ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Further Topics of Performance Optimization

Hosted by Alex Thalhammer

Outline

1. Don't use Angular resolvers (if you ask me)
2. Smart vs Dumb Components
3. API Architecture
4. RxJS & NgRx
5. Web Workers for heavy calculations
6. Service Worker / PWA
7. Scheduling
8. Building with Nx
9. Use latest Angular

#1: Don't use Angular resolvers

- Better to show the title and everyting possible, even just the frame
- Instead use local spinners where data is being loaded

#2: Thought experiment

- What if <flight-card> would handle use case logic?
  - e.g. communicate with API (thru a service)

- Number of requests ==> Performance?

- Traceability?

- Reusability?

Smart vs. Dumb Components

#2: Smart vs. Dumb Components

Smart	Dumb
• Use Case controller • Container	• Independent of Use Case • Reusable • Leaf

#3: API Architecture

- Try to minimize API calls
  - E.g. fetch data in list not list item
  - If possible aggregate data in backend, not frontend
- Think about caching API calls
  - If possible, maybe valid for limited time only
- Maybe use GraphQL?

#4: Use RxJS & NgRx

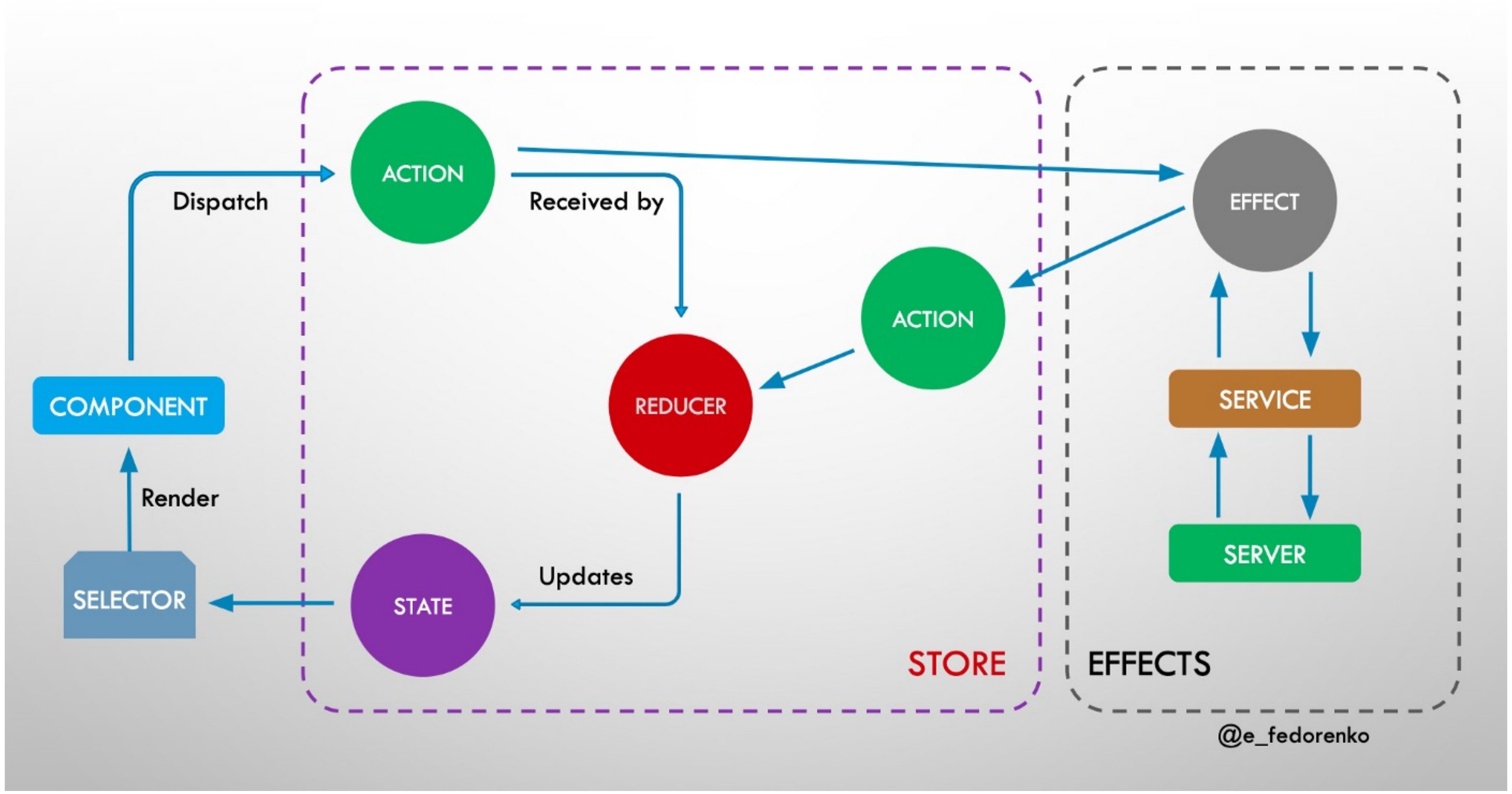
- Use RxJS properly
  - Share hot observables where possible
  - Pipe operaters
  - Use async pipe
  - Manage subscriptions

- Use State Management (NgRx preferred, else NGXS)
  - By using Redux libraries properly, you can improve its performance, by reducing the number of events that occur during data communication

#4: State management options

- Global
  - NgRx (better)
  - NgXS
- Local
  - State Services / Facades with BehaviourSubjects or Signals
  - @rx-angular/state

#4: Global state management (NgRx)



https://medium.com/angular-in-depth/how-i-wrote-ngrx-store-in-63-lines-of-code-dfe925fe979b

#5: Web Workers for heavy calculations

- Problem: JS is single threaded, how to do heavy calculations?

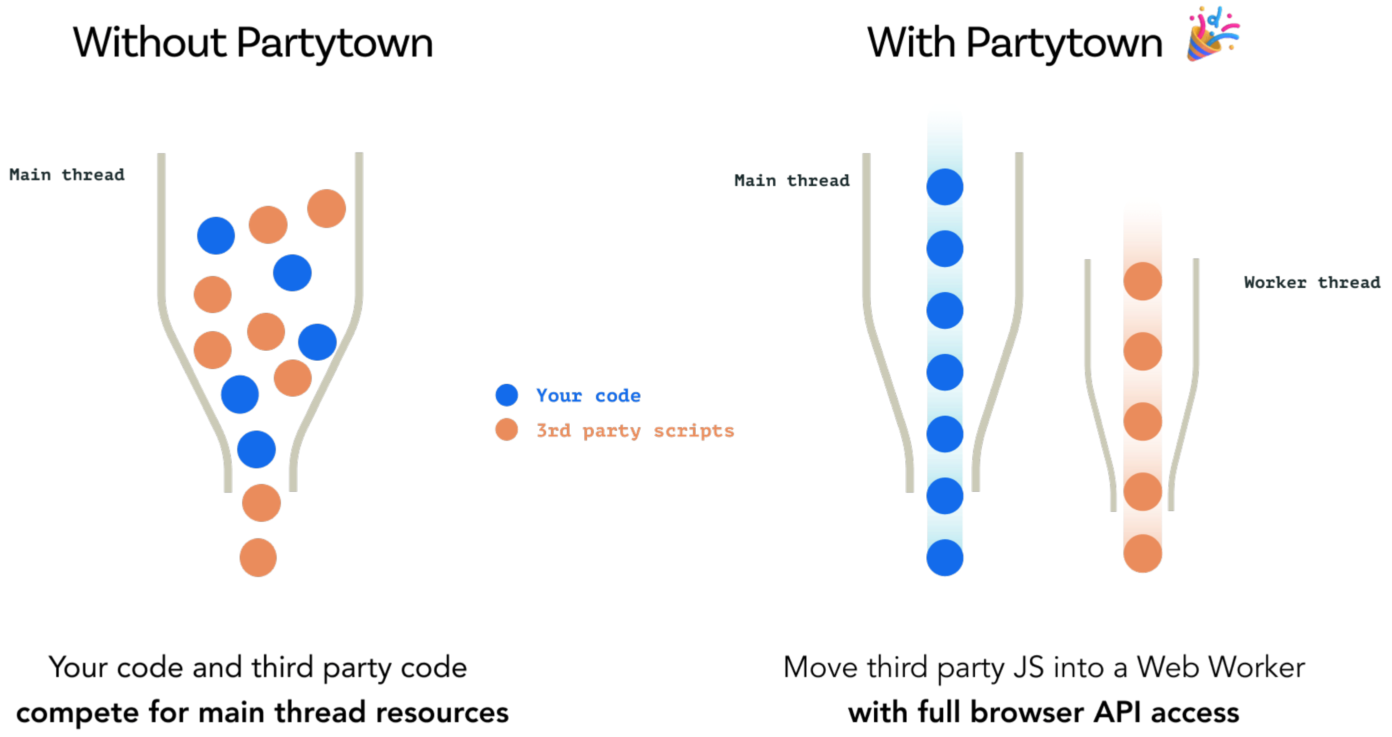
- Solution: Delegate to web worker, it will create a new thread called the Worker Thread that will run a JS script parallel to the main thread

#5: Web Workers – Use cases

- Import external scripts
- Make XMLHttpRequest / API requests
- Use setTimeout() and setInterval()
- Spawn other workers
- Use IndexedDB, Notifications API, Web Crypto API, WebAssembly, WebSockets, WebGL, OffscreenCanvas, ImageData…
- Terminate themselves when you deem they are no longer needed
- …

#5: Web Workers – Implementations

- Worklet API
- partytown
- Comlink?
- …



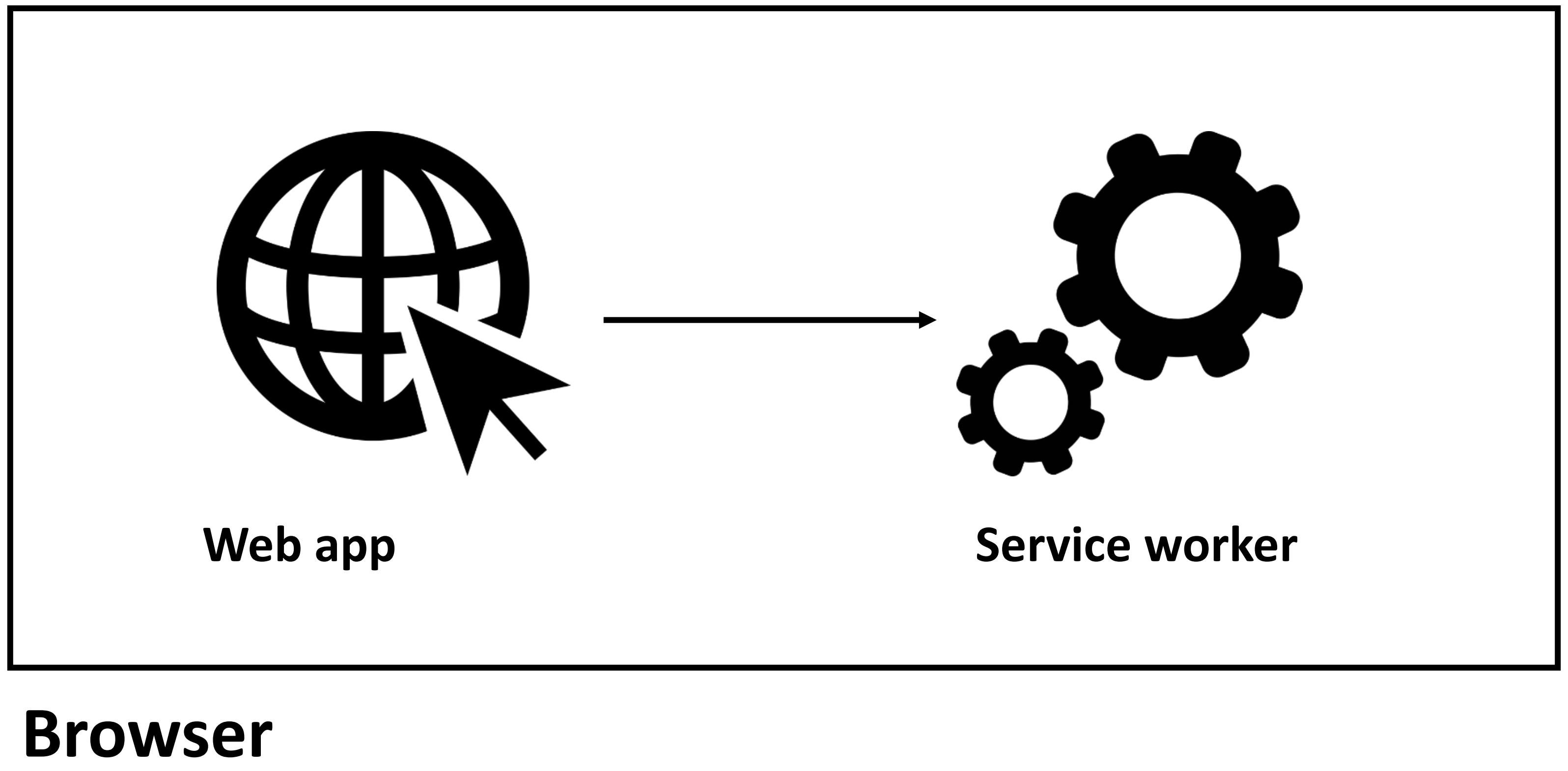
Your code and third party code **compete for main thread resources**

Move third party JS into a Web Worker **with full browser API access**

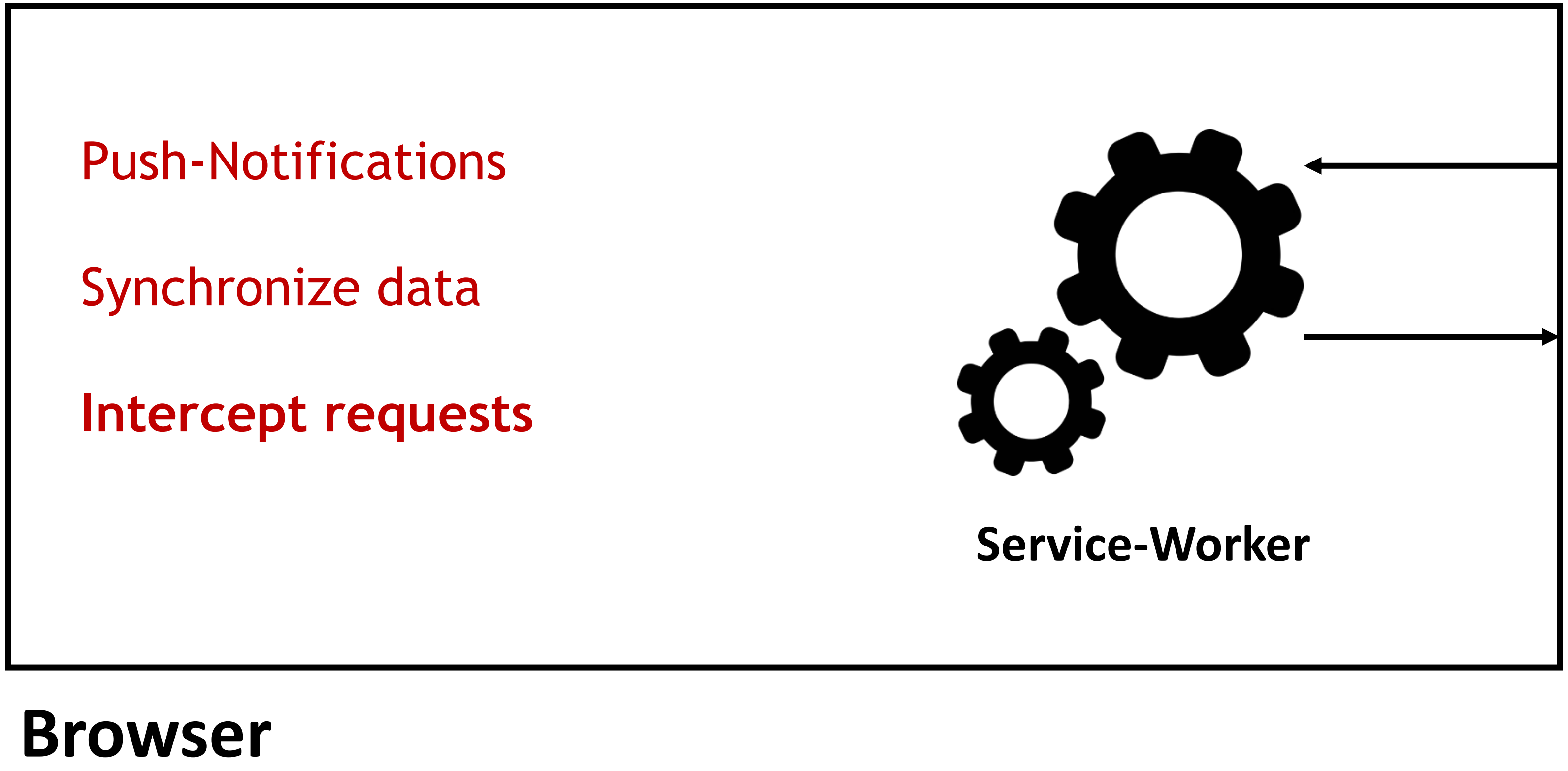
#6: Service Workers (PWA)

- Handle offline state (no connection)
- Web push notifications (new in iOS)
- Proxy or serving HTTP requests
- Background code execution
- Process payments
- …

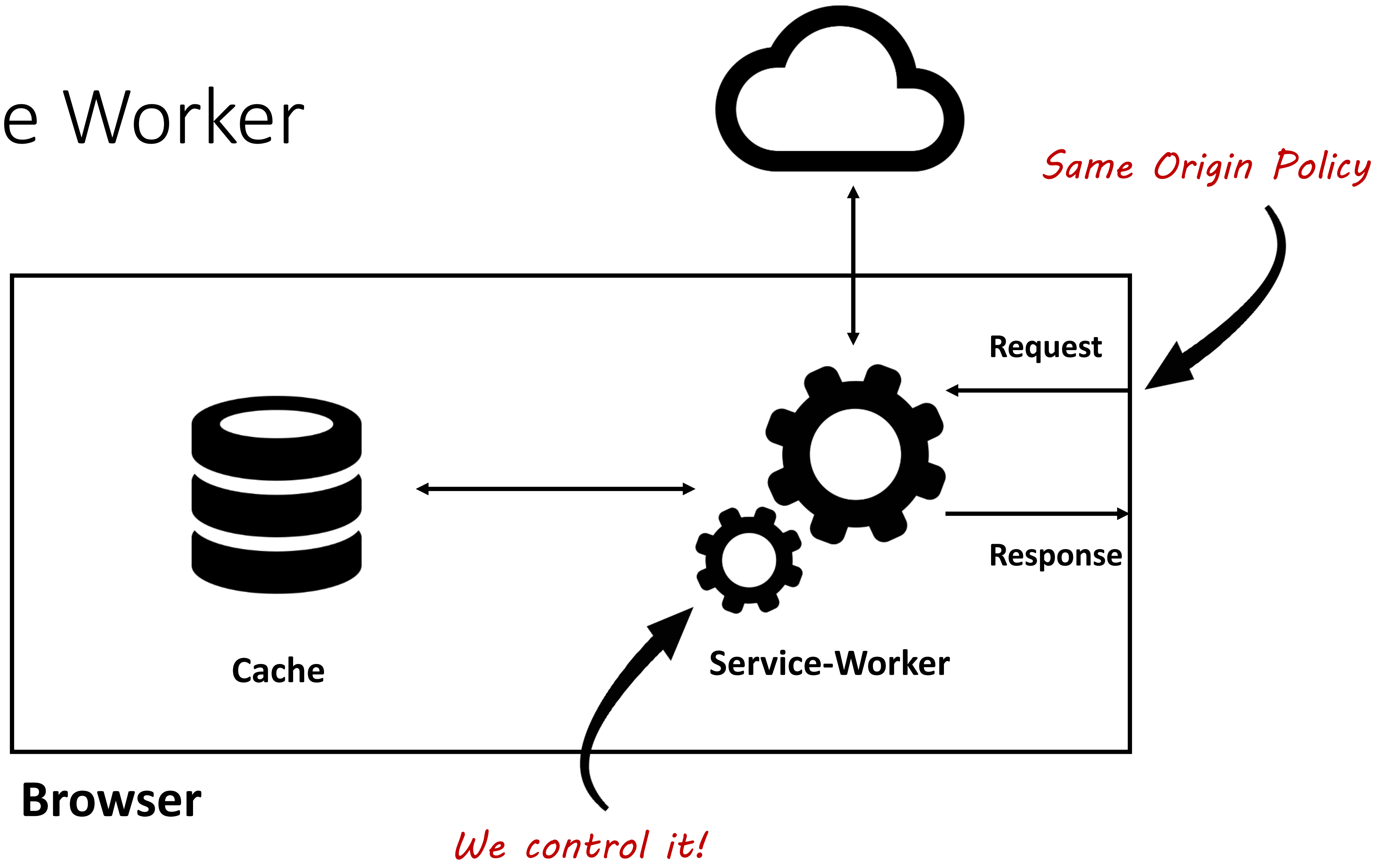
Service Worker



Service Worker



Service Worker



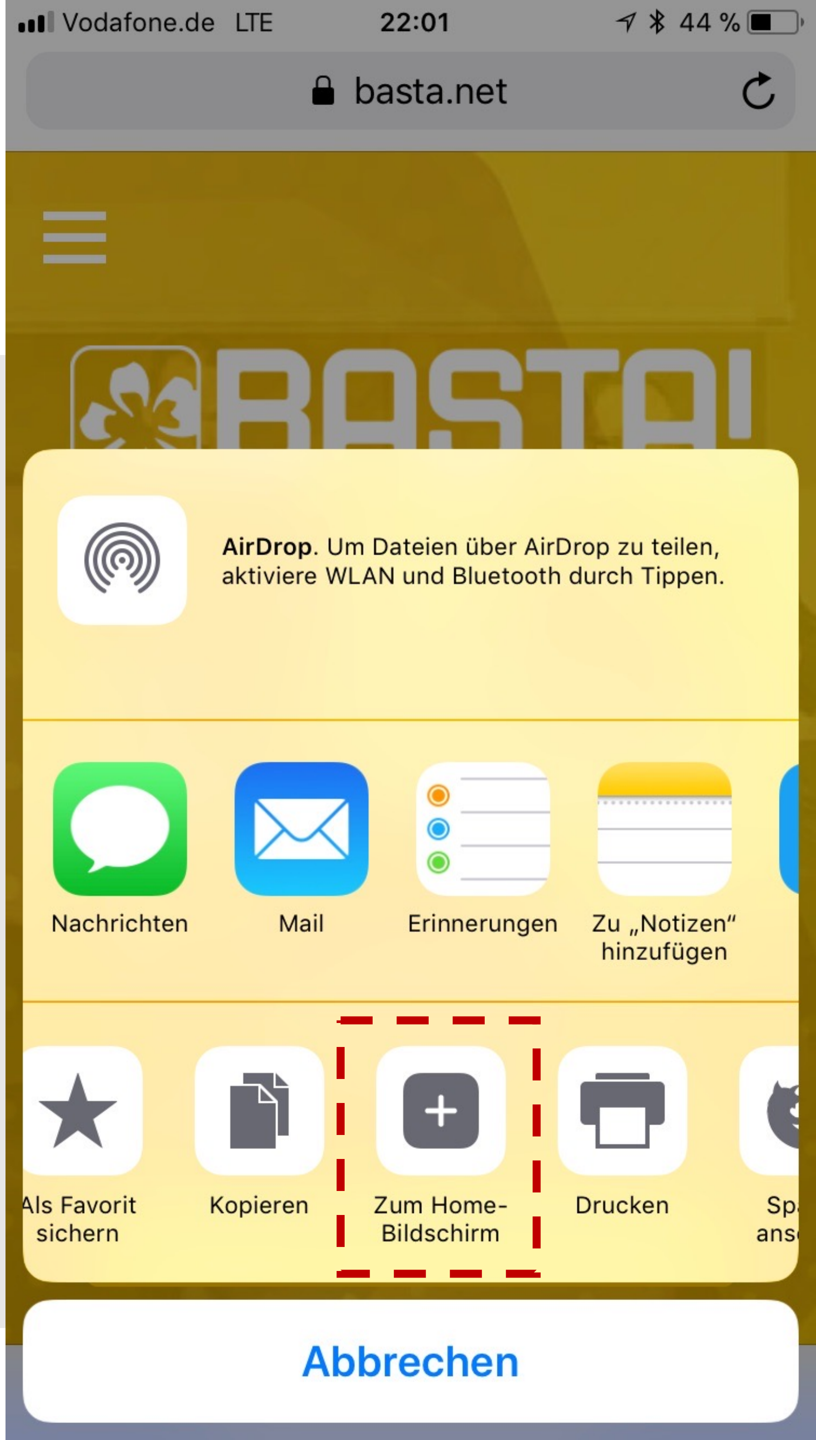
Cache Strategies

- Cache only
- Network only
- Try Cache, fallback to Network
- Try Network, fallback to Cache
- …

Add to home screen

Web App Manifest

```
{
  "name": "Hotel PWA-Demo",
  "short_name": "Hotel",
  "icons": [{
        "src": "images/touch/icon-128x128.png",
        "sizes": "128x128",
        "type": "image/png"
     }, […] ],
  "start_url": "/index.html?homescreen=1",
  "display": "standalone",
  […]
}
```



ANGULAR ARCHITECTS INSIDE KNOWLEDGE SOFTWARE ARCHITECT

@angular/pwa

- Installiert @angular/service-worker
  - npm install
  - Importiert Angular-Modul
  - Generiert ngsw-config.json
  - Generiert Web App Manifest
- ng add @angular/pwa

#7: Scheduling

- Use setTimeout() to delay work
- Use setInterval() to invoke tasks continously
- Don't forget to clearTimeout() and clearInterval() on destroy
- Can lead to unwanted Change Detection

#8: Building with Nx

- For bigger / enterprise Apps use @nx

- Nx is a 3rd party extension for Angular CLI supporting
  - Monorepo workspace
    - Split App(s) into buildable parts / libs
    - Only recompile changed parts (both during serve & build)
    - Possible to have a cloud build cache
  - Other features like
    - Schematics / generators
    - Access restrictions
    - Dependy graph
  - Out-of-the-box support for JEST, (Cypress | Playwright) & Storybook

#9: Use latest Angular

- Try to update to latest version
  - My recommendation: Wait for X.1.0 or X.2.0
- From v. 12 – 15 migration should be easy (and automatic)
- Caution with v. 16: ViewEngine support dropped for libraries
- Use https://update.angular.io

Recap

1. Don't use Angular resolvers (if you ask me)
2. Smart vs Dumb Components
3. API Architecture
4. RxJS & NgRx
5. Web Workers for heavy calculations
6. Service Worker / PWA
7. Scheduling
8. Building with Nx
9. Use latest Angular