

In the UI:

```
const networkDataReceived = false;

const fetchRequest = fetch('/data.json')
  .then((response) => response.json())
  .then((data) => {
    networkDataReceived = true;
    updatePage(data);
  });

caches.match('/data.json')
  .then((response) => response.json()).then((data) => {
    if (!networkDataReceived) updatePage(data)
  })
```

# Cache then network

In the Service Worker:

```
self.addEventListener('fetch', (event) => {
  event.respondWith(
    caches.open(CACHE_DATA).then((cache)  =>{
      return fetch(event.request).then((response) => {
        cache.put(event.request, response.clone());

        return response;
      });
    })
  );
});
```

# Quiz time

forms.gle/
MgvVxE51zYm6aipZ6

# Cleaning the cache

```
navigator.storage.estimate().then(({usage, quota}) => {
  console.log(`Using ${usage} out of ${quota} bytes.`);
});
```

```
function trimCache(cacheName, limit) {
  return caches.open(cacheName).then((cache) => {
    return cache.keys().then((requests) => {
      return Promise.all(
        requests.slice(0, limit).map((request) => cache.delete(request))
      );
    });
  });
}
```

# Cleaning the cache

```
if (navigator.serviceWorker.controller) {
  navigator.serviceWorker.controller.postMessage({
    type: "TRIM_CACHE",
  });
}


self.addEventListener("message", (message) => {
  const data = message.data;

  switch (data.type) {
    case "TRIM_CACHE":
      trimCache(CACHE_IMAGES, 50);
      break;
    default:
      break;
  }
});
```

# Save for offline

```
const url = window.location.href;

button.addEventListener('click', () => {
  caches.open(CACHE_PAGES).then((cache) => cache.add(url));
})
```

# Save for offline

```
const url = window.location.href;
const saved = JSON.parse(localStorage.getItem(STORAGE_KEY)) || [];

caches.open(CACHE_PAGES).then((cache) => {
  cache.add(url).then(() => {
    const metadata = {
      url: url,
      title: document.querySelector(".article__title").textContent.trim(),
      subtitle: document
        .querySelector(".article__subtitle")
        .textContent.trim(),
    };

    localStorage.setItem(STORAGE_KEY, JSON.stringify([...saved, metadata]));
  });
});
```

```
const list = document.querySelector(".offline-articles");

caches.open(CACHE_PAGES).then((cache) => {
  cache.keys().then((requests) => {
    requests.forEach((request) => {
      const metadata = saved.find((x) => x.url === request.url);

      if (!metadata) return;

      const listItem = document.createElement("li");
        listItem.innerHTML = `
        <h3><a href="${metadata.url}">${metadata.title}</a></h3>
        <p>${metadata.subtitle}</p>
      `;

      list.appendChild(listItem);
    });
  });
});
```

9:14

Work & Careers

Saved to myFT

Leadership + myFT

# How brain science found its way into business school

Techniques to improve productivity, influence decision-making and handle stress move into core curricula

© Stephen S. Reardon

Thomas Bonfiglio, regional director of American Medical Response in New York State, finds there are benefits to meditation © Stephen S. Reardon

17 HOURS AGO by **Seb Murray**

myFT Updated 9:11pm

9:15

myFT

**Following**
Only available online

Business School Insider

US Interest Rates

Edit

**Saved articles**
Latest 50 available offline

How brain science found its way into business school

What MBA students want — and what they need

Powell hints at faster pace of rate rises

Done

**Recommended reads**

myFT Updated 9:11pm

# Adding to the cache safely

```
event.waitUntil(
  caches.open(CACHE_STATIC).then((cache) => {
    return Promise.all(
      staticUrlsToCache.map((url) =>
        cache.add(url).catch((error) => {
          console.log("Failed to add", url, "to the cache", error);
        })
      )
    );
  })
);
```

# Debugging Service Workers

- chrome://inspect/#service-workers
- chrome://serviceworker-internals
- Devtools console
- Devtools source breakpoints

# Auditing - Lighthouse

— web.dev/measure/

# Workbox

Google Developers

## Installing is not the goal

— Most people don't install the application or reuse it

— PWA is enabled even if not installed

## Choosing the right moment

— Not deterministic

— Save and reuse `beforeinstallprompt` event

## Google Chrome criteria

1. Meets user engagement **heuristic**
2. Manifest with
    1. `name/short_name`
    2. `icons` (at least 192x192)
    3. `start_url`
    4. `display: 'fullscreen' | 'standalone' | 'minimal-ui'`
3. Service worker with `fetch` handler

# Save the prompt

```
let deferredPrompt;

window.addEventListener('beforeinstallprompt', (e) => {
  e.preventDefault(); // Hides banner
  deferredPrompt = e;
  console.log('Saved install prompt for later');
  showInstallPromotion();
});
```

# Use the prompt event

```
buttonInstall.addEventListener('click', (e) => {
  hideMyInstallPromotion();
  deferredPrompt.prompt();
  deferredPrompt.userChoice.then((choiceResult) => {
    if (choiceResult.outcome === 'accepted') {
      console.log('User accepted the install prompt');
    } else {
      console.log('User dismissed the install prompt');
    }
  })
});
```

# Trigger install prompt event

— chrome://flags/#bypass-app-banner-engagement-checks

— Lighthouse Audit for PWA

**Bypass user engagement checks**

Bypasses user engagement checks for displaying app banners, such as requiring that users have visited the site before and that the banner hasn't been shown recently. This allows developers to test that other eligibility requirements for showing app banners, such as having a manifest, are met. – Mac, Windows, Linux, Chrome OS, Android

#bypass-app-banner-engagement-checks

Enabled

```
window.addEventListener('appinstalled', (event) => {
  console.log('The app is installed');
});
```

```
@media (display-mode: fullscreen) { /* Safari 13 */
  .back-btn {
    display: inline;
  }
}

window.addEventListener('load', () => {
  if (matchMedia('(display-mode: fullscreen)').matches) {
    console.log('The app is installed');
  } else {
    console.log('The app is in the browser');
  }
});
```

# App Stores

— Microsoft Store
— Google Play Store via Trusted Web Activity
— ~~Apple App Store~~

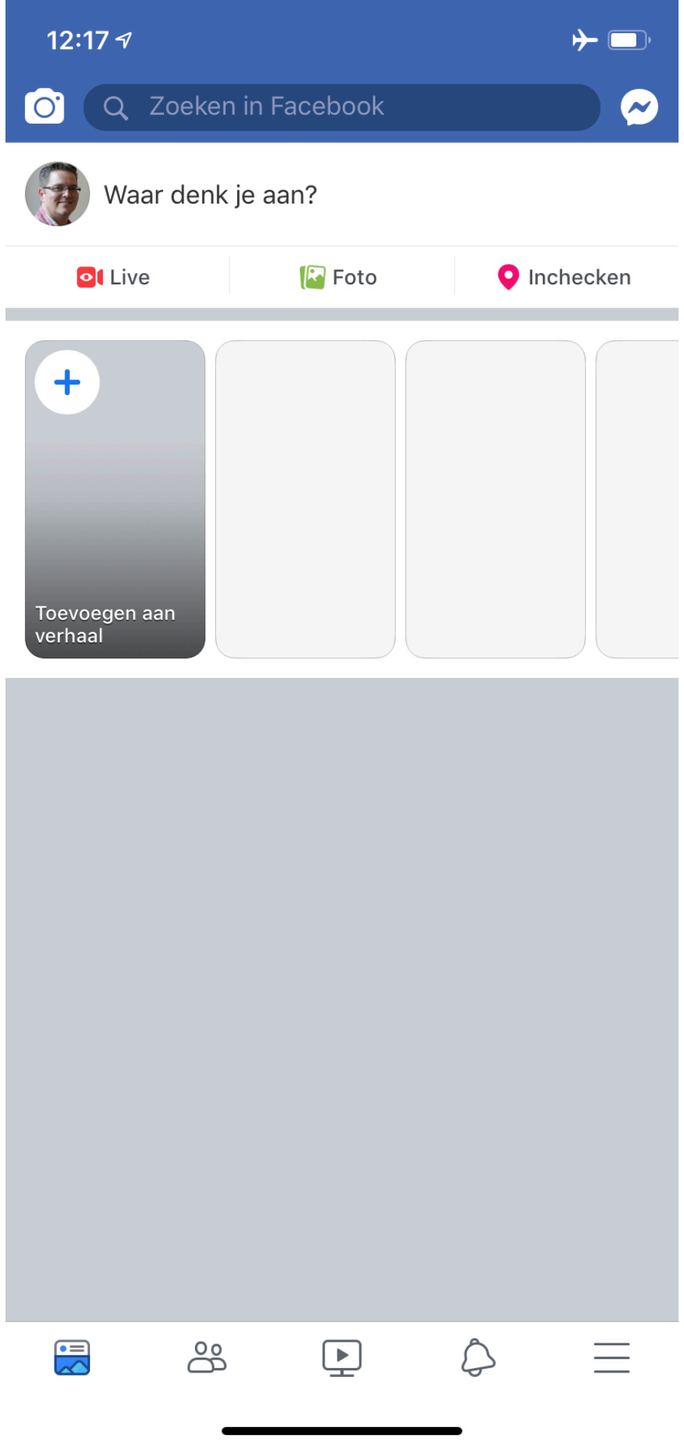
Be aware of stricter standards.

# Design

- Native UI
  - Difficult
  - Must be uncanny
  - Loose of brand
- Page transitions
- Immersive

# App Shell

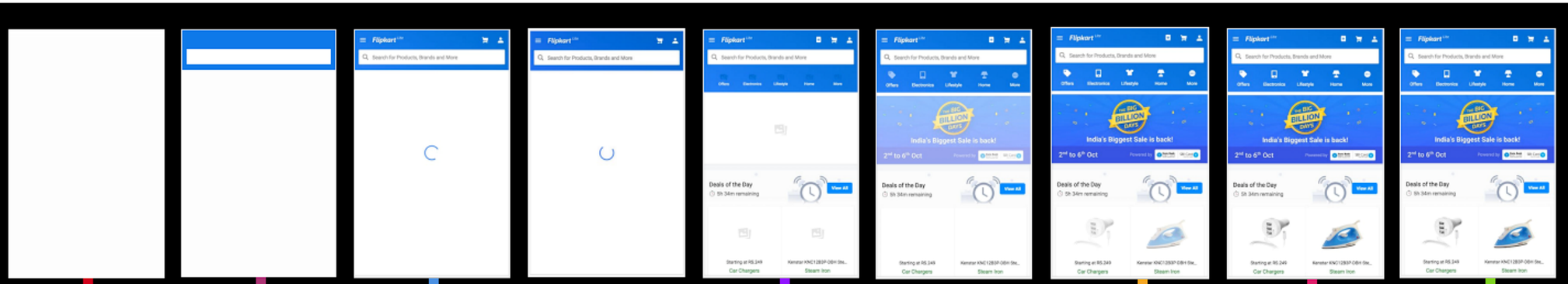
— The shell is the set of UI wrapping the content
— Progressive Web Apps with React
— Benefits even for not PWAs (*Washington Post*)


12:17
Zoeken in Facebook
Waar denk je aan?
Live
Foto
Inchecken
Toevoegen aan verhaal

## is it happening?

## is it useful?

## is it usable?



**Navigation begins**

Time to first byte

**First Paint**

The first non-blank paint on screen

**First Contentful Paint**

Navigation has successfully started

**First Meaningful Paint**

Page's primary content is visible

**Visually ready**

Page looks nearly done

**Time to Interactive**

Visually usable and engagable

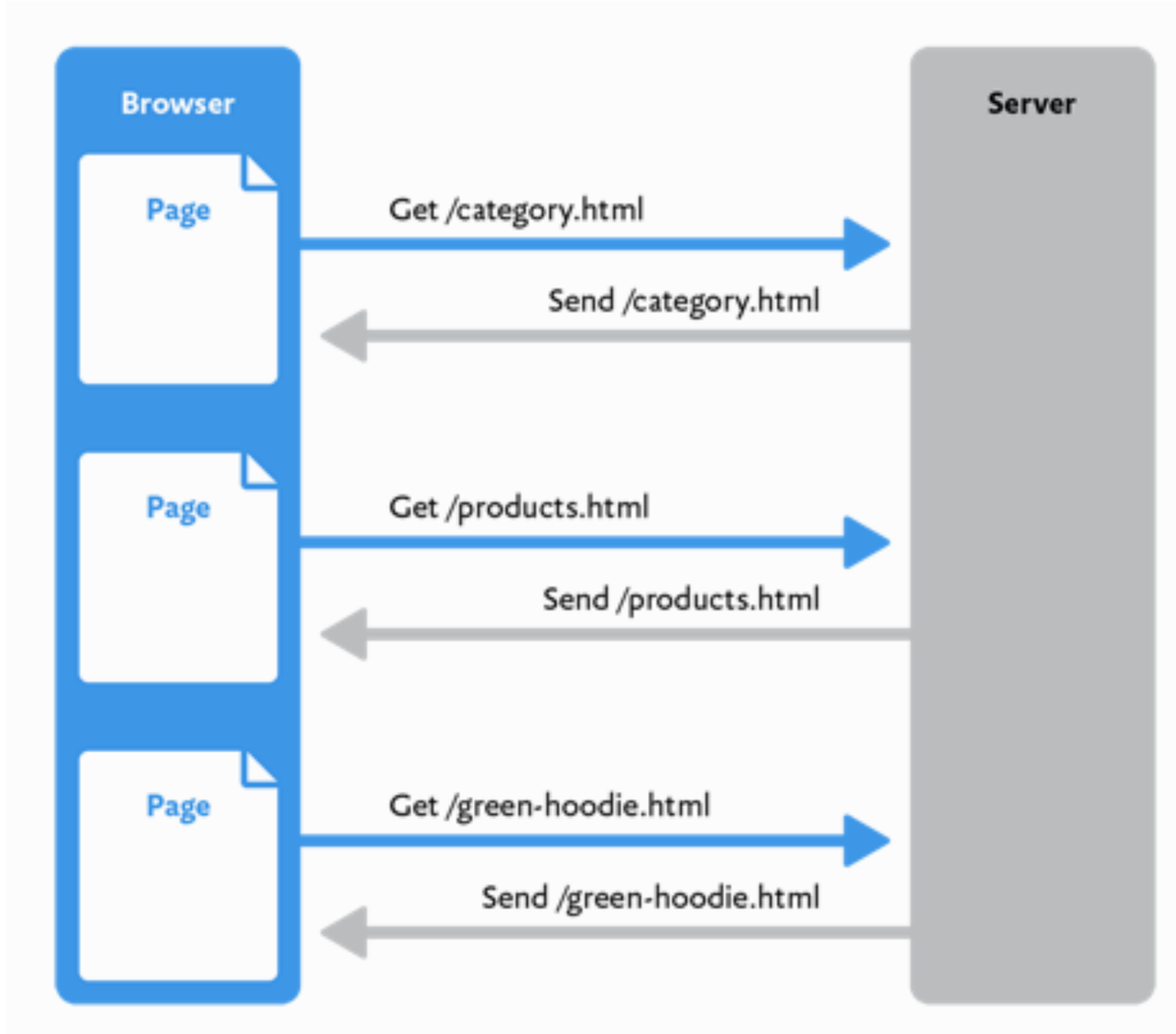
**Fully Loaded**

End of load lifecycle

@addyosmani

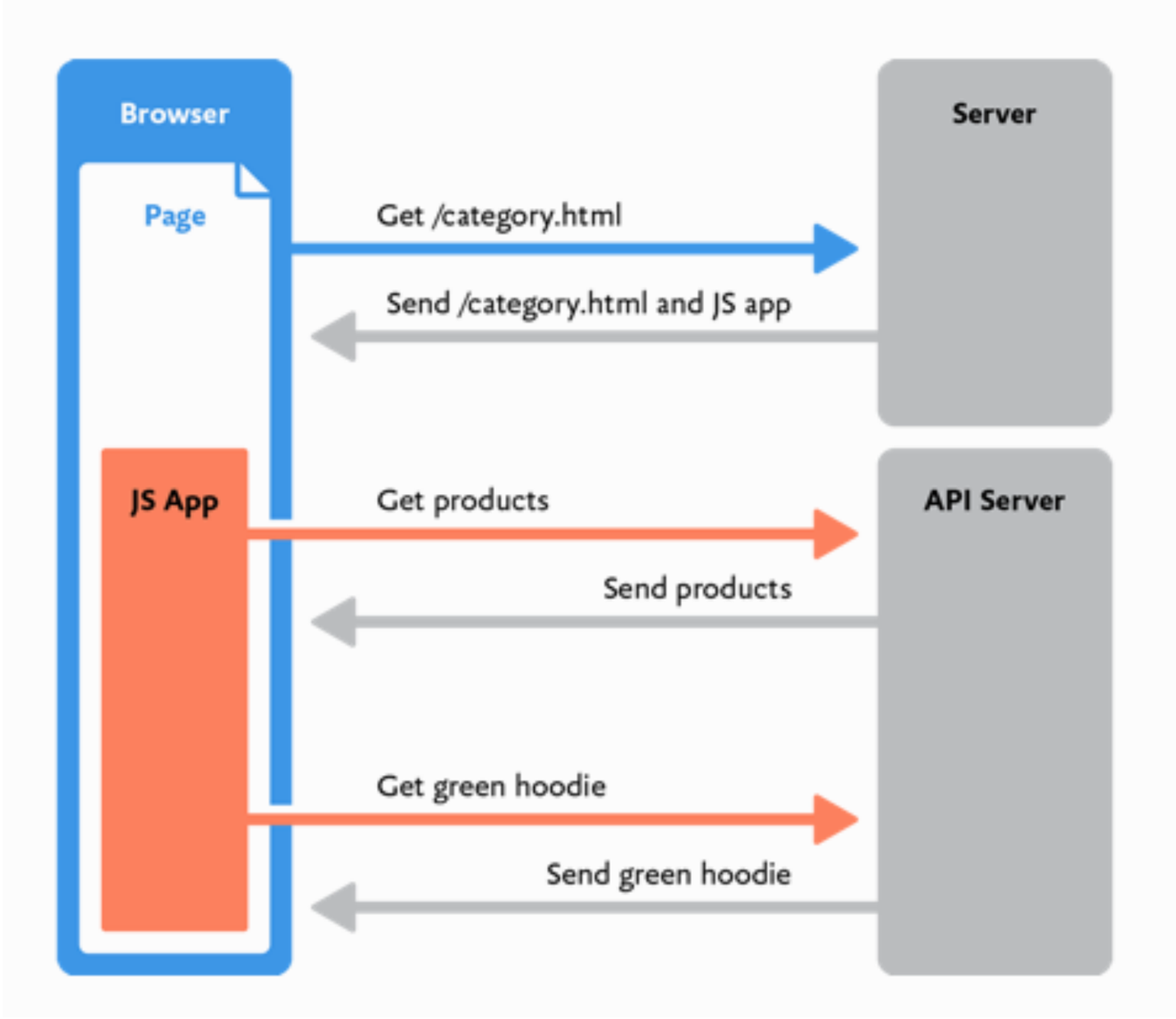
# Single Page Applications

— One .html per page



# Single Page Applications

— Only one .html forever



# PRPL Pattern

- Preload (Push)
  - `<link rel="preload" as="style" href="css/main.css">`
- Render
  - Critical CSS, SSR, Route splitting
- Pre-cache
- Lazy

# App Shell is optional

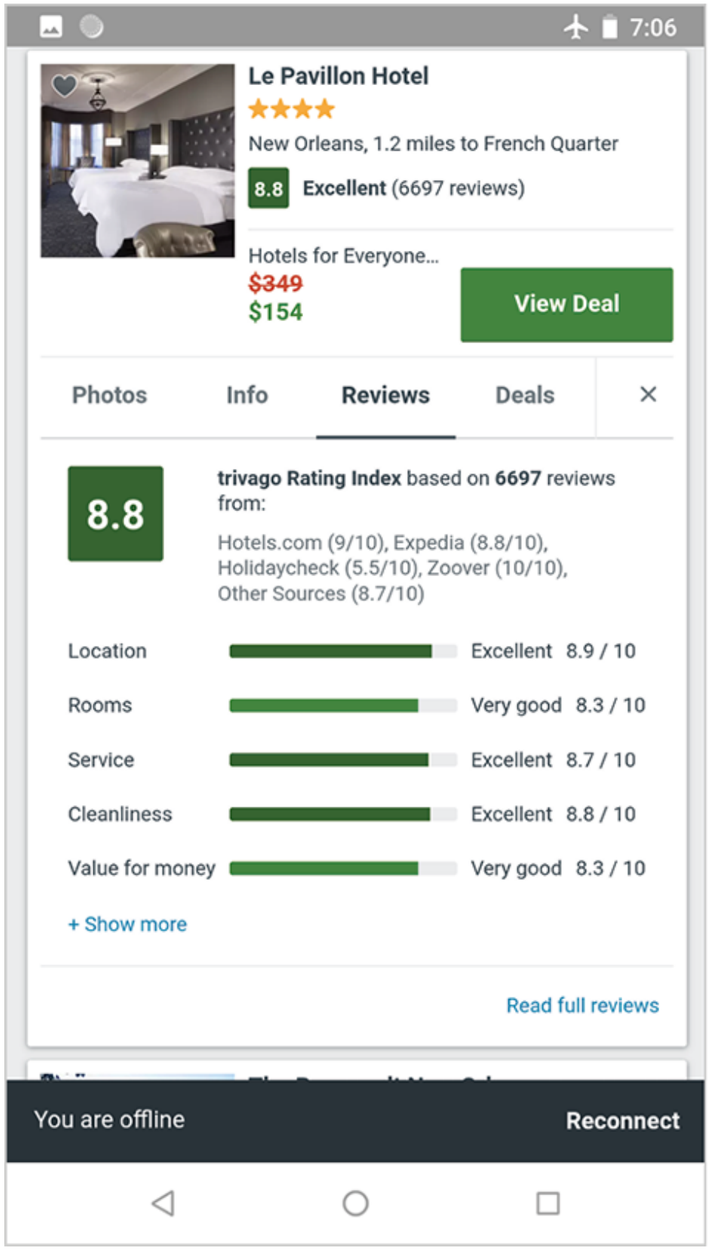
— You can have PWA in CMS like Drupal w/o shell

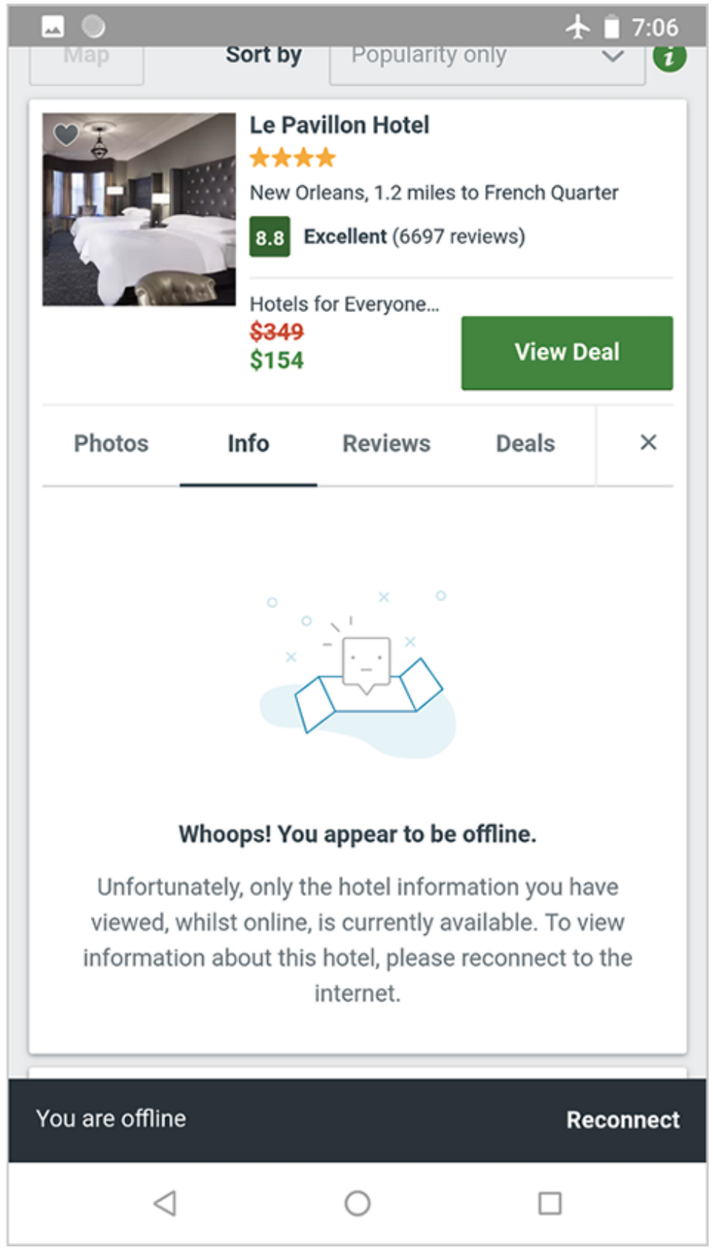
— Turbolink

# Offline support

— Trip details
— Music/podcast playing
— Recent content in socials

- Provide a offline notification
  - `navigator.onLine` / `BackgroundSync`


7:06
Le Pavillon Hotel
New Orleans, 1.2 miles to French Quarter
8.8 Excellent (6697 reviews)
Hotels for Everyone...
$349
$154
View Deal
Photos
Info
Reviews
Deals
8.8
trivago Rating Index based on 6697 reviews from:
Hotels.com (9/10), Expedia (8.8/10), Holidaycheck (5.5/10), Zoover (10/10), Other Sources (8.7/10)
Location Excellent 8.9 / 10
Rooms Very good 8.3 / 10
Service Excellent 8.7 / 10
Cleanliness Excellent 8.8 / 10
Value for money Very good 8.3 / 10
+ Show more
Read full reviews
You are offline
Reconnect


7:06
Map
Sort by
Popularity only
Le Pavillon Hotel
New Orleans, 1.2 miles to French Quarter
8.8 Excellent (6697 reviews)
Hotels for Everyone...
$349
$154
View Deal
Photos
Info
Reviews
Deals
Whoops! You appear to be offline.
Unfortunately, only the hotel information you have viewed, whilst online, is currently available. To view information about this hotel, please reconnect to the internet.
You are offline
Reconnect

# Saving offline changes

- Use `localStorage/sessionStorage` or `IndexedDB`
- `BackgroundSync`
- Inform the user of the network status

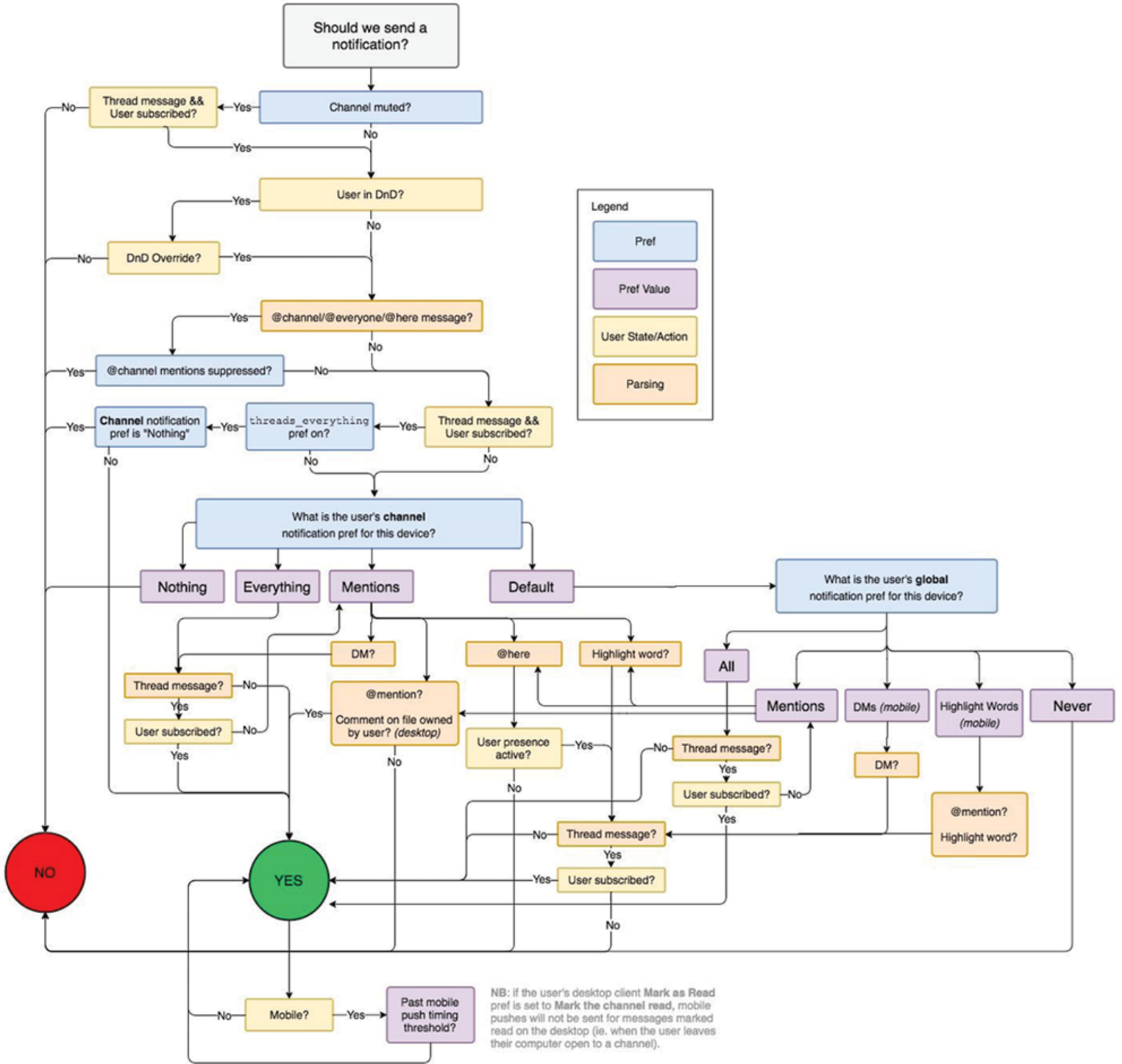
# Push Notifications

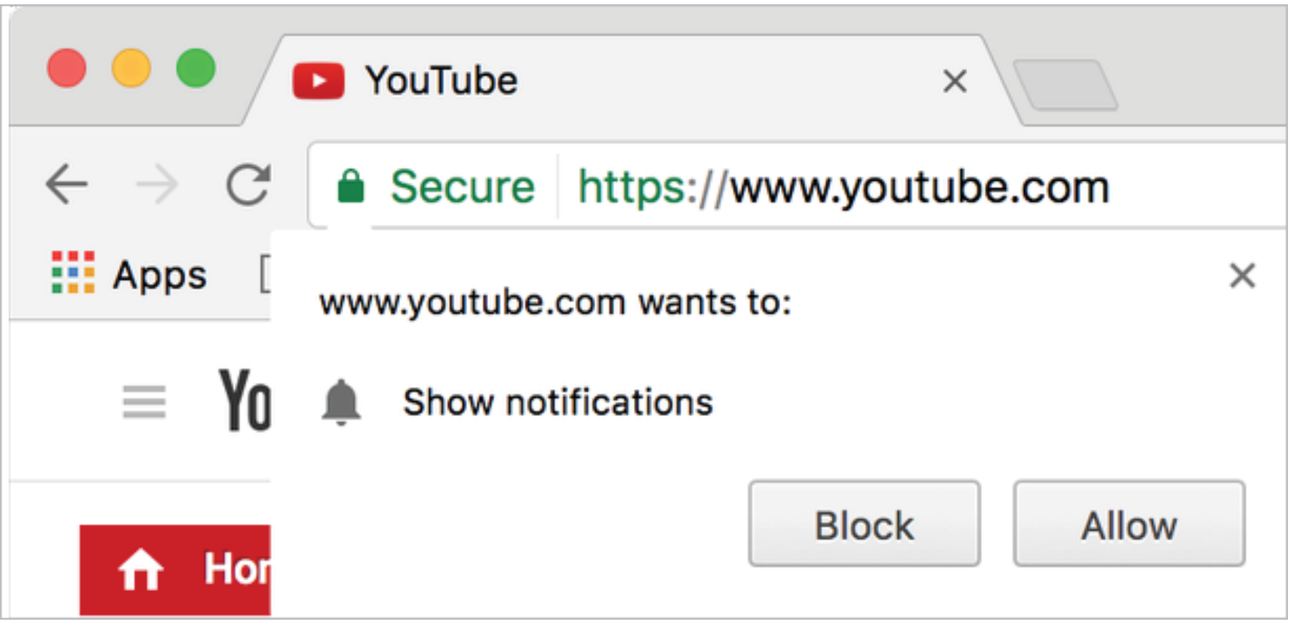
People don't like notifications.

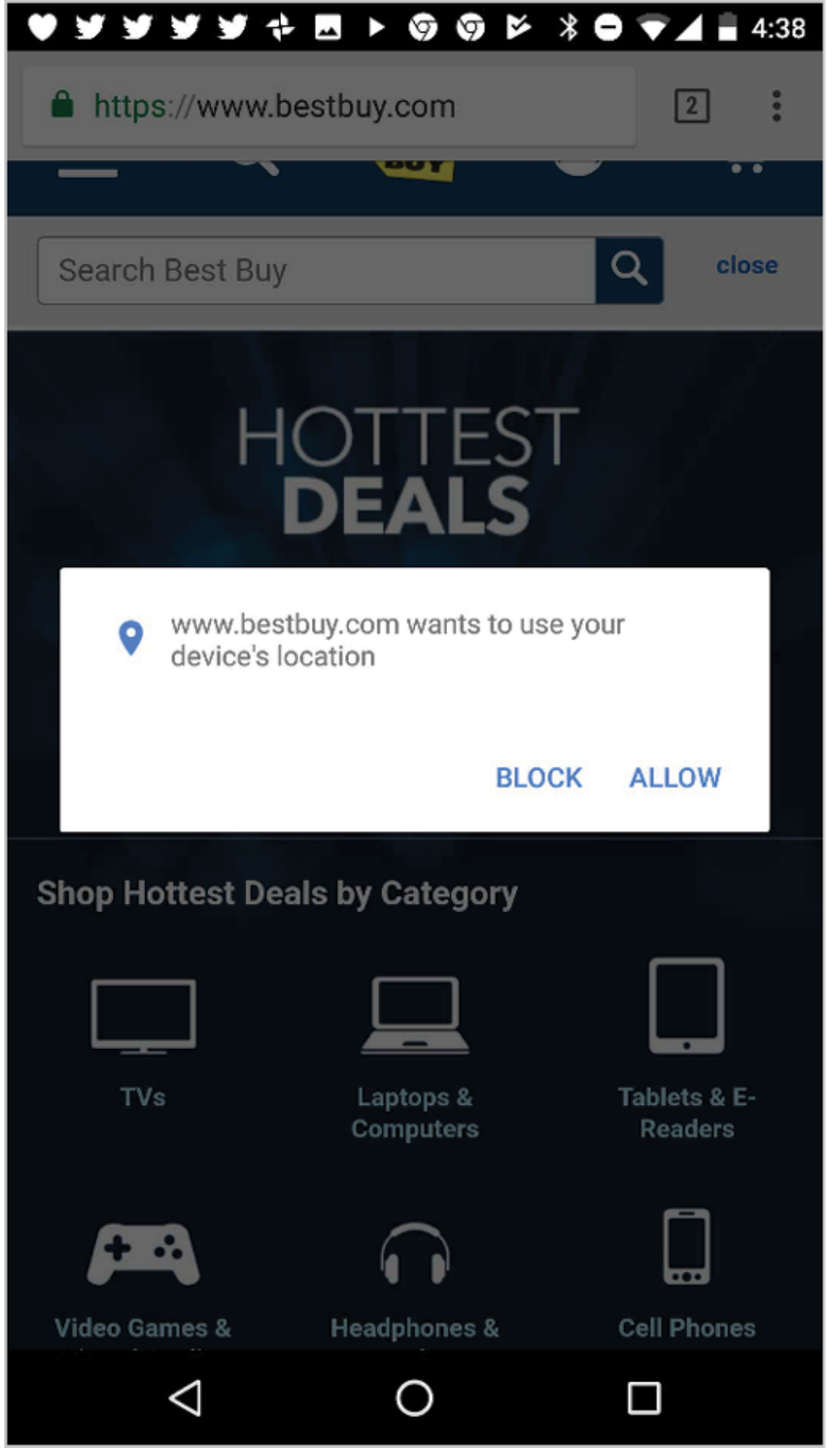
- Supported in Chrome, Firefox, Opera
  - caniuse.com/#feat=push-api
  - No in iOS and everywhere with quirks

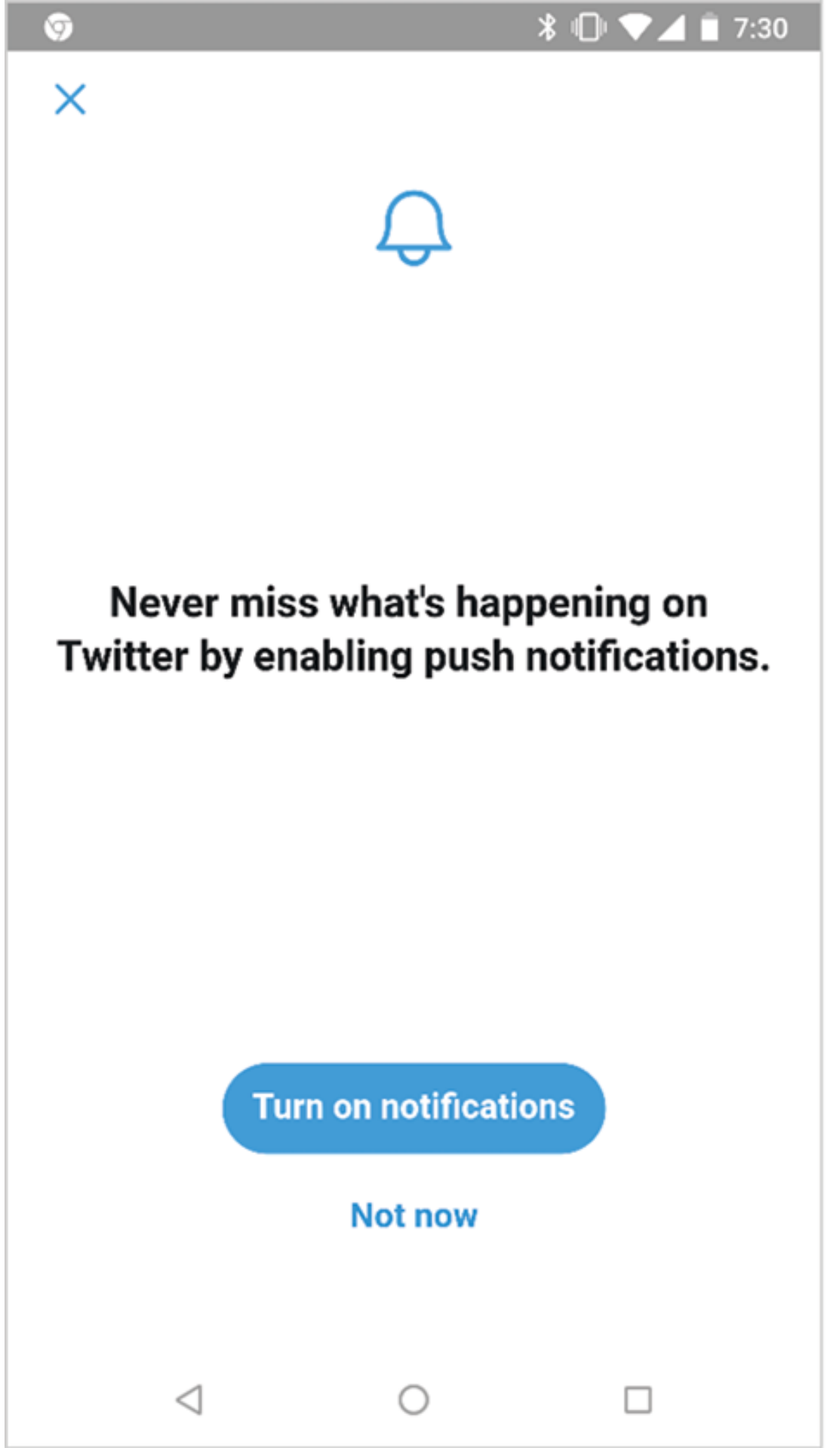
# Best practises

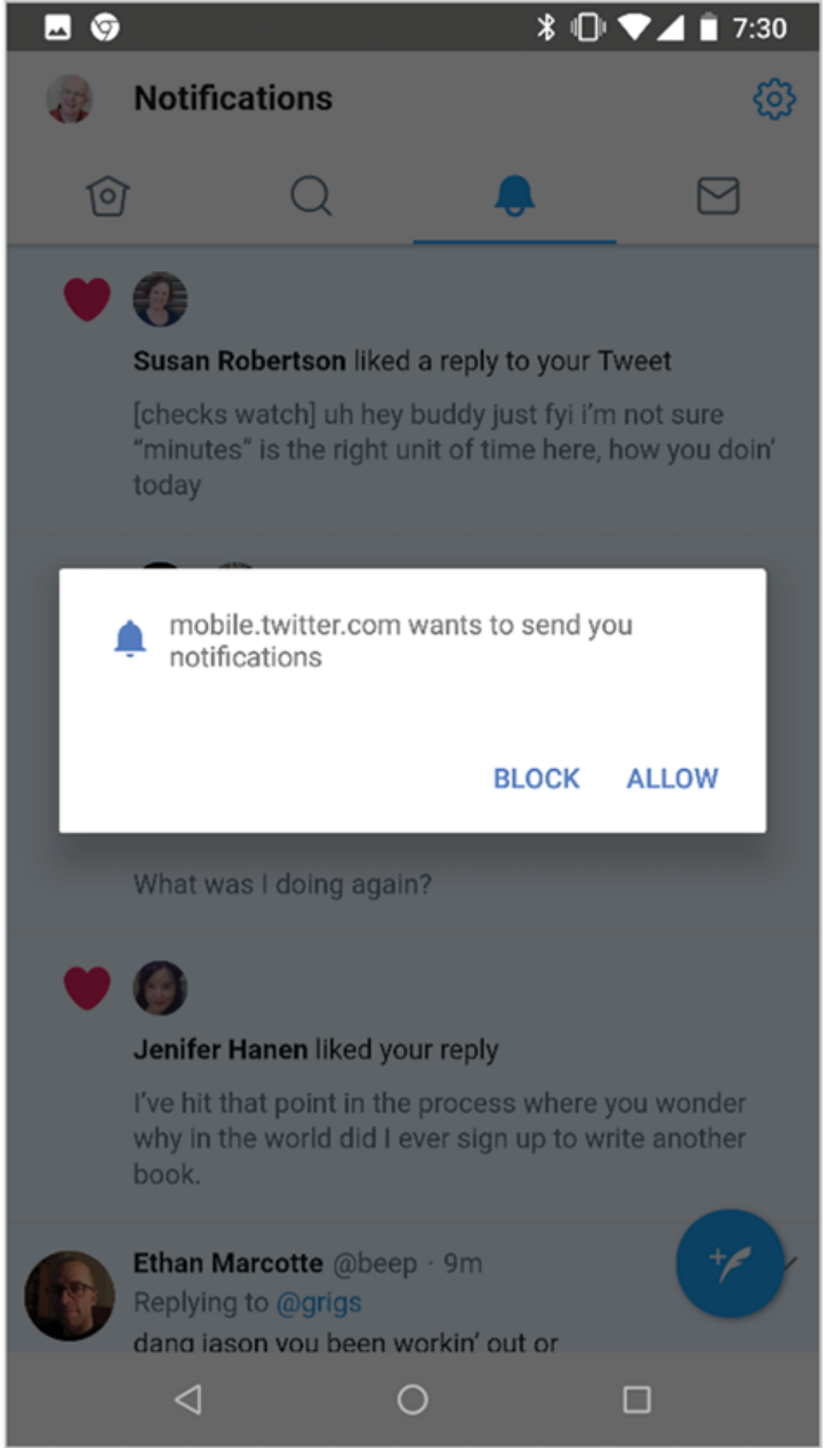
- Personalise by user interest
- Choose the right time to ask permission
  - 90% of users ignore the request
- Design your notifications


Should we send a notification?
Thread message && User subscribed?
Channel muted?
User in DnD?
DnD Override?
@channel/@everyone/@here message?
@channel mentions suppressed?
Channel notification pref is "Nothing"
threads_everything pref on?
Thread message && User subscribed?
What is the user's channel notification pref for this device?
Nothing
Everything
Mentions
Default
What is the user's global notification pref for this device?
DM?
@here
Highlight word?
All
Thread message?
User subscribed?
@mention? Comment on file owned by user? (desktop)
User presence active?
Mentions
DMs (mobile)
Highlight Words (mobile)
Never
Thread message?
User subscribed?
DM?
@mention? Highlight word?
Thread message?
User subscribed?
NO
YES
Mobile?
Past mobile push timing threshold?
Legend
Pref
Pref Value
User State/Action
Parsing
NB: if the user's desktop client Mark as Read pref is set to Mark the channel read, mobile pushes will not be sent for messages marked read on the desktop (ie. when the user leaves their computer open to a channel).


YouTube
Secure
https://www.youtube.com
Apps
www.youtube.com wants to:
Show notifications
Block
Allow


4:38
https://www.bestbuy.com
Search Best Buy
close
HOTTEST
DEALS
www.bestbuy.com wants to use your device's location
BLOCK
ALLOW
Shop Hottest Deals by Category
TVs
Laptops & Computers
Tablets & E-Readers
Video Games &
Headphones &
Cell Phones

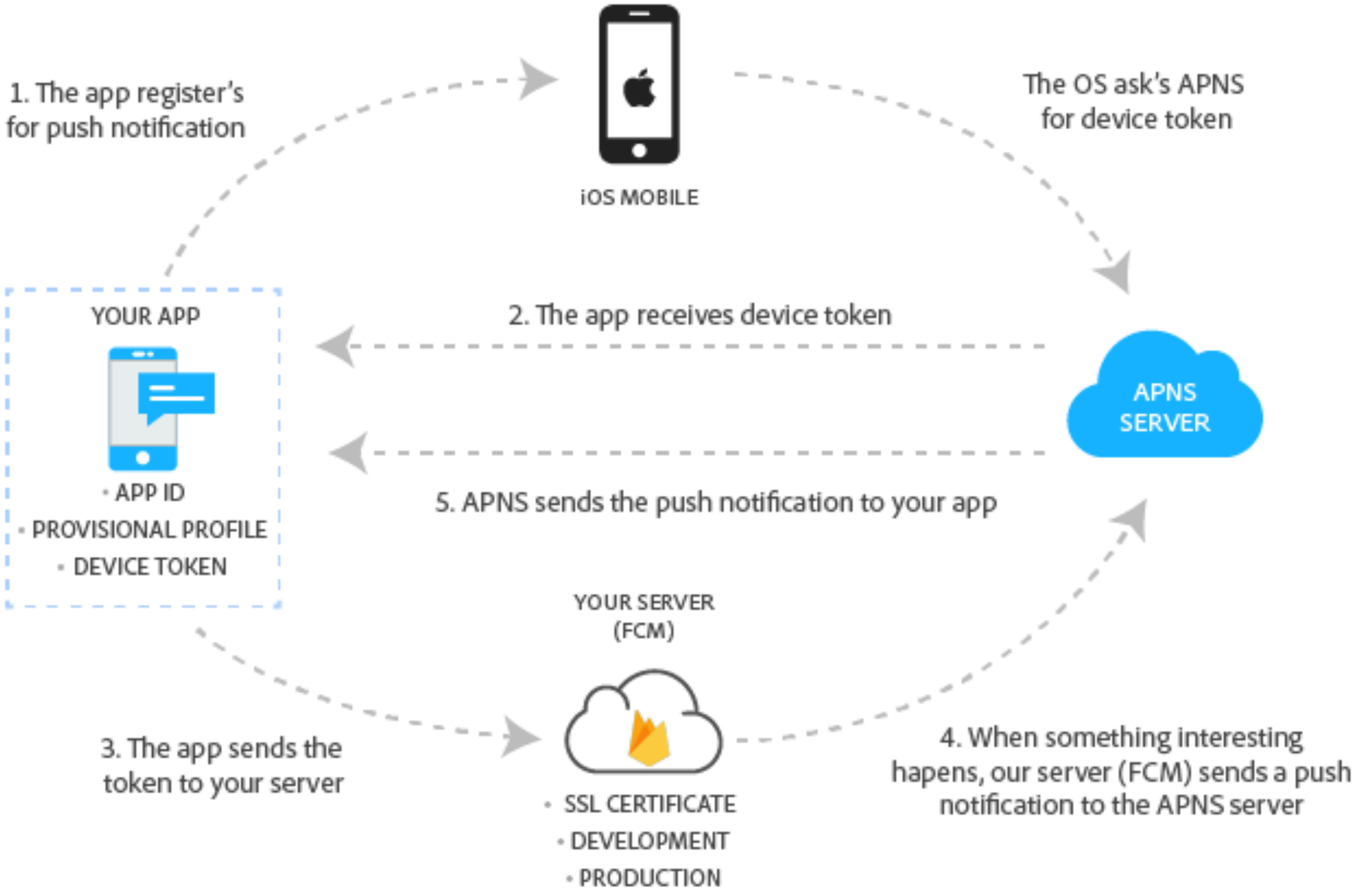

7:30
Never miss what's happening on Twitter by enabling push notifications.
Turn on notifications
Not now


7:30
Notifications
Susan Robertson liked a reply to your Tweet
[checks watch] uh hey buddy just fyi i'm not sure "minutes" is the right unit of time here, how you doin' today
mobile.twitter.com wants to send you notifications
BLOCK
ALLOW
What was I doing again?
Jenifer Hanen liked your reply
I've hit that point in the process where you wonder why in the world did I ever sign up to write another book.
Ethan Marcotte @beep · 9m
Replying to @grigs
dang jason you been workin' out or

## 3 phases

1. Client requests Subscription
2. Server sends Push Message
3. Client shows Notification on device

# Push Notification Flow



# 1. Client requests Subscription

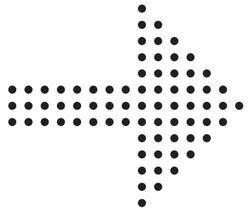
1. Get Permission to Send Push Messages

2. Get PushSubscription

3. Send PushSubscription to Your Server

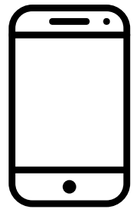
# 2. Server sends Push Message

Your Server



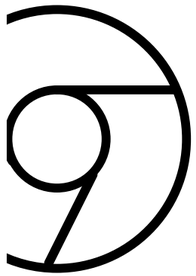
Web Push Protocol Request

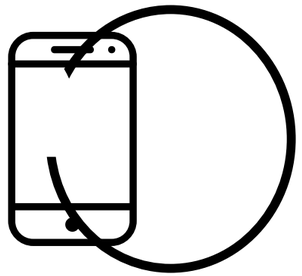
Push Service



Message Arrives on the Device

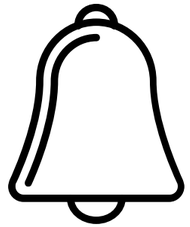
# 3. Client shows Notification on device





1. Message Arrives on Device

2. Browser Wakes Up Service Worker

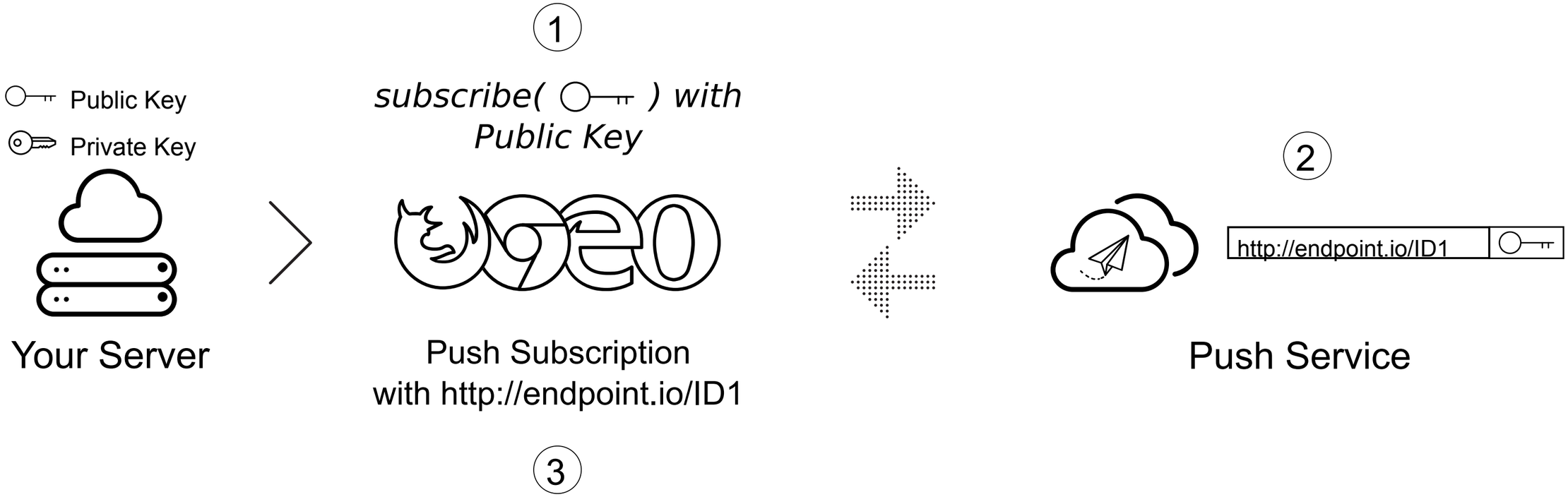


3. Push Event is Dispatched

```
const supportNotifications = 'Notification' in window;

if (supportNotifications || Notification.permission !== 'default') {
  button.style.display = 'none';
}

button.addEventListener('click', () => {
  Notification.requestPermission().then(permission => {
    if (permission !== 'granted') return;

    new Notification('Hei there');
  });
})
```

— Can request only after user event

— User can block requests on mobile


1
Public Key
Private Key
subscribe( ) with
Public Key
2
http://endpoint.io/ID1
Your Server
Push Subscription
with http://endpoint.io/ID1
Push Service
3

— tools.reactpwa.com/vapid

— web-push generate-valid-keys

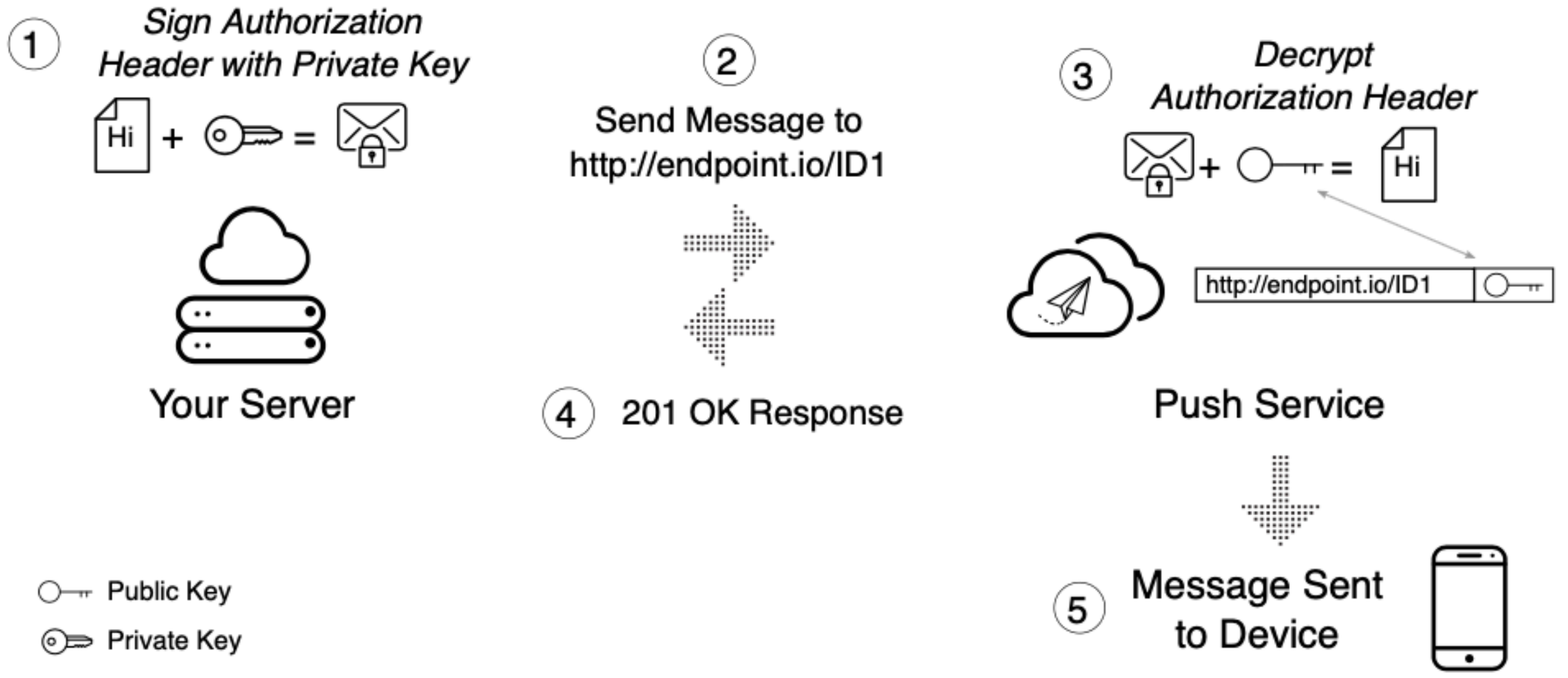
```
function subscribeToPush() {
  return navigator.serviceWorker.ready
    .then(registration => {
      if (!registration.pushManager) return;

      const vapidPublicKey = 'BNNQBC2bIR1I6S0yBoUK6-bgKpa...';

      return registration.pushManager.subscribe({
        userVisibleOnly: true,
        applicationServerKey: urlBase64ToUint8Array(vapidPublicKey),
      });
    })
    .then(pushSubscription => {
      console.log(JSON.parse(JSON.stringify(pushSubscription)));
      return pushSubscription;
    });
}
```

# PushSubscription JSON


```
{
  "endpoint": "https://fcm.googleapis.com/fcm/send/f1f9wU5uWf0:APA91bF4l...",
  "expirationTime": null,
  "keys": {
    "p256dh": "BKsjzTMuVXeuDx7Ac2ccEXuMlAzIP...",
    "auth": "ugNjT6RtpDd..."
  }
}
```

```
function subscribeToPush() {
  return navigator.serviceWorker.ready.then(registration => {
    if (!registration.pushManager) return;

    const vapidPublicKey = 'BNNQBC2bIR1I6S0yBoUK6-bgKpa...';

    return registration.pushManager.subscribe({
      userVisibleOnly: true,
      applicationServerKey: urlBase64ToUint8Array(vapidPublicKey),
    });
  });
}

function requestNotifications() {
  return Notification.requestPermission().then(permission => {
    if (permission !== 'granted') return;

    return permission;
  });
}

requestNotifications()
  .then(() => subscribeToPush())
  .then(pushSubscription => {
    if (pushSubscription) {
      window.fetch(`https://server.com/push-subscriptions/`, {
        method: 'POST',
        body: JSON.stringify(pushSubscription),
      });
    }
  })
```

```
navigator.serviceWorker.ready.then(registration => {
  if (!registration.pushManager) return;

  registration.pushManager.getSubscription().then(subscription => {
    const isSubscribed = subscription !== null
  });
});
```

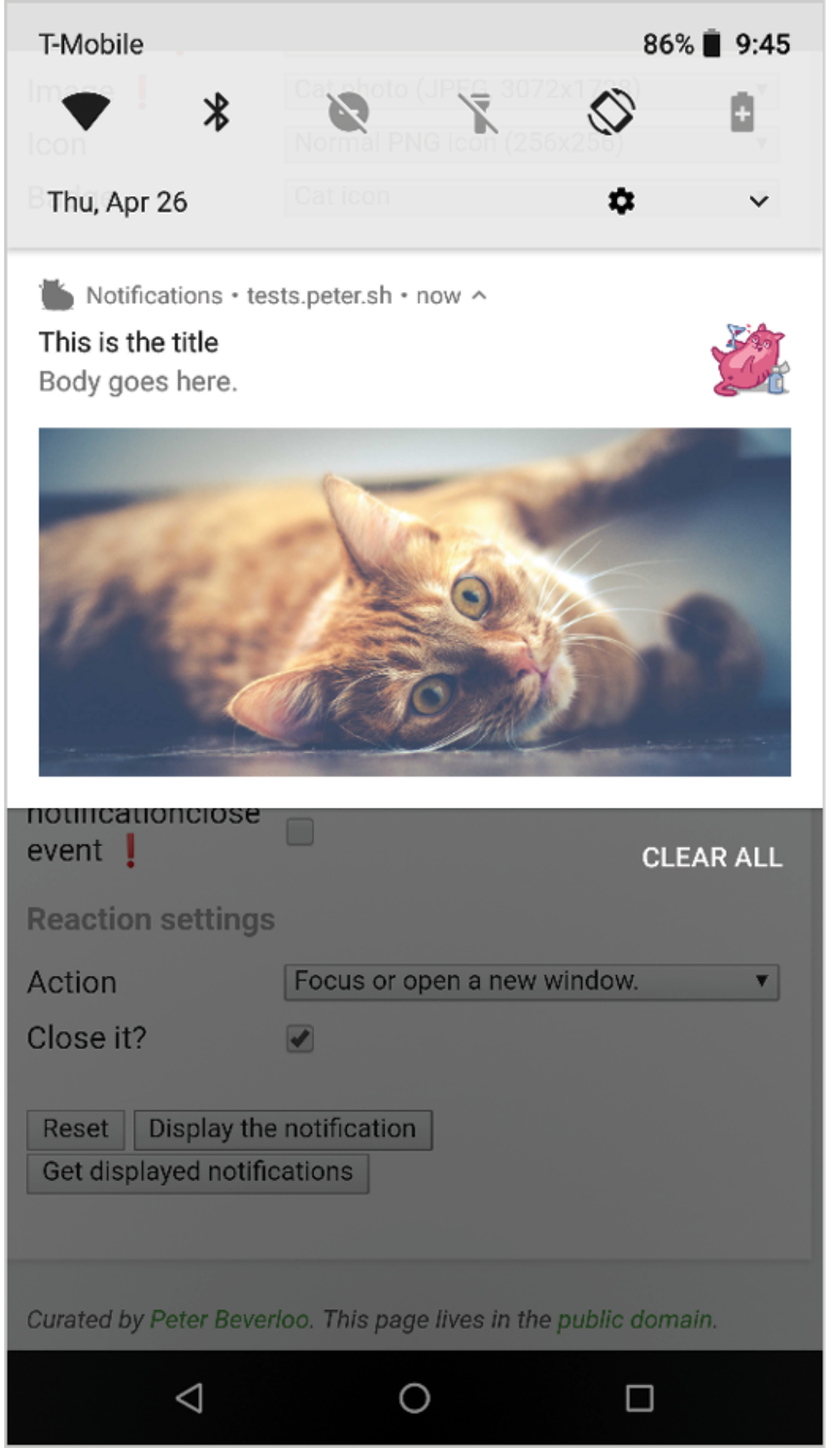

1
Sign Authorization Header with Private Key
Hi
+
=
Your Server
2
Send Message to http://endpoint.io/ID1
4
201 OK Response
3
Decrypt Authorization Header
+
=
Hi
http://endpoint.io/ID1
Push Service
5
Message Sent to Device
Public Key
Private Key

```
self.addEventListener('push', event => {
  if (!event.data) return;

  const message = event.data.text();

  event.waitUntil(self.registration.showNotification(message));
});
```

# Design your notifications

```
{
  body: "<String>",
  icon: "<URL String>", // 192x192
  image: "<URL String>",

  tag: "<String>",

  actions: "<Array of Action>"
}
```

notification-generator

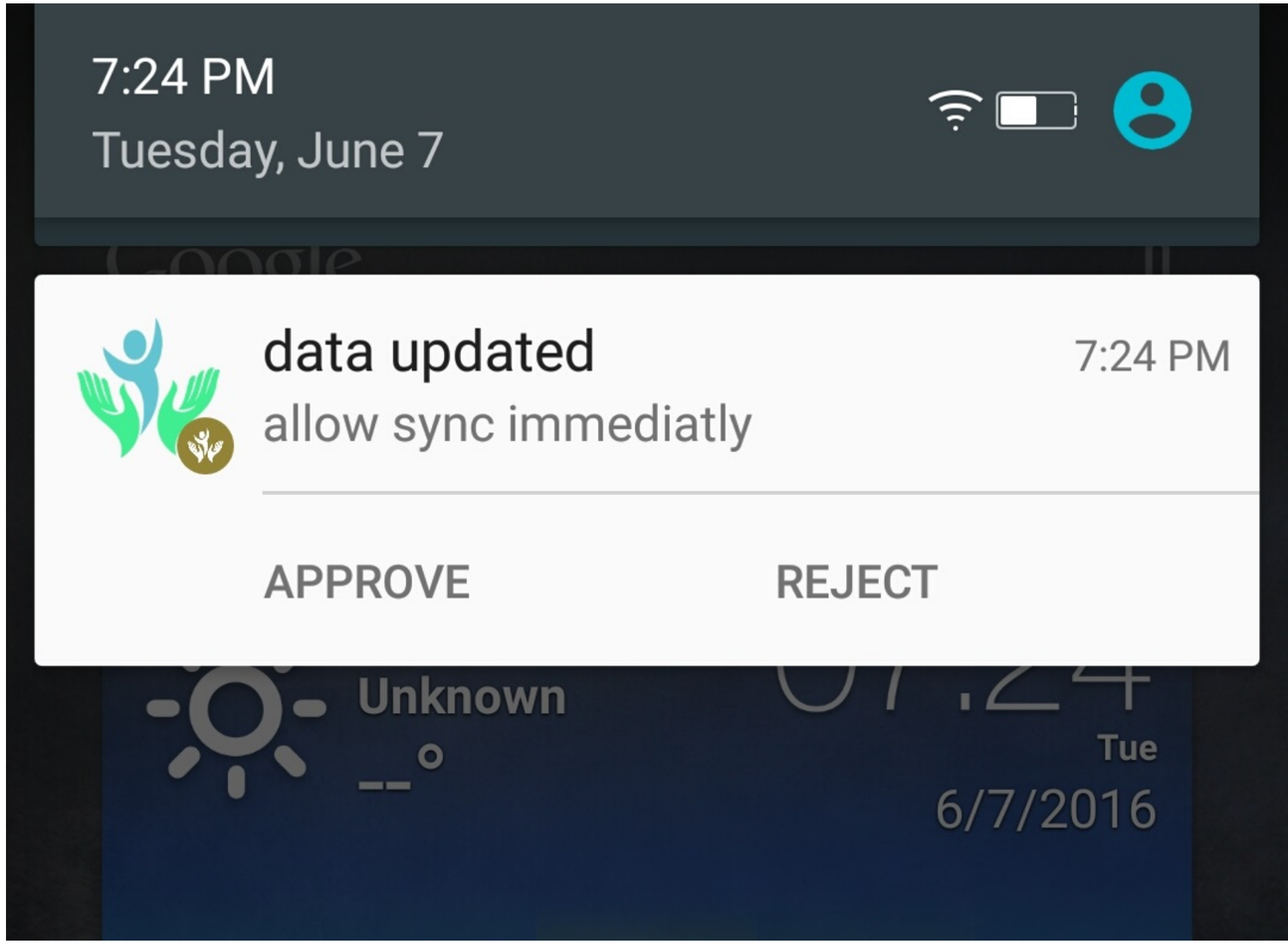



— Visually not consistent in all browsers/OS

# Notification actions

```
const title = 'data updated';
const body = 'allow sync immediately'
const options = {
  actions: [
    {
      action: 'coffee-action',
      title: 'Coffee',
      icon: '/images/demos/action-1-128x128.png'
    },
    {
      action: 'doughnut-action',
      title: 'Doughnut',
      icon: '/images/demos/action-2-128x128.png'
    }
  ]
};
```

— Supported only on Chrome and Opera

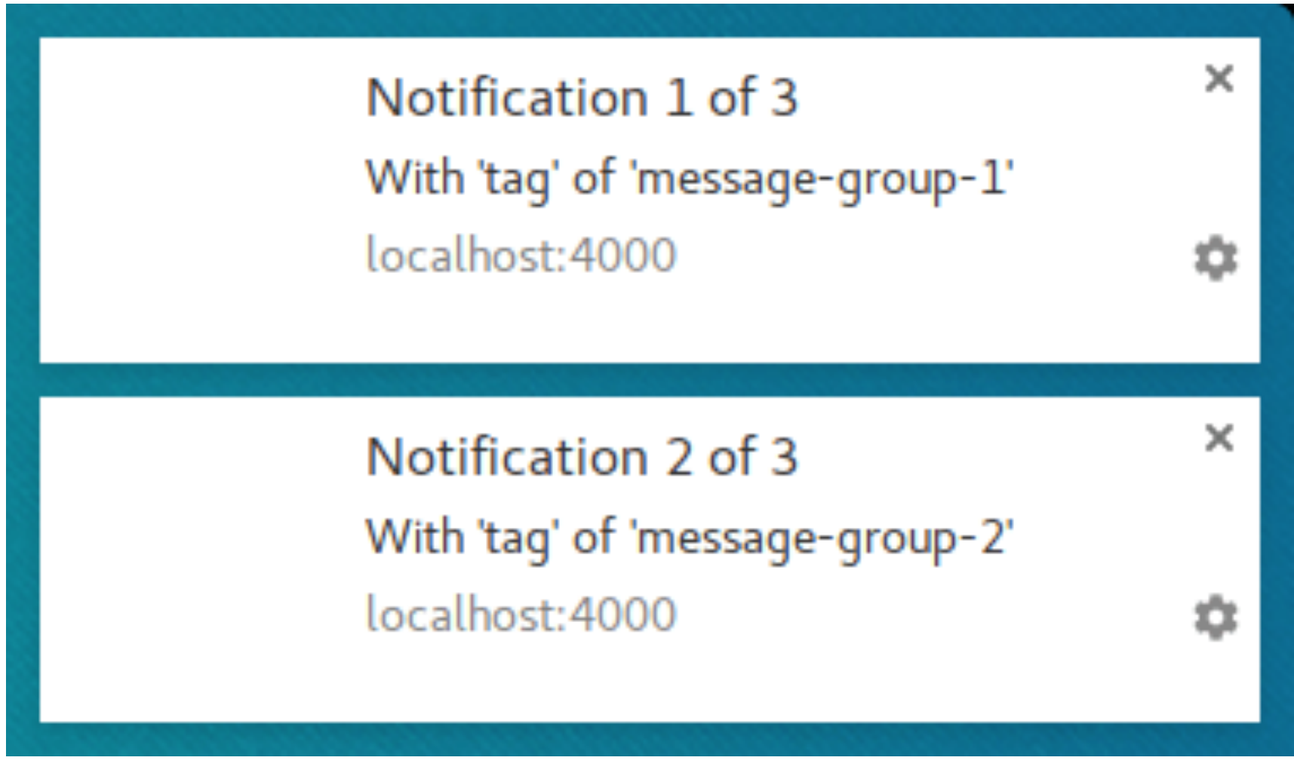


# Open the application on notification click

```
self.addEventListener('push', event => {
  event.waitUntil(
    self.registration.showNotification("It's your turn in the queue", {
      body: 'Move up',
      icon: '/images/icons/icon-192x192.png',
      data: { url: '/' },
    }),
  );
});

self.addEventListener('notificationclick', event => {
  const notification = event.notification;
  notification.close();

  event.waitUntil(
    self.clients.matchAll({ type: 'window' }).then(clientList => {
      const thereIsFocused = clientList.find(client => client.focused);
      if (thereIsFocused) return;

      const hasWindowToFocus = clientList.length > 0;
      if (hasWindowToFocus) clientList[0].focus();

      if (!hasWindowToFocus) {
        clients
          .openWindow(notification.data.url)
          .then(windowClient => (windowClient ? windowClient.focus() : null));
      }
    }),
  );
});
```

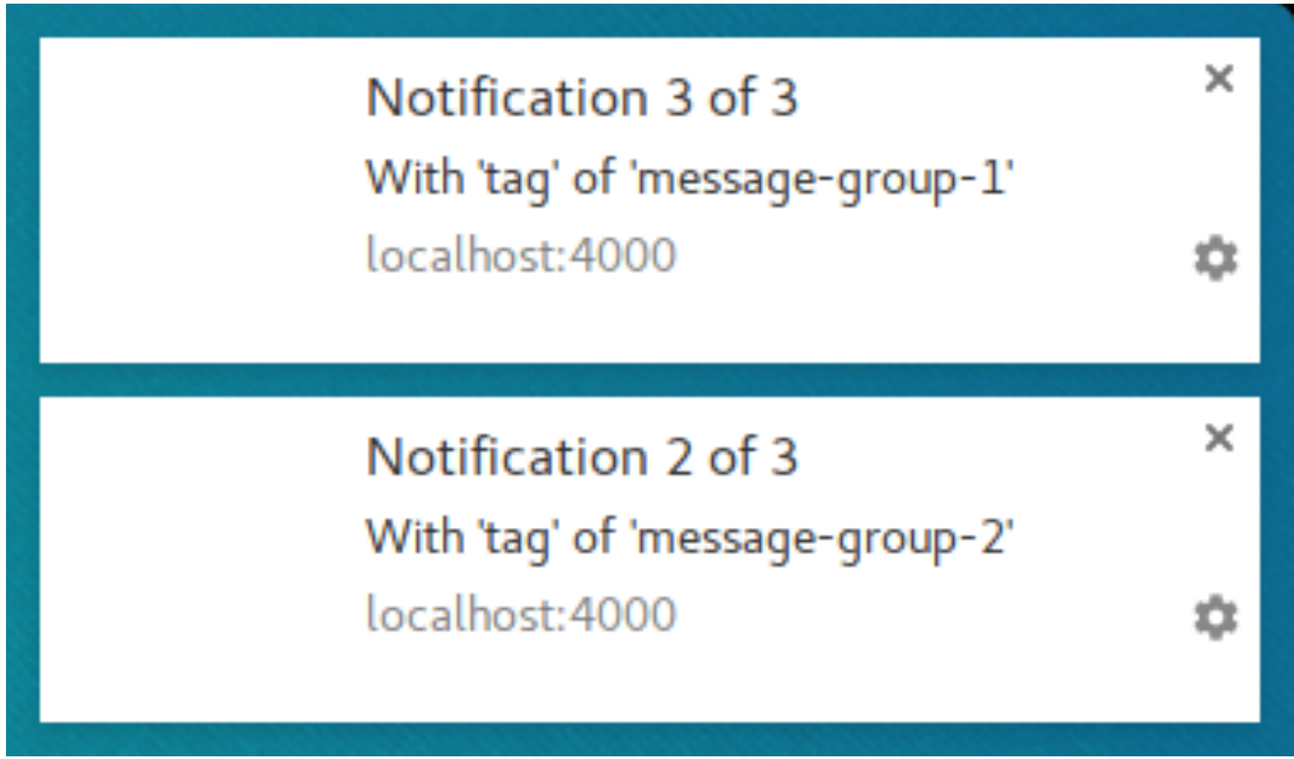
# Notification tag

```
registration.showNotification('Notification 1 of 3', {
  body: "With tag of message-group-1',
  tag: 'message-group-1'
});
registration.showNotification('Notification 2 of 3', {
  body: "With tag of message-group-2',
  tag: 'message-group-2'
});
```



# Notification tag

```
const title = 'Notification 3 of 3';
const options = {
  body: "With tag of message-group-1',
  tag: 'message-group-1'
};
registration.showNotification(title, options);
```

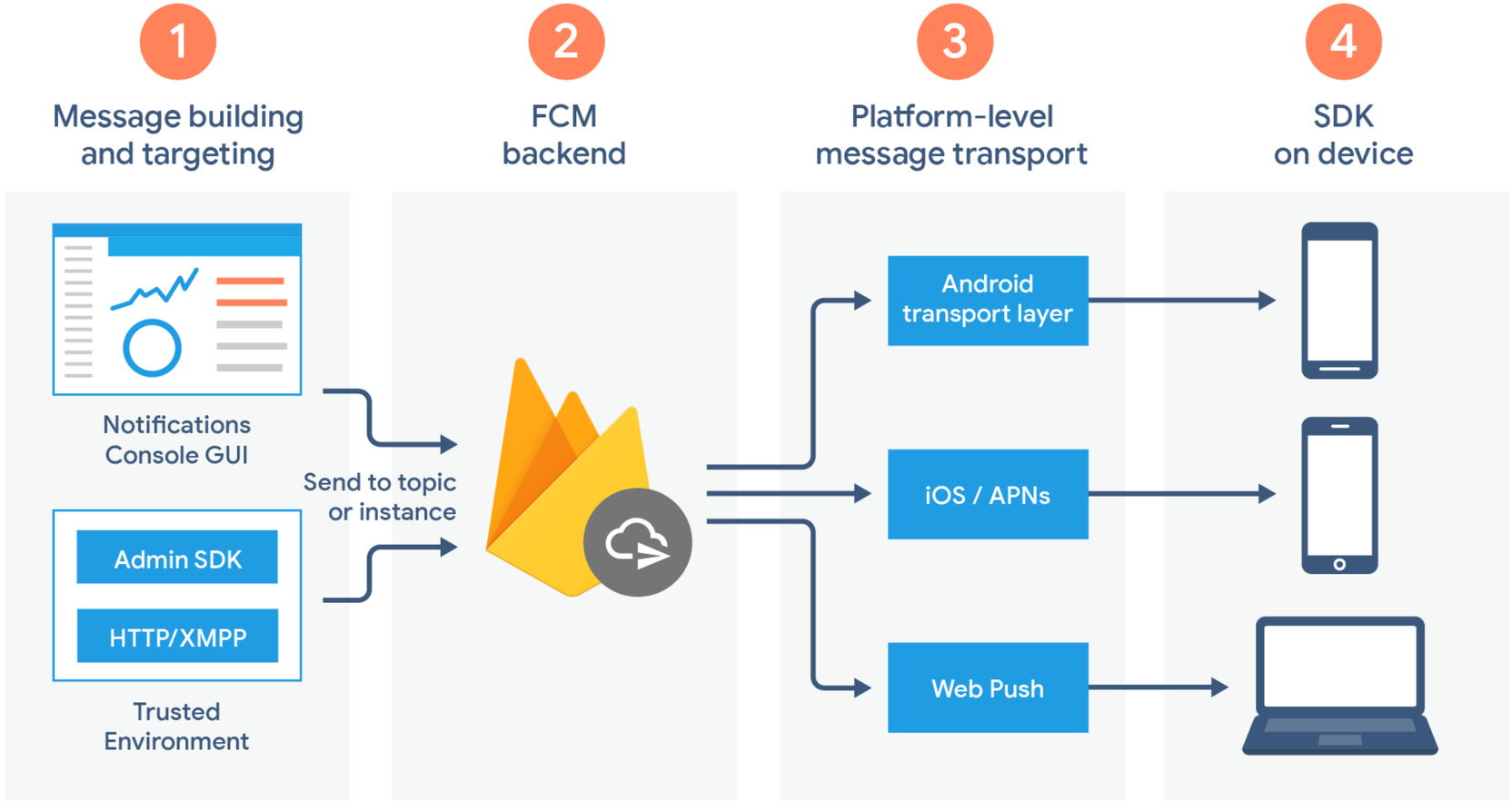


# Notification gotchas

— On desktop, are shown only if the browser is open

— On mobile Chrome, the PWA is opened as standalone only if used recently

# Firebase Cloud Messaging



# BackgroundSync

— Default backoff strategy
— On Chrome it makes three attempts
— Advanced: implement a Queue of failed requests

# BackgroundSync constraints

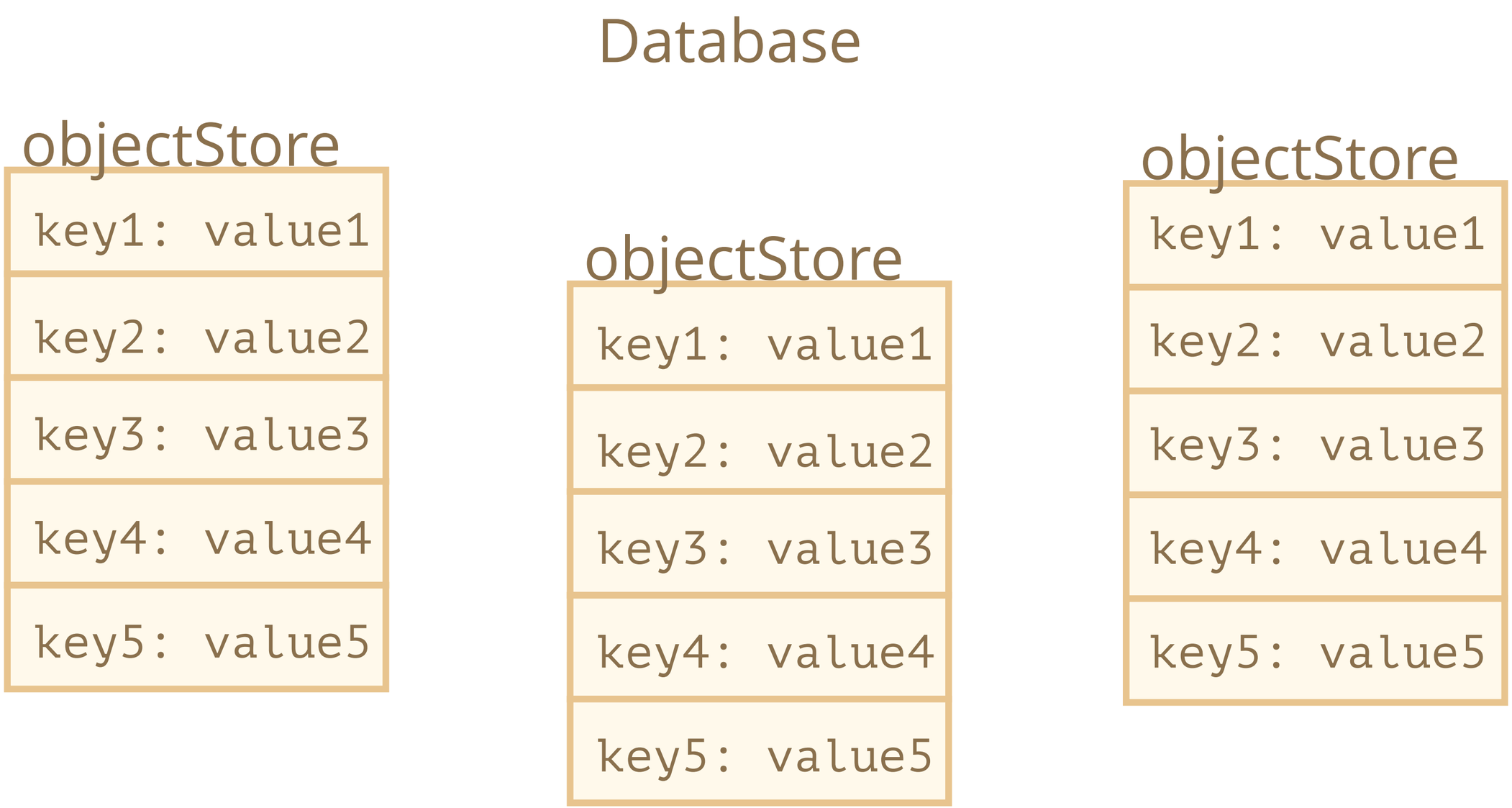
— A sync event can be registered only if the page is active

— The execution time is limited

```
return request(`bulletins/${bulletinId}/messages`, {
  method: 'POST',
  body: JSON.stringify(body),
}).catch(() => {
  if (window.indexedDB && window.SyncManager) {
    getIDB()
      .then(db => {
        const request = db
          .transaction('new-messages', 'readwrite')
          .objectStore('new-messages')
          .put({ body, bulletinId });

        return promisifyRequest(request);
      })
      .then(() => {
        return navigator.serviceWorker.ready.then(registration => {
          return registration.sync.register('new-message');
        });
      });
  }
});
```

# IndexedDB

- In-browser NoSQL database
- Bulti-in async API
  - Results are dispatched as events
- A *key-value* object-oriented database
- Transactional (ACI~~D~~)
- Supports concurrency


Database
objectStore
key1: value1
key2: value2
key3: value3
key4: value4
key5: value5
objectStore
key1: value1
key2: value2
key3: value3
key4: value4
key5: value5
objectStore
key1: value1
key2: value2
key3: value3
key4: value4
key5: value5

```
export const getIDB = () => {
  return new Promise<IDBDatabase>((resolve, reject) => {
    const request = indexedDB.open('daisyhub', 1);

    request.addEventListener('error', event => {
      const error = (event.target as IDBRequest).error;
      reject(error);
    });

    request.addEventListener('success', () => {
      const db = request.result;

      db.addEventListener('versionchange', () => {
        db.close();
        window.location.reload();
      });

      resolve(db);
    });

    request.addEventListener('upgradeneeded', () => {
      const db = request.result;
      if (!db.objectStoreNames.contains('new-messages')) {
        db.createObjectStore('new-messages', { keyPath: 'body.authorId' });
      }
    });

    request.addEventListener('blocked', () => {
      reject('Request to open IDB was blocked');
    });
  });
};
```

```
const request = indexedDB.open("store", 2);

request.onupgradeneeded = function() {
  // Existing db version
  const db = request.result;

  switch(db.version) {
    case 0:
      ...
    case 1:
      ...
  }
};
```

```
self.addEventListener('sync', event => {
  switch (event.tag) {
    case 'new-message': {
      console.log('New message to send')
      break;
    }
    default:
      break;
  }
});
```

```
const operation = getIDB('daisyhub', 1)
  .then(db => {
    const request = db.transaction('new-messages').objectStore('new-messages').getAll();

    request.addEventListener('success', () => {
      console.log(request.result);
    })
  })
event.waitUntil(operation);
```

```
function promisifyRequest(request) {
  return new Promise((resolve, reject) => {
    request.onsuccess = function() {
      resolve(request.result);
    };

    request.onerror = function() {
      reject(request.error);
    };
  });
}
```

```
const operation = getIDB('daisyhub', 1)
  .then(db => {
    const request = db.transaction('new-messages').objectStore('new-messages').getAll();

    return promisifyRequest(request).then(messages => {
      console.log(messages)
    });
  })
event.waitUntil(operation);
```

```
const operation = getIDB('daisyhub', 1)
  .then(db => {
    const request = db.transaction('new-messages').objectStore('new-messages').getAll();

    return promisifyRequest(request).then(messages => {
      const fetches = messages.map(message => {
        const { bulletinId, body } = message;

        return fetch(
          `https://europe-west2-turnips-274820.cloudfunctions.net/app/bulletins/${bulletinId}/messages`,
          {
            headers: {
              'Content-Type': 'application/json',
            },
            method: 'POST',
            body: JSON.stringify(body),
          },
        )
      });

      return Promise.all(fetches);
    });
  })
event.waitUntil(operation);
```

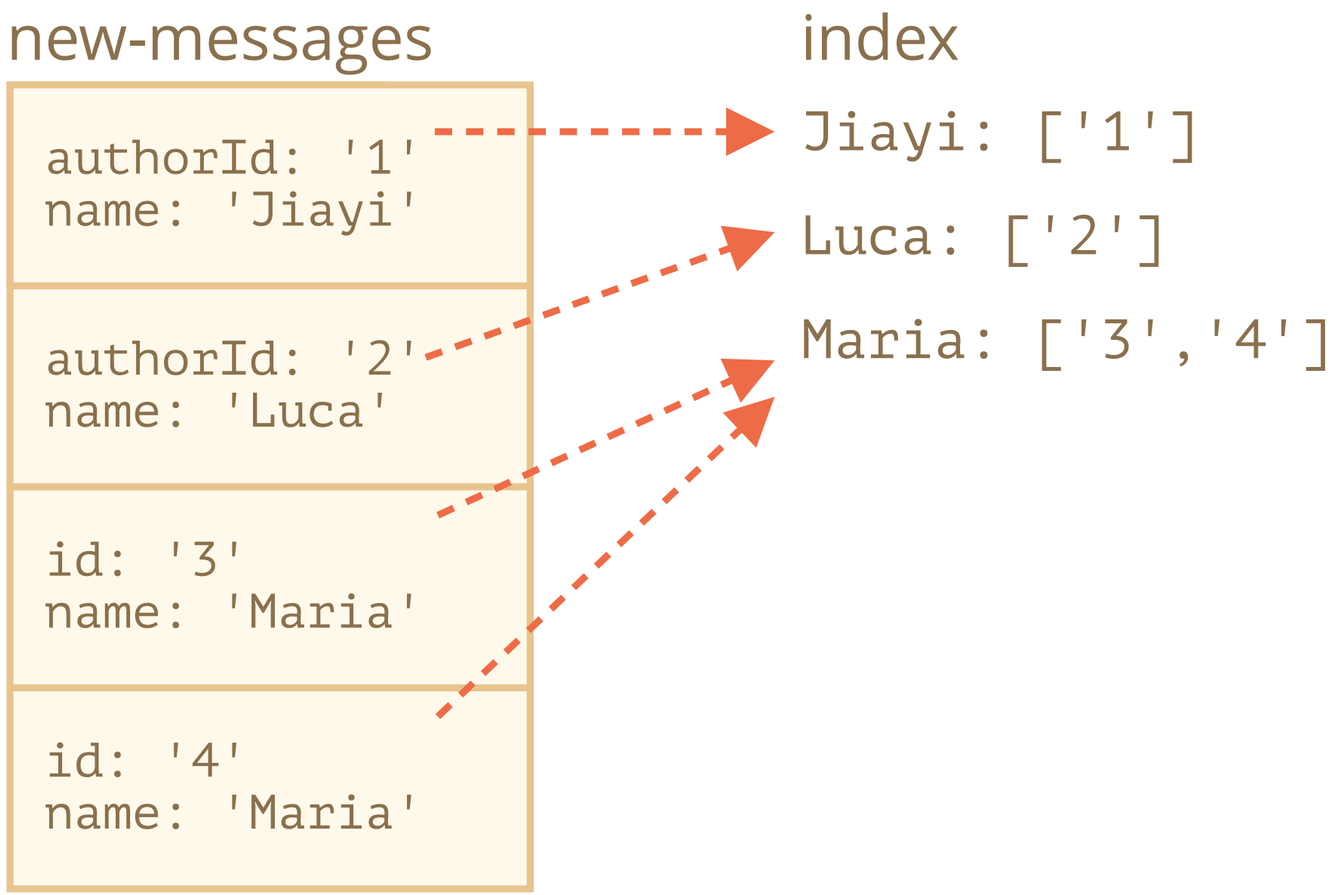
```
.then(() => {
  const request = db
    .transaction('new-messages', 'readwrite')
    .objectStore('new-messages')
    .delete(body.authorId);

  return promisifyRequest(request);
});
```

```
.catch(error => {
  if (event.lastChance) {
    self.registration.showNotification('Message error', {
      body: 'A message could not be sent due to connection issues',
    });
  }

  throw error;
});
```

# Transactions should be short-lived, for performance reasons.

```
const newMessages = db
  .transaction('new-messages', 'readwrite')
  .objectStore('new-messages')
const request = newMessages
  .add({ body, bulletinId });

request.onsuccess = function() {
  fetch('/messages/', { method: 'POST' }).then(response => {
    const request2 = newMessages.add({ ... }); // <=

    request2.onerror = function() {
      console.log(request2.error.name); // TransactionInactiveError
    };
  });
};
```

# Indexes


new-messages
authorId: '1'
name: 'Jiayi'
authorId: '2'
name: 'Luca'
id: '3'
name: 'Maria'
id: '4'
name: 'Maria'
index
Jiayi: ['1']
Luca: ['2']
Maria: ['3','4']

# Simpler abstractions

— jakearchibald/idb-keyval

— jakearchibald/idb

```
if (!idb)
  self.importScripts('https://unpkg.com/idb@5.0.2/build/iife/index-min.js');
```

# Planning your PWA

1. Plan features
    1. Offline support
    2. Push Notifications
    3. Payment Request API
2. Start from a good baseline
3. Measure

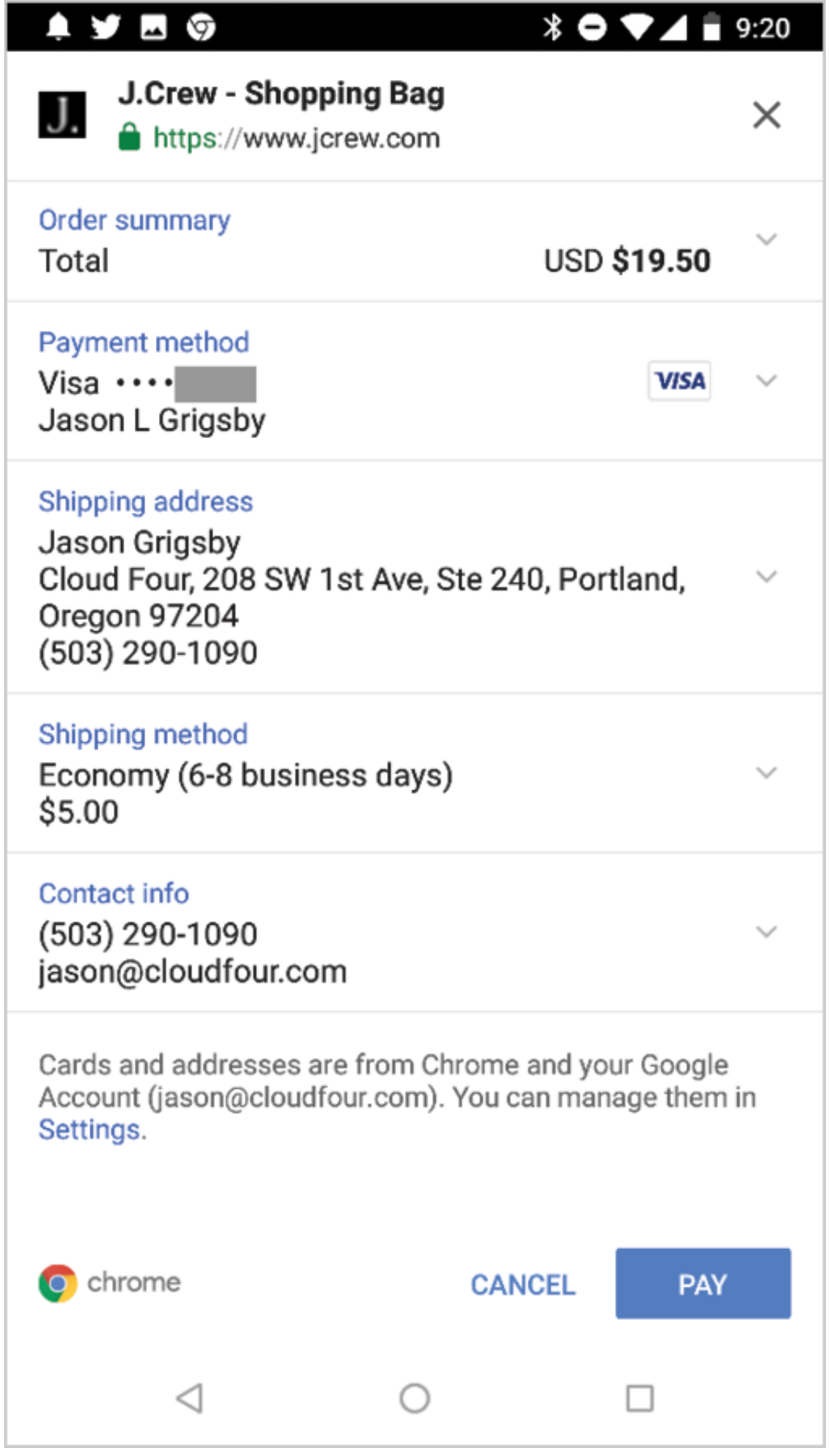
## Step by step development

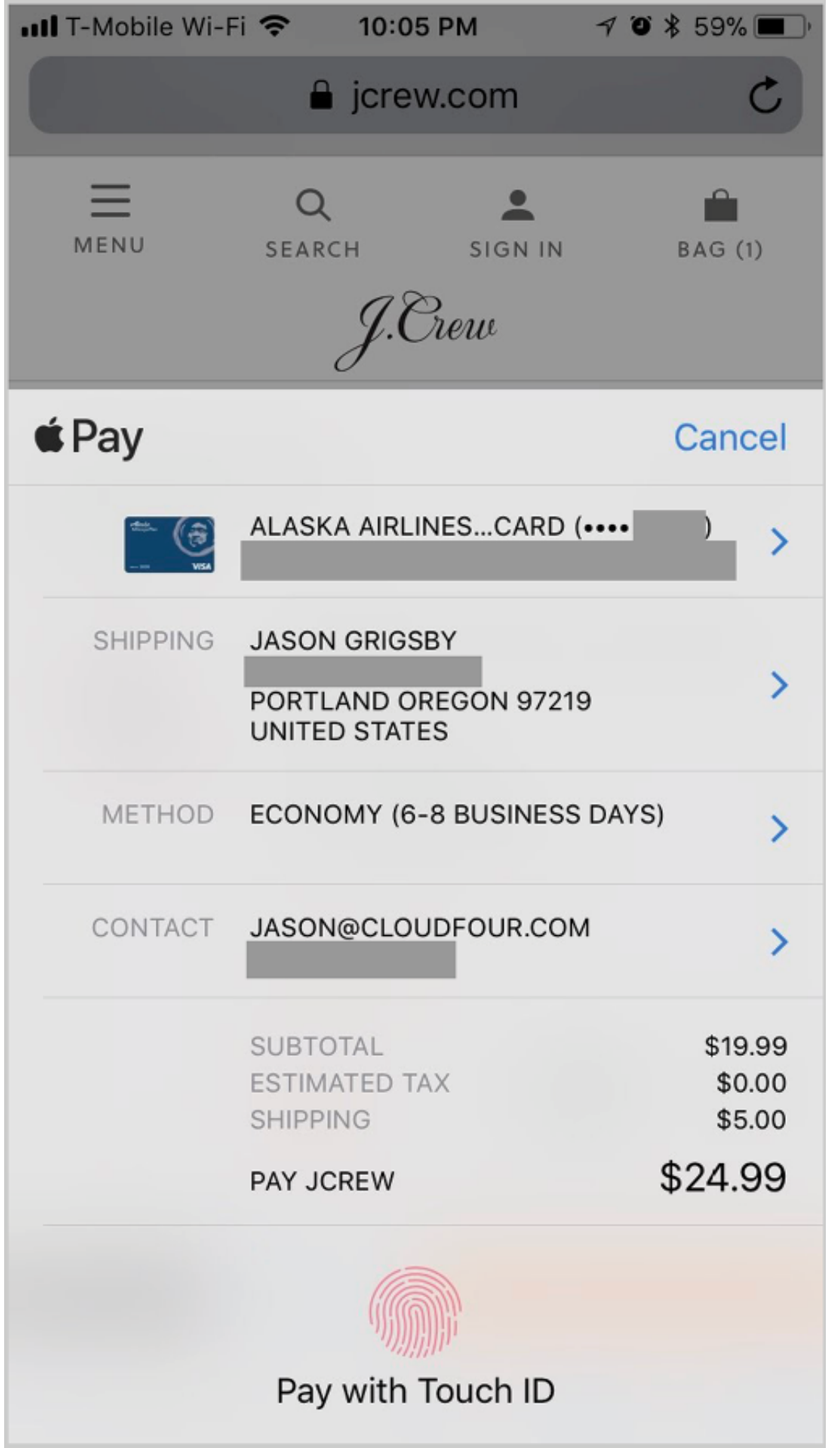
1. Add a manifest and a cache-first Service Worker
2. Offline support with notification
3. Content loader, page transitions, animations etc.
4. Push notifications

# Beyond PWAs

— IndexedDB

— Payment Request API


9:20
J.Crew - Shopping Bag
https://www.jcrew.com
Order summary
Total
USD $19.50
Payment method
Visa ••••
Jason L Grigsby
Shipping address
Jason Grigsby
Cloud Four, 208 SW 1st Ave, Ste 240, Portland, Oregon 97204
(503) 290-1090
Shipping method
Economy (6-8 business days)
$5.00
Contact info
(503) 290-1090
jason@cloudfour.com
Cards and addresses are from Chrome and your Google Account (jason@cloudfour.com). You can manage them in Settings.
chrome
CANCEL
PAY


T-Mobile Wi-Fi
10:05 PM
59%
jcrew.com
MENU
SEARCH
SIGN IN
BAG (1)
J.Crew
Pay
Cancel
ALASKA AIRLINES...CARD (••••
SHIPPING
JASON GRIGSBY
PORTLAND OREGON 97219
UNITED STATES
METHOD
ECONOMY (6-8 BUSINESS DAYS)
CONTACT
JASON@CLOUDFOUR.COM
SUBTOTAL $19.99
ESTIMATED TAX $0.00
SHIPPING $5.00
PAY JCREW $24.99
Pay with Touch ID

# Thanks!