Get Progressive with it

About me

Matt LaForest

Brighton, MI (about 20 miles north of Ann Arbor)

TED Conferences - Senior Software Engineer

Lots of languages, mostly JS now

Pretty much no free time (wife, 3 kids 6 and under, dog)

Web “Apps” a brief history (inspired by actual events)

- The World Wide Web “created” by Tim Berners-Lee and first launched around Christmas 1990

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval i

Everything there is online about W3 is linked directly or indirectly to this do
Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mo
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Web “Apps” a brief history (inspired by actual events)

- The World Wide Web
  “created” by Tim Berners-Lee
  and first launched around
  Christmas 1990

- Cell phones at the time not
  really ready for the web

Web “Apps” a brief history (inspired by actual events)

- Netscape and Sun form an alliance in 1995 to counter Microsoft

- Brandon Eich creates Javascript (called Mocha at the time) in 10 days

Web “Apps” a brief history (inspired by actual events)

- Netscape and Sun form an alliance in 1995 to counter Microsoft

- Brandon Eich creates Javascript (called Mocha at the time) in 10 days

- Cell phones… well they still aren’t quite ready for the web

Web “Apps” a brief history (inspired by actual events)

- GMail - 2004
- Google Maps - 2005

Web “Apps” a brief history (inspired by actual events)

- GMail - 2004
- Google Maps - 2005
- Cell phones are starting to make some progress

Web “Apps” a brief history (inspired by actual events)

- GMail - 2004
- Google Maps - 2005
- Cell phones are starting to make some progress
- Early smartphones start to emerge

Web “Apps” a brief history (inspired by actual events)

Web “Apps” a brief history (inspired by actual events)

The full Safari engine is inside of iPhone. And so, you can write amazing Web 2.0 and Ajax apps that look exactly and behave exactly like apps on the iPhone. And these apps can integrate perfectly with iPhone services. They can make a call, they can send an email, they can look up a location on Google Maps.

And guess what? There’s no SDK that you need! You’ve got everything you need if you know how to write apps using the most modern web standards to write amazing apps for the iPhone today. So developers, we think we’ve got a very sweet story for you. You can begin building your iPhone apps today.

- Steve Jobs (2007) iPhone release

Web “Apps” a brief history (inspired by actual events)

- HTML 4.01 was released in 1999

- HTML 5 not formally released as a recommendation until 2014

- Javascript 3 had been released in 1999

- Javascript 5 (4 shan’t be spoken of) wouldn’t be released until 2009

Enter Native Apps

- Apple announces development of Native SDK in October 2007

- No more web standards used Objective C (later Swift)

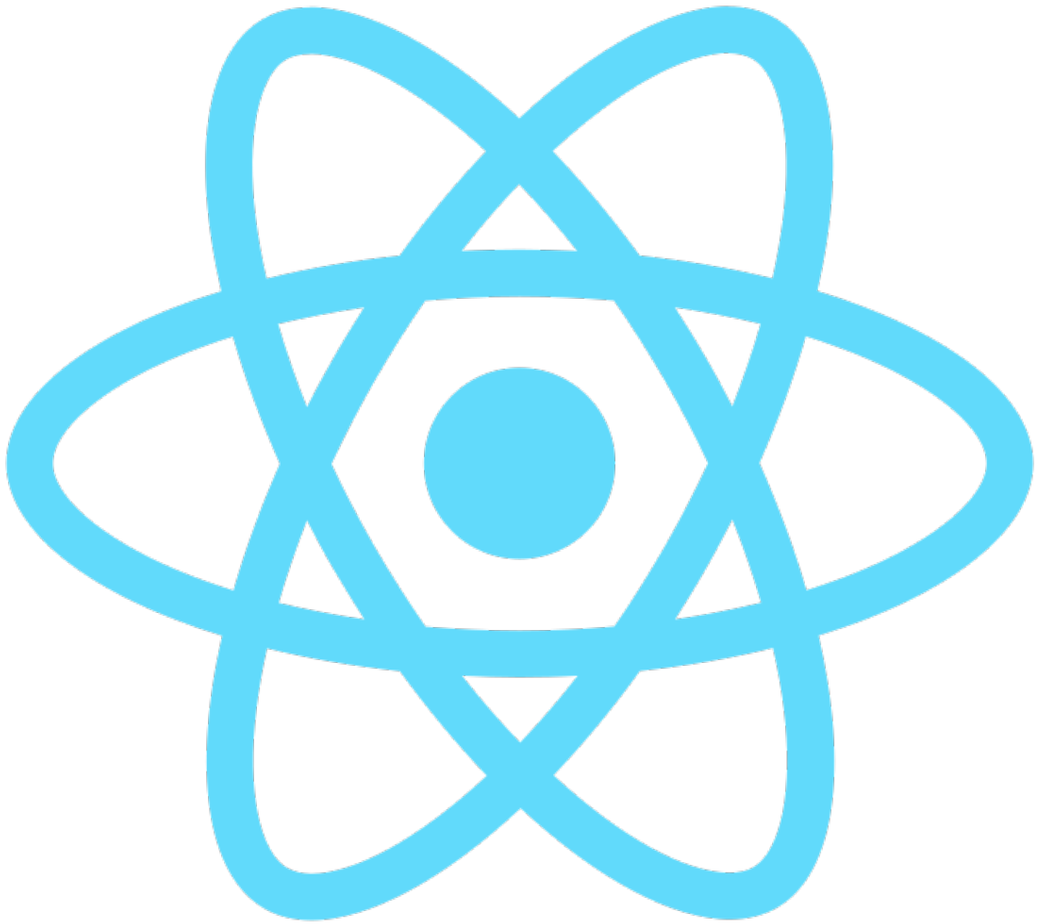
- App Store opens July 2008

Hybrid apps

- Native SDK includes WebView
- JS wrappers provide access to native interfaces (Cordova)
- Uncanny valley
- Offline support poor

React Native (et al)

- JS application that renders native UI components
- Alleviates the uncanny valley (somewhat)
- Creates a new set of challenges
- Tacit approval from Apple, but questions about whether that will continue



Does it have to be native?

- Frances Berman and Alex Russell coin the term Progressive Web App - 2015

- Web apps leveraging new powers provided by Service Workers, App Manifests, and other modern APIs

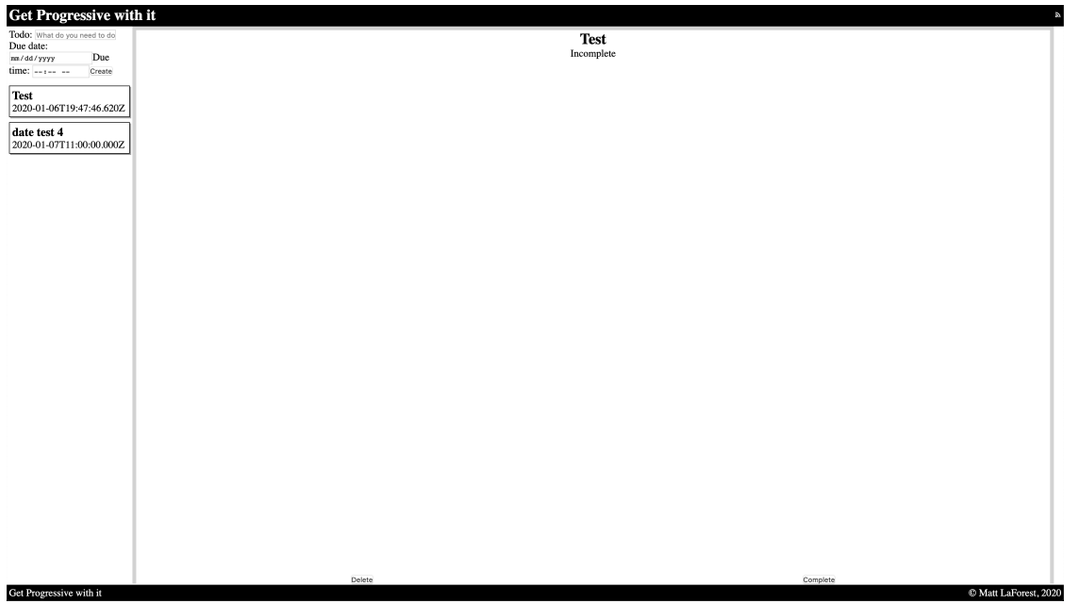
- Installable as “Apps” to your mobile home screens

Progressive Web Apps

- Fast even on slow devices and networks
- Can work offline
- Support push notifications
- Can leverage other new browser APIs as the become available (like camera access)

What are we building?

- Lightweight Todo app
- A few caveats
  - App “shell” is mostly written, not what we are focusing on today
  - App itself written in React, but you do NOT need to be familiar with React to do this
  - I am not a designer don’t @me
  - Please, I beg of you don’t deploy this anywhere there are so many security holes and design flaws I didn’t even try to count them



Demo

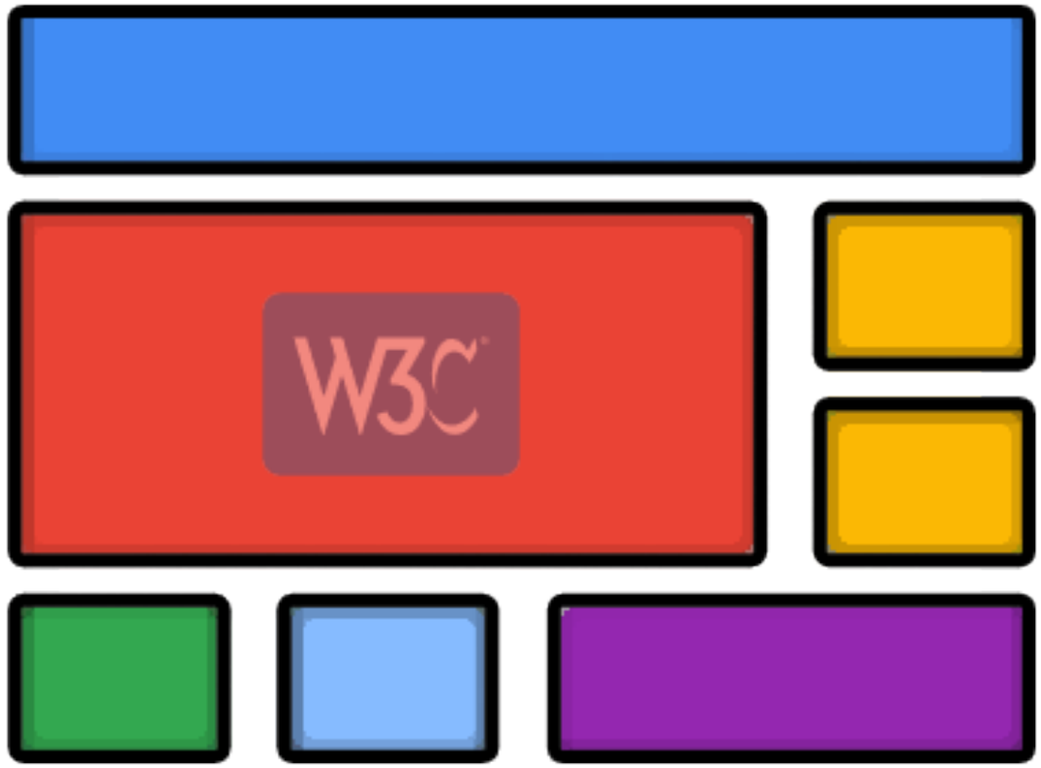
Semantic CSS Grid Responsive Design

Responsive web design

- Fluid grid with break points to support different size screens
- Mobile first or desktop first approach
- Traditionally done by manipulating widths and floats of elements

CSS Grid

- Second new modern layout system for CSS, after Flexbox
- Pretty widely supported, particularly within the mobile devices
- Allows for defining the grid semantically



CSS Grid

```
<div class="site">
    <header class="site-header"></header>
    <main class="site-main"></main>
    <aside class=“site-sidebar"></aside>
    <footer class="site-footer"></footer>
</div>
```

```
.site {
  display: grid;
  grid-template-columns: 15em 1fr;
  grid-template-rows: 1fr minmax(1em, 3em);
  grid-template-areas: "header header"
                       "main sidebar"
                       “footer footer";
}
```

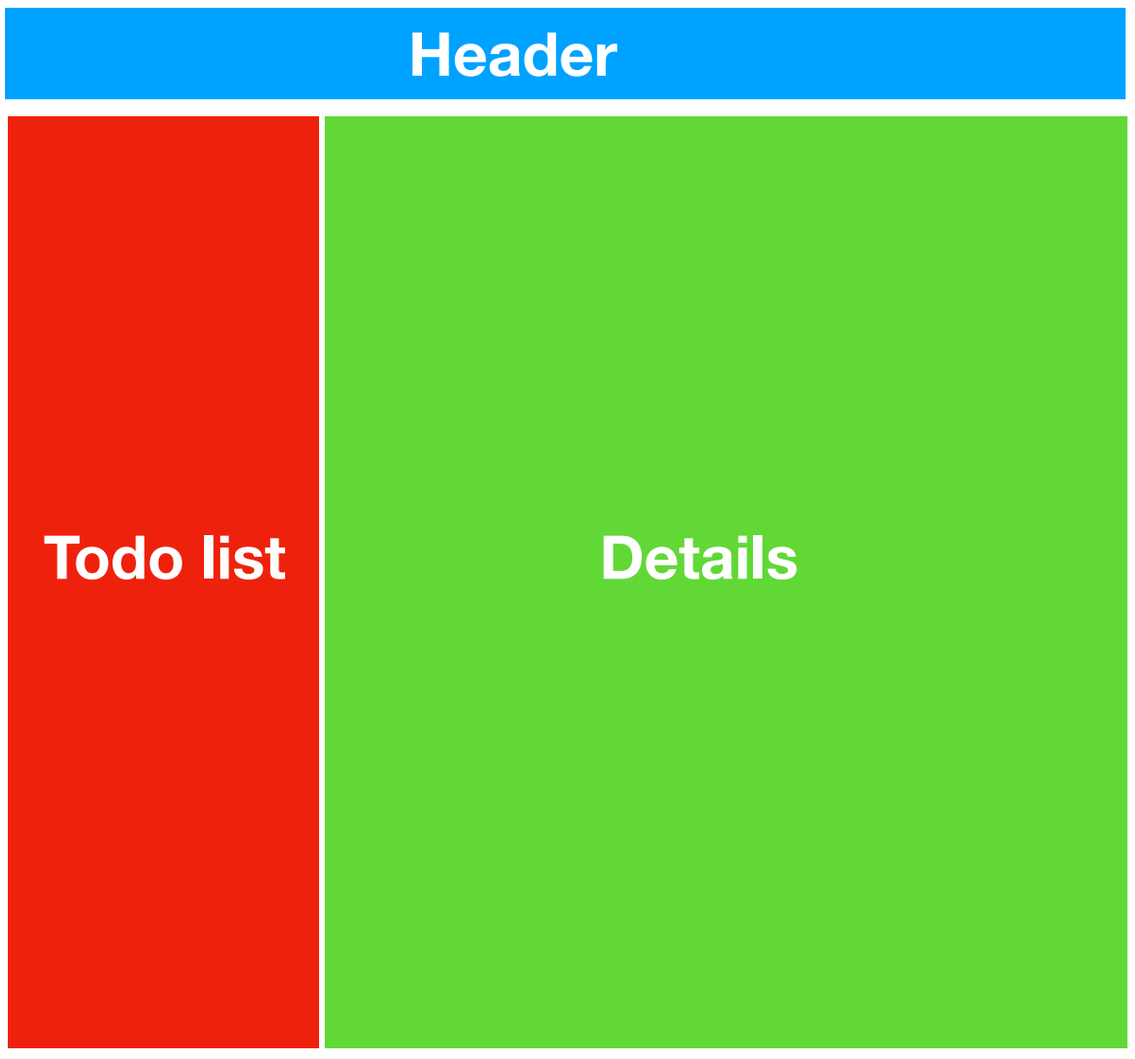
```
.site {
  display: grid;
  grid-template-columns: 15em 1fr;
  grid-template-rows: 1fr minmax(1em, 3em);
  grid-template-areas: "header header"
                       "main sidebar"
                       “footer footer";
}
```

Exercise

Convert CSS to Grid

Small screen - Single column

Large screen - Two column



CSS Grid Solution

Making your PWA Installable

Competing with native

- Users spend more time in native apps than in their mobile web browser. (90% vs 10%)
- Need to have a button on their phone
- It needs to work like an app not just a glorified wrapper around your website

Declaring your manifest

```
{
  “name”
  “short_name”
  “icons”: []
  “start_url”
  “background_color”
  “display”
  “orientation”
  “scope”
  }
```

Getting on the home screen

- Have to be served via https
- Have to have a manifest
- Installation process differs
  - Android presents a banner
  - iOS requires you to go through the share menu

Exercise

Create an app manifest for our application. It should include:

- **Icons found in the icon dir (48, 72, 96, 144, 168, 192)**
- **A name**
- **Background color**
- **Display type**

Feel free to play around with the value

App Manifest

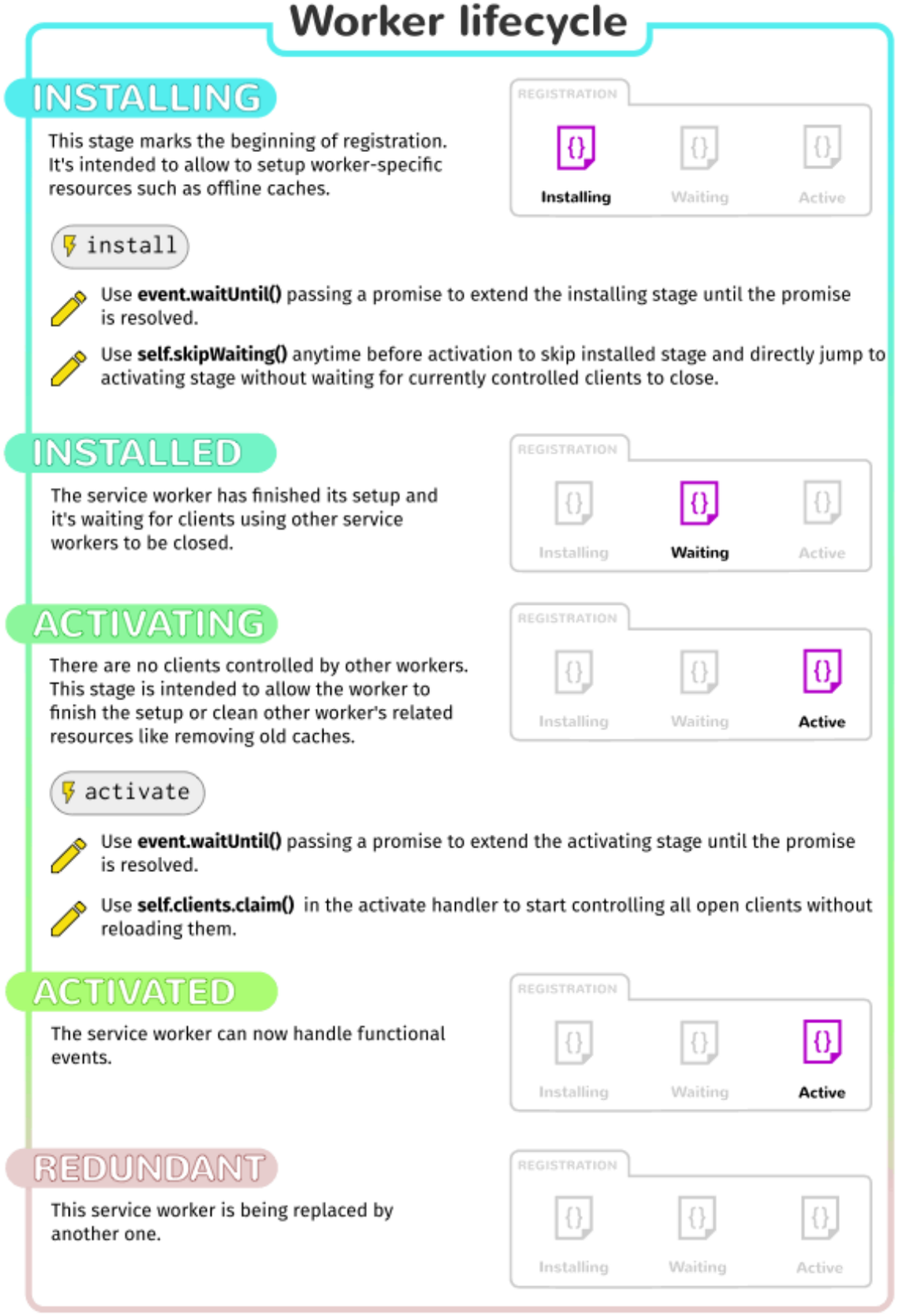
Solution

Creating your first Service Worker

Service Workers

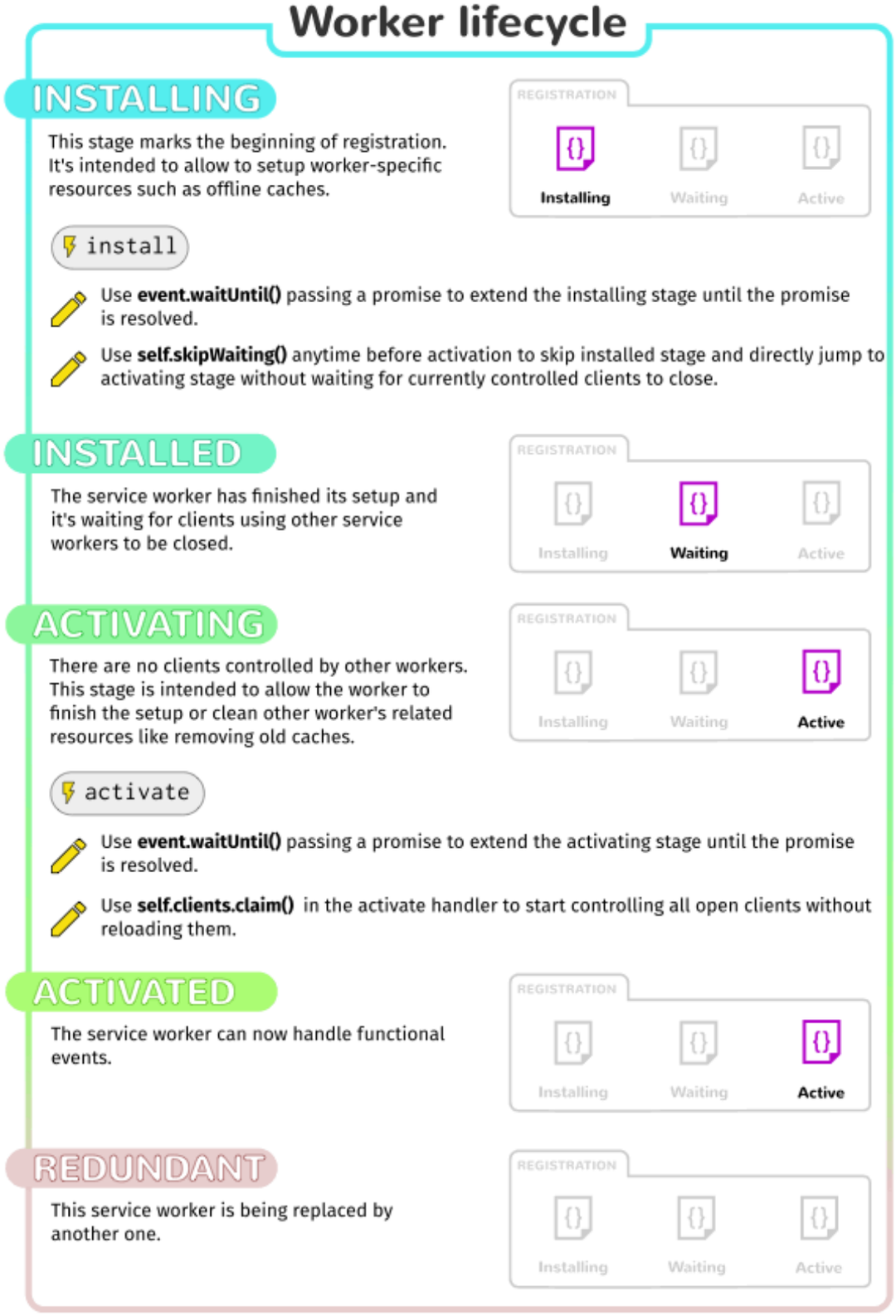
- The real workhorse of PWAs
- Manages the connections to the backend server for you.
- Caches assets for offline use

Lifecycle


Worker lifecycle
INSTALLING
This stage marks the beginning of registration. It's intended to allow to setup worker-specific resources such as offline caches.
REGISTRATION
Installing
Waiting
Active
install
Use event.waitUntil() passing a promise to extend the installing stage until the promise is resolved.
Use self.skipWaiting() anytime before activation to skip installed stage and directly jump to activating stage without waiting for currently controlled clients to close.
INSTALLED
The service worker has finished its setup and it's waiting for clients using other service workers to be closed.
ACTIVATING
There are no clients controlled by other workers. This stage is intended to allow the worker to finish the setup or clean other worker's related resources like removing old caches.
activate
Use event.waitUntil() passing a promise to extend the activating stage until the promise is resolved.
Use self.clients.claim() in the activate handler to start controlling all open clients without reloading them.
ACTIVATED
The service worker can now handle functional events.
REDUNDANT
This service worker is being replaced by another one.

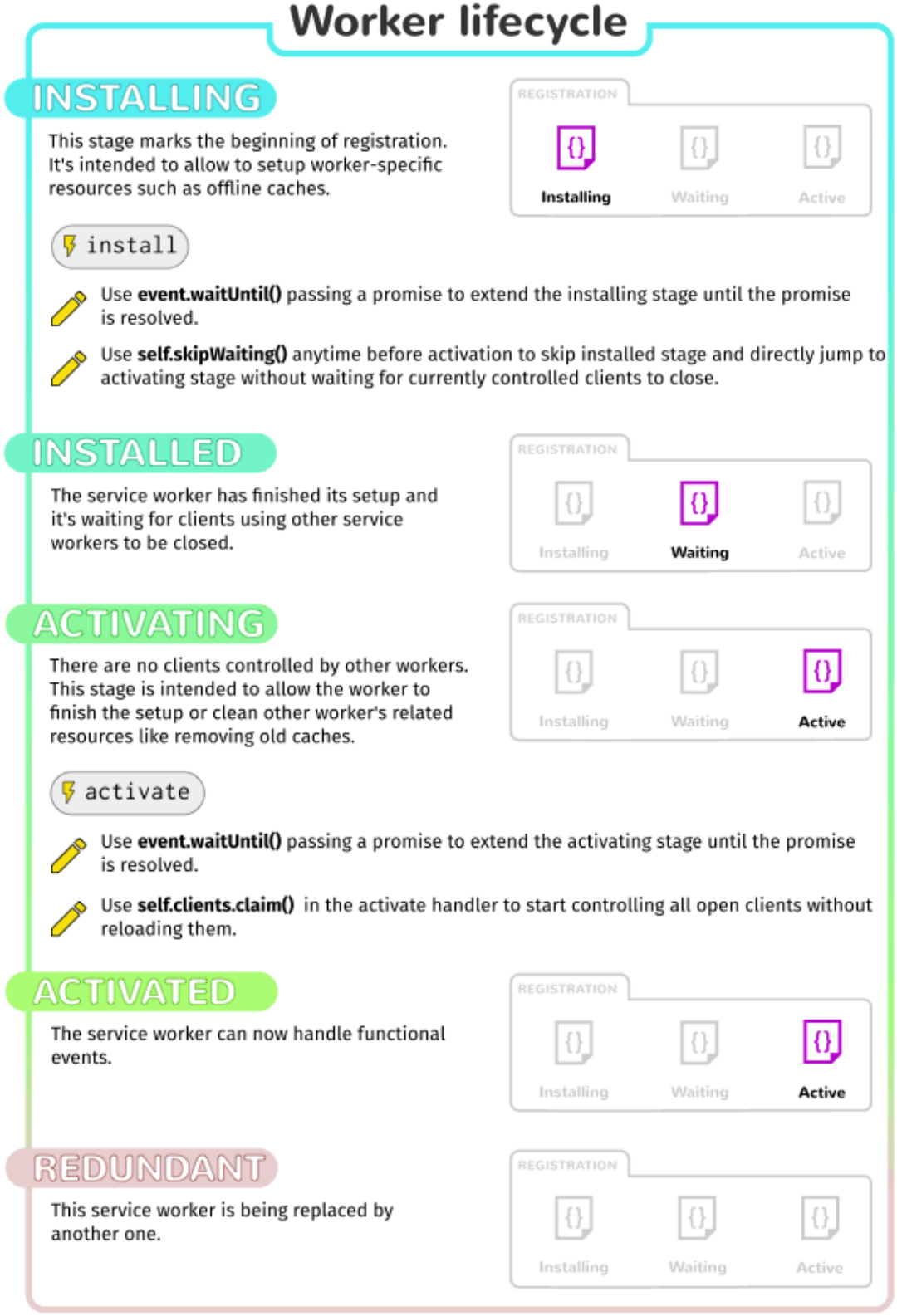
Lifecycle

- **Use progressive enhancement to register your service worker**
- **Create and populate cache**
- **Clean up old cache**
- **Worker is active**



The details

- **Worker is scoped to the path it installed from**
- **Worker doesn’t become active until page visit after it is installed**
- **Version your caches**



Registering a Service worker

```
navigator.serviceWorker.register('./service-worker.js').then
(registration ⇒ {
  console.log('Service Worker is registered!')
})
```

Listening for events

```
self.addEventListener('install', (event) ⇒ {
    …
})
```

- **install**
- **activate**
- **fetch**
- **Push**

Setting up a cache

```
event.waitUntil(
  /* open method available on caches,
     takes in the name of cache as the first parameter.
     It returns a promise that resolves to the instance of
     cache. */
  caches.open(CACHE_NAME)
  .then (cache => cache.addAll(urlsToCache))
)
```

Exercise

Install your service worker

- **Cache all images, css and js during install event, clean up old caches during activate**
- **Dev tools -> Applications -> Caches**

Notes:

- **Put service worker in “/public”**
- **Stop “yarn dev” process**
- **Run “yarn start”**
- **Look in “/dist” for your js/img files**

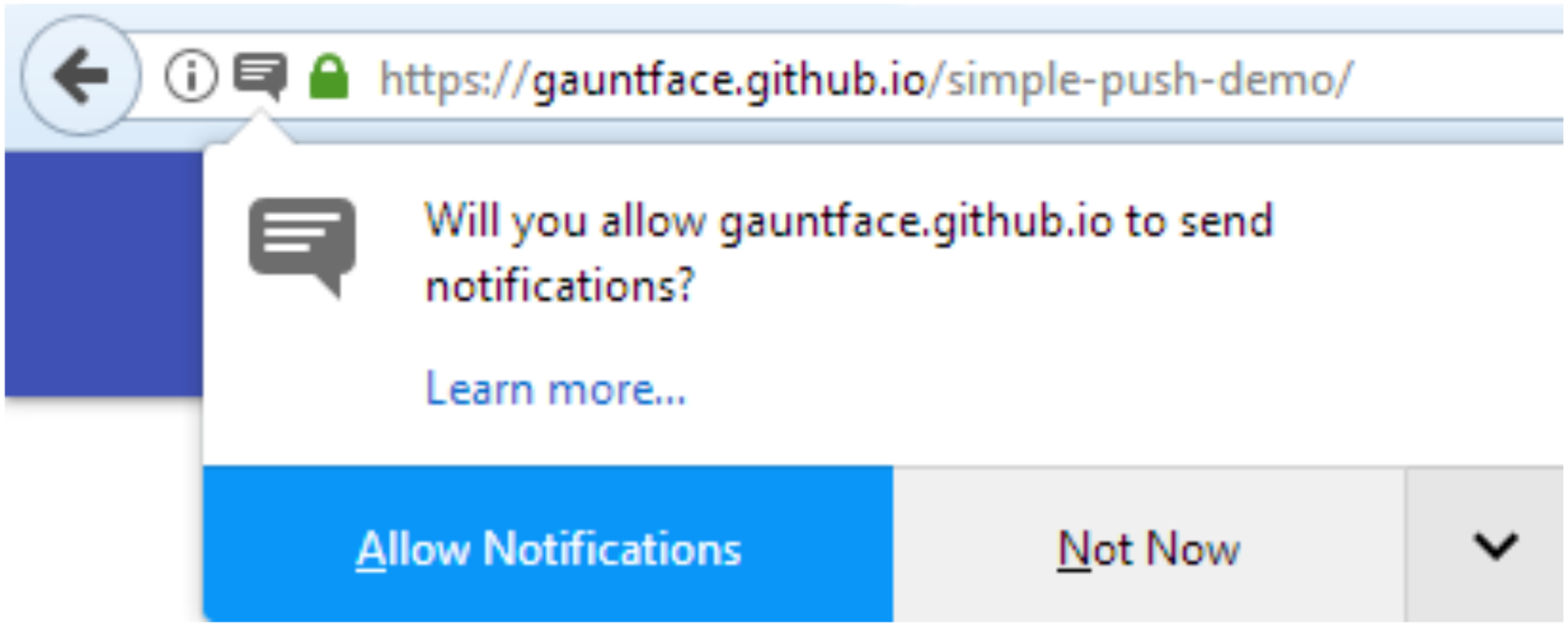
Solution

Pushing All the things

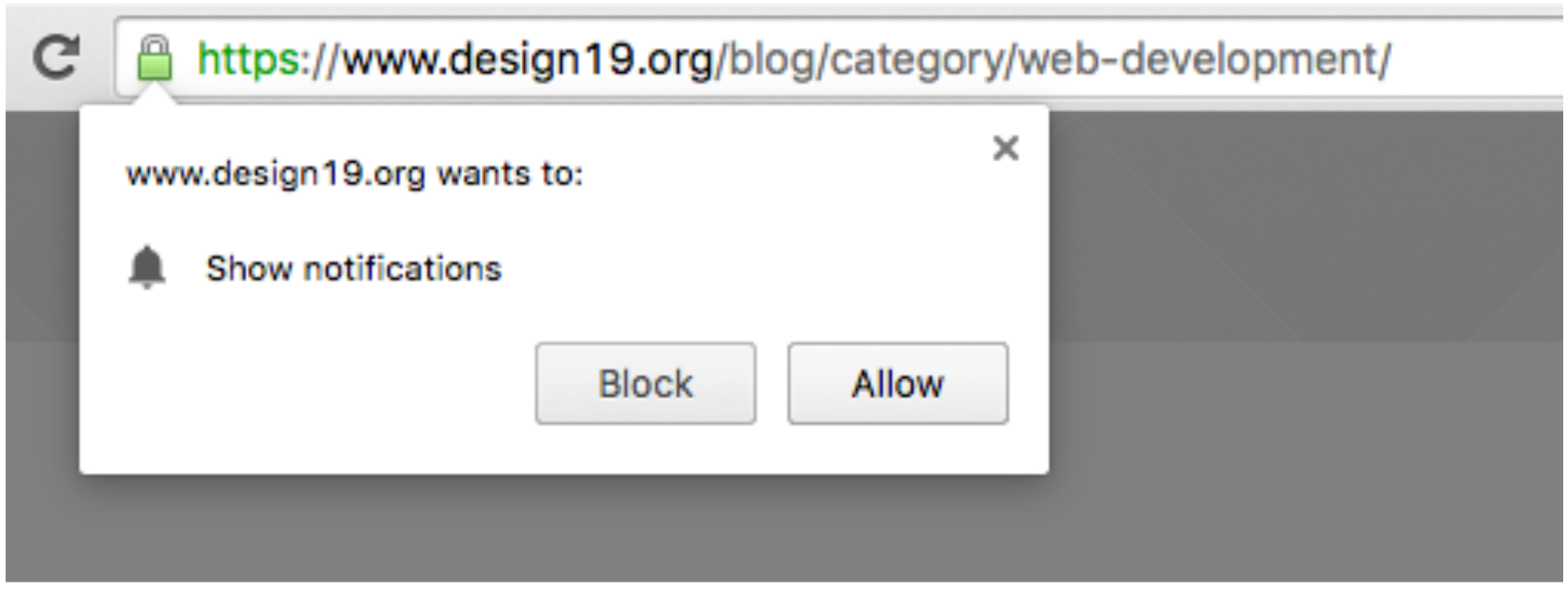
Responsible permission requests

Everyone wants to tell you something

Everyone wants to tell you something



Everyone wants to tell you something



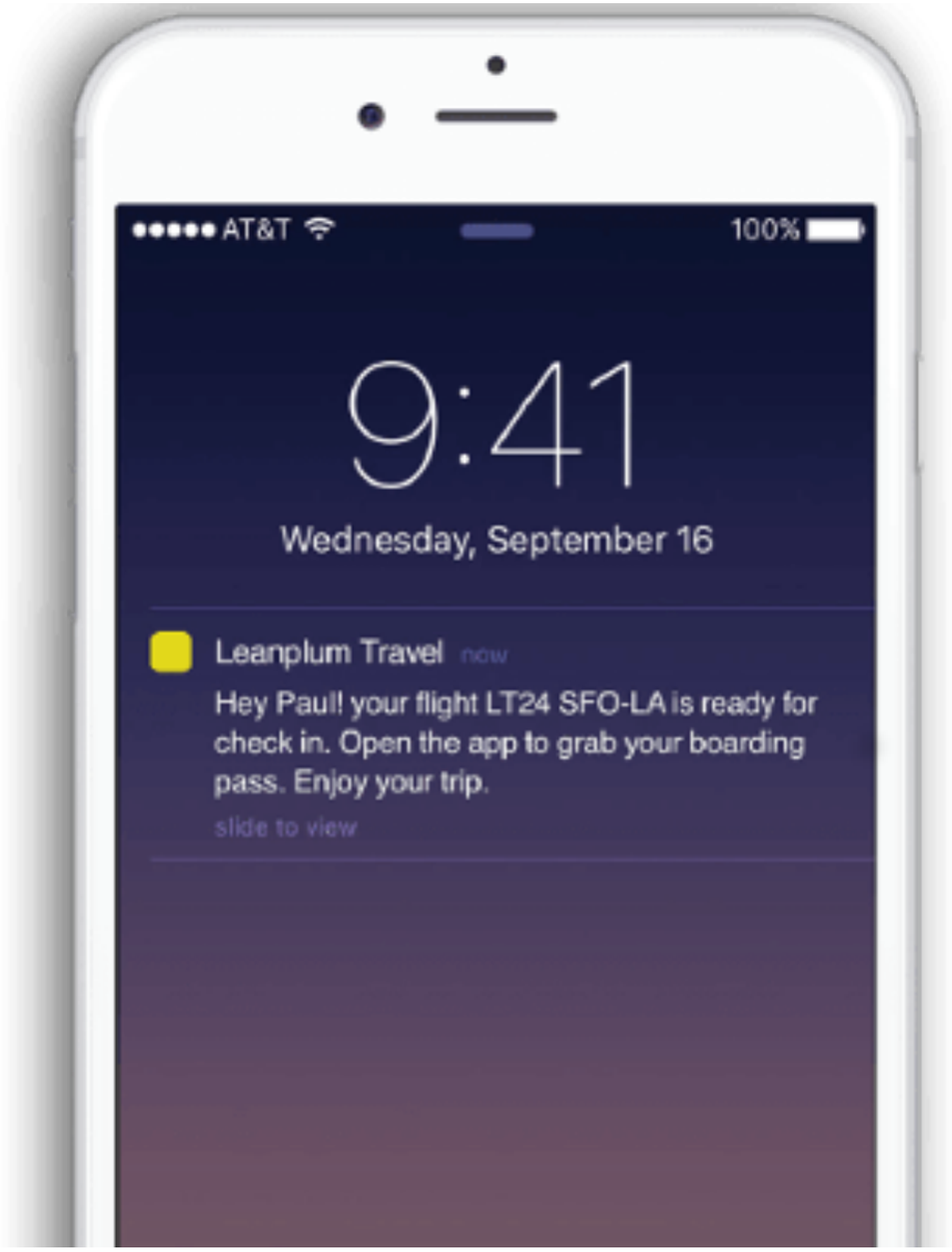
Everyone wants to tell you something

Don’t be that guy

- **Wait to ask for permission to do something until you can show value**
- **Explain in the app before popping the browser request**
- **Only ask for the things you need each time**
- **Be responsible with your usage**

Live push notifications

- Users expect their apps to tell them when something important happens
- You need to think like an app now
- Remember that talk about responsibility?



A couple caveats

- This is a very powerful tool, but key management is essential
- Don’t be a jerk

The subscription

```
{
     "endpoint": “https://fcm.googleapis.com/fcm/send/unique-slug",
     "expirationTime": null,
     "keys": {
          "p256dh": “public-key",
          "auth": “token"
     }
}
```

The setup

```
if (Notification.permission === 'default') {
    Notification.requestPermission().then (result => {
        if (result === 'denied') {
            return
        }

        if (result === 'granted') {
            serviceWorkerRegistration.pushManager.getSubscription()
                .then (subscription => {
                    if (!subscription) {
                        const applicationServerKey = ''
                        serviceWorkerRegistration.pushManager.subscribe({
                            userVisibleOnly: true,
                            applicationServerKey
                        })
                    } else {
                        saveSubscriptionInDB(subscription, userId)
                    }
                })
        }
    })
}
```

The worker

```
self.addEventListener('push', (event) ⇒ {
    let options = {
        body: event.data.body,
        icon: 'images/example.png',
    }
    event.waitUntil(
       self.registration.showNotification(event.data.title, options)
    )
})
```

Exercise

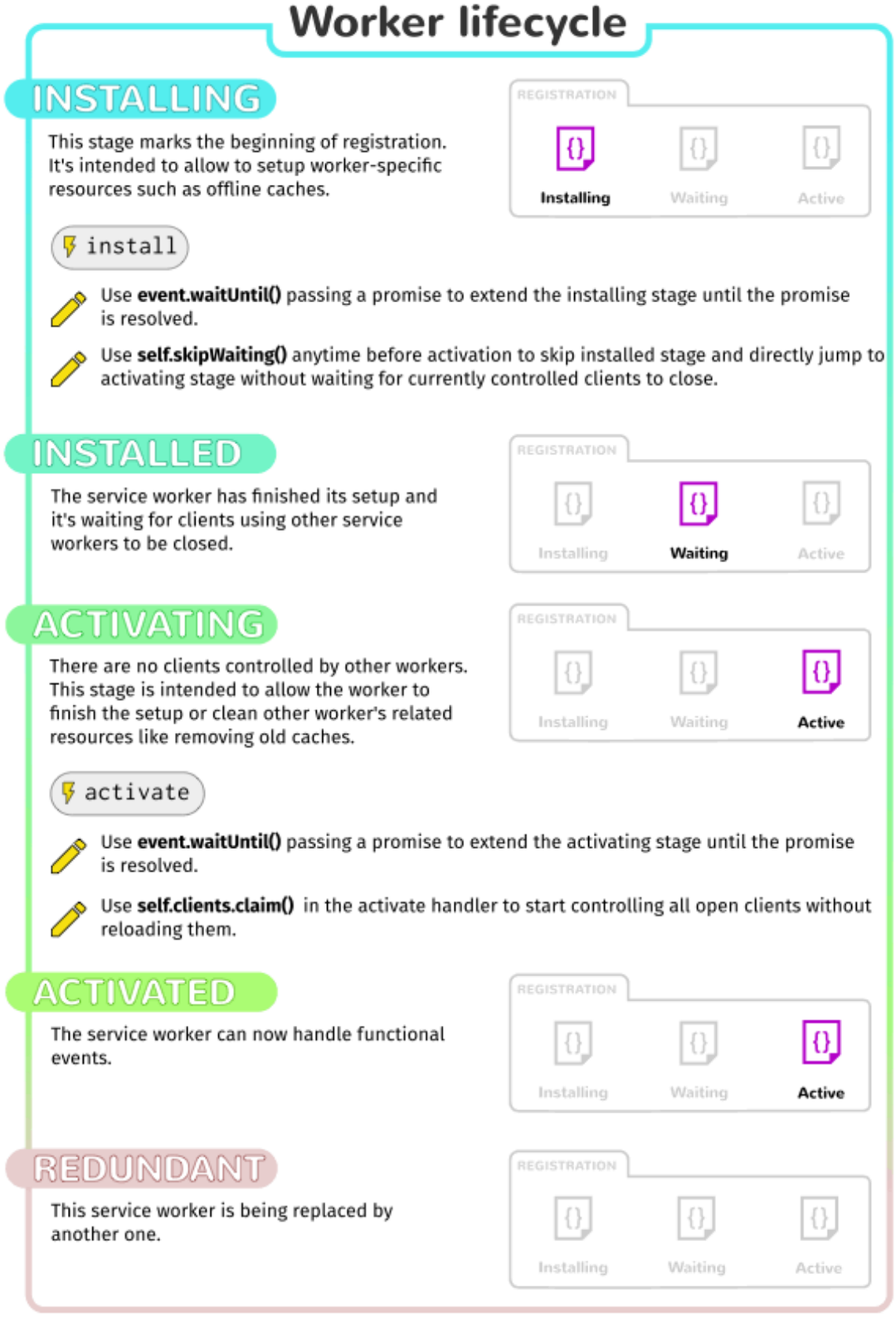
Set up your service worker to handle push notifications

- Request notification permission
- Stub a call with console to save the subscription
- Setup the service worker to push them

Going offline

Lifecycle

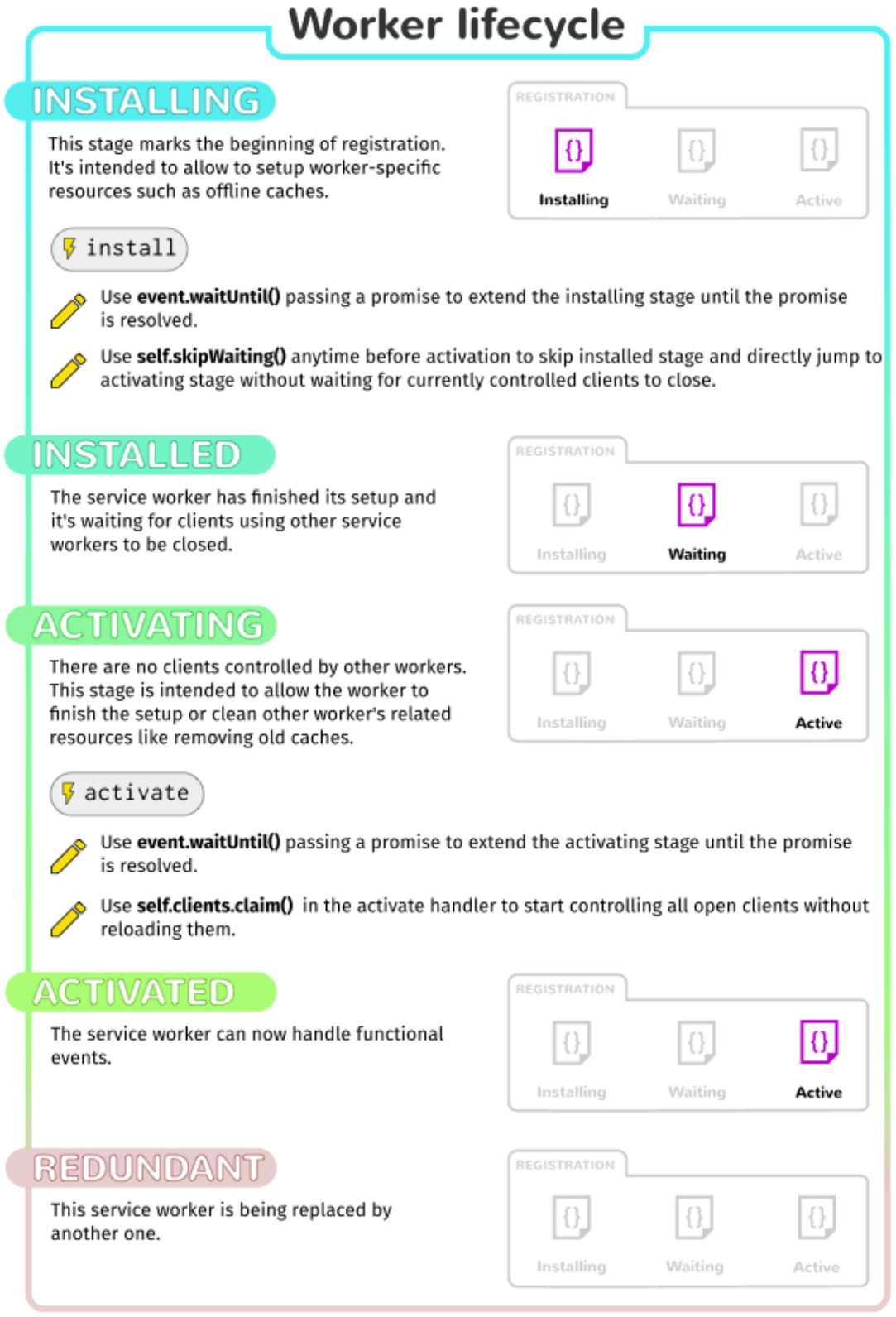
Now that we have a service worker what else can we do with it?



Lifecycle

Now that we have a service worker what else can we do with it?

Act as a client side proxy for your AJAX requests

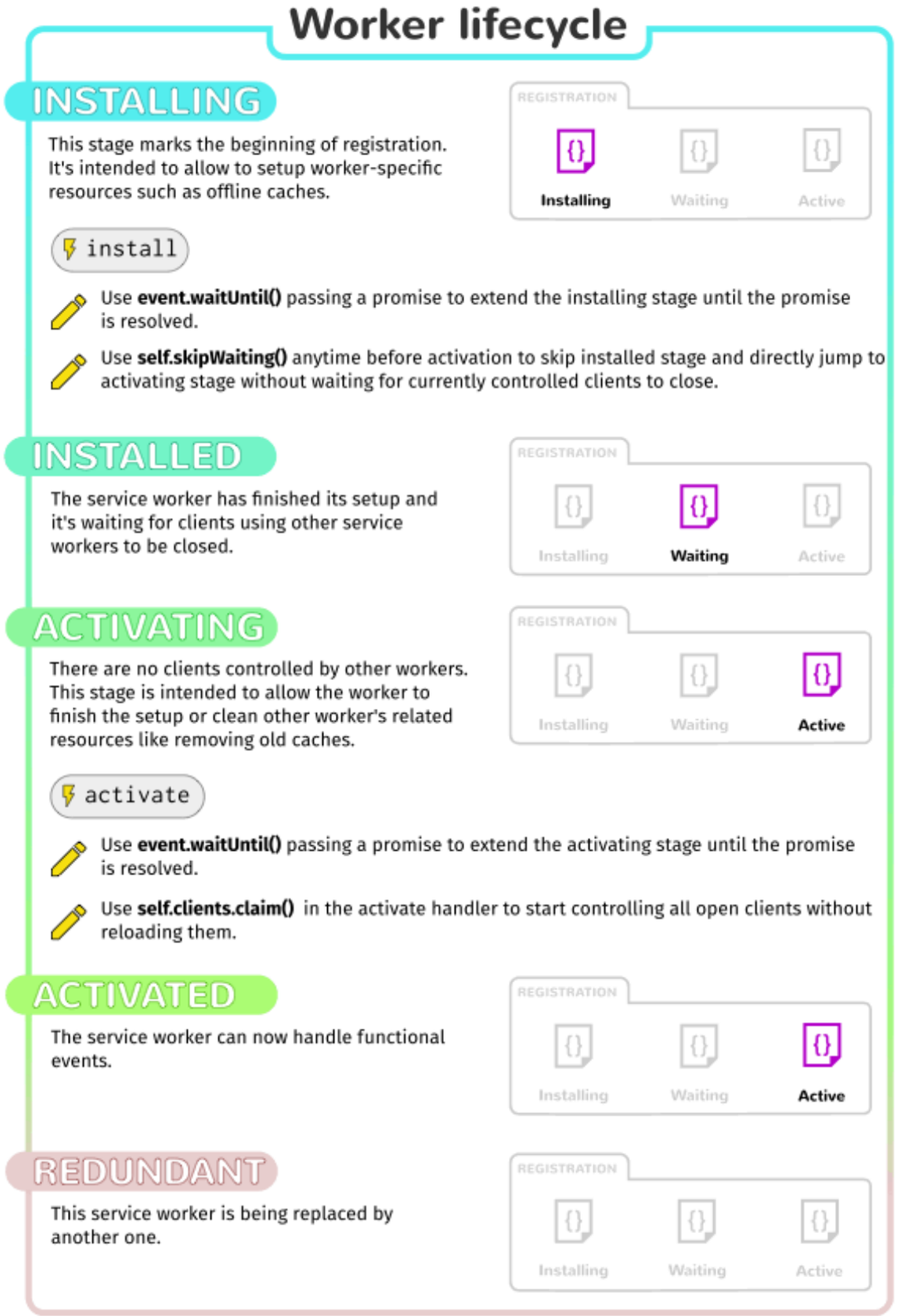


Lifecycle

Now that we have a service worker what else can we do with it?

Act as a client side proxy for your AJAX requests

This means that offline first becomes a real thing. Now you can serve data immediately and refresh in the background.



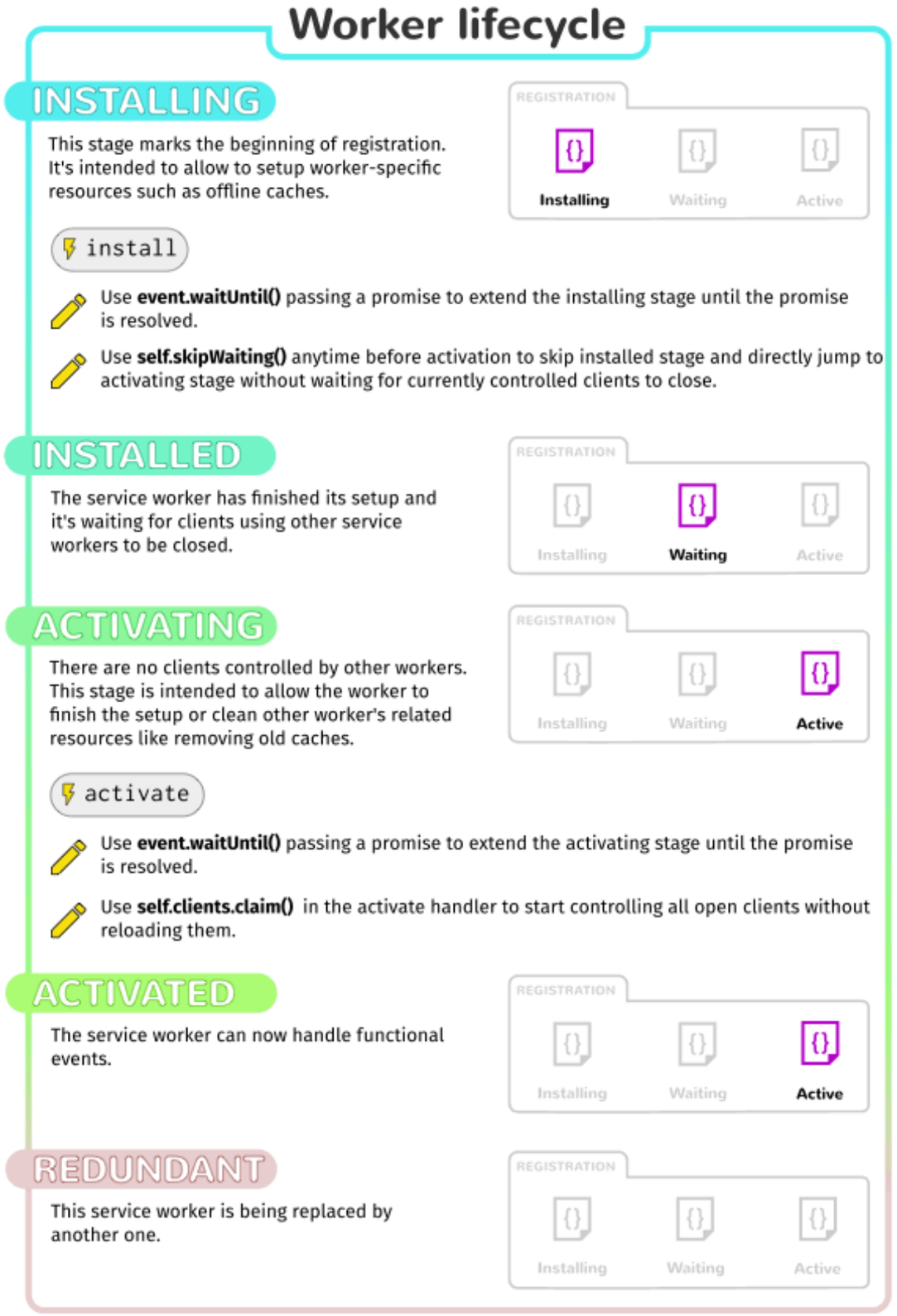
Lifecycle

Now that we have a service worker what else can we do with it?

Act as a client side proxy for your AJAX requests

This means that offline first becomes a real thing. Now you can serve data immediately and refresh in the background.

(And maybe even more, but we might get to that later)



Caching requests

```
self.addEventListener('fetch', (event) ⇒ {
    event.respondWith(
        caches.open('cache')
            .then (cache ⇒ {
                return cache.match(event.request).then (response ⇒ {
                if (response) {
                    return response
                }

                // Didn’t find it use fetch and store store your
                // response with cache put
            })
        })
    )
})
```

Exercise

Set up your service worker to cache requests

- **Think about your cache do you need to invalidate it**

- **Don’t forget to clone your request/ response**

Going Offline

Solution

Recap

- PWAs are about delivering an App like experience via real web API
- Need to think like a native developer
- Don’t be a jerk
- More possibilities opening all the time

Oh and…

- Use tools to help you, you don’t have to do it all by hand
- Webpack sw-precache
- manifest generators
- Node web push

Resources

- Smashing Magazine
- The Service Worker Cookbook
- Google Developer - PWA
- MDN

Thanks!