Frontend Monolithen: Rette sich wer kann!

In 2016 the average webpage size passed the size of Doom

ronan cremin
@xbs

In about 7 months average web page size will be same as Doom install image.

Well done us! Onwards & upwards!

5:48 PM - Jul 30, 2015

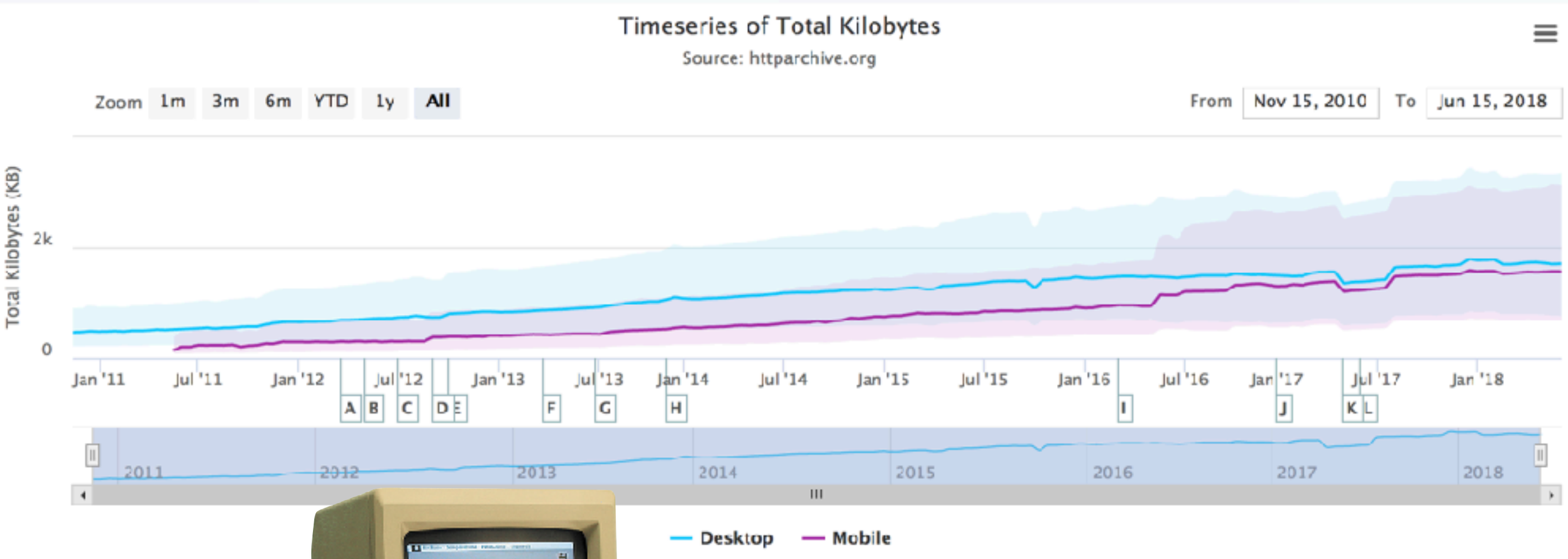
1,761 3,907 people are talking about this

https://mobiforge.com/research-analysis/the-web-is-doom

Web-Page Obesity

MEDIAN DESKTOP
1709.4 KB
▲265.5%

MEDIAN MOBILE
1565.8 KB
▲981.4%



The first Mac had
128k of RAM in 1984.

Web: Size does Matter!

The Cost of JavaScript

http://thedomain:80/app#/first

Browser

Server

initial request
http://thedomain:80/app

html-shell

assets

optional ajax request
http://thedom

Network Cost

html-shell is loaded

"navigate" to "first" screen

CPU Cost

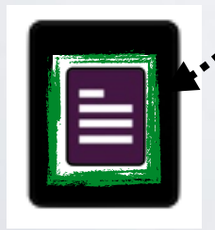
with JS

http://thedomain:80/app#/**first**

navigation

"navigate" to "second" screen

load data & render screen with JS



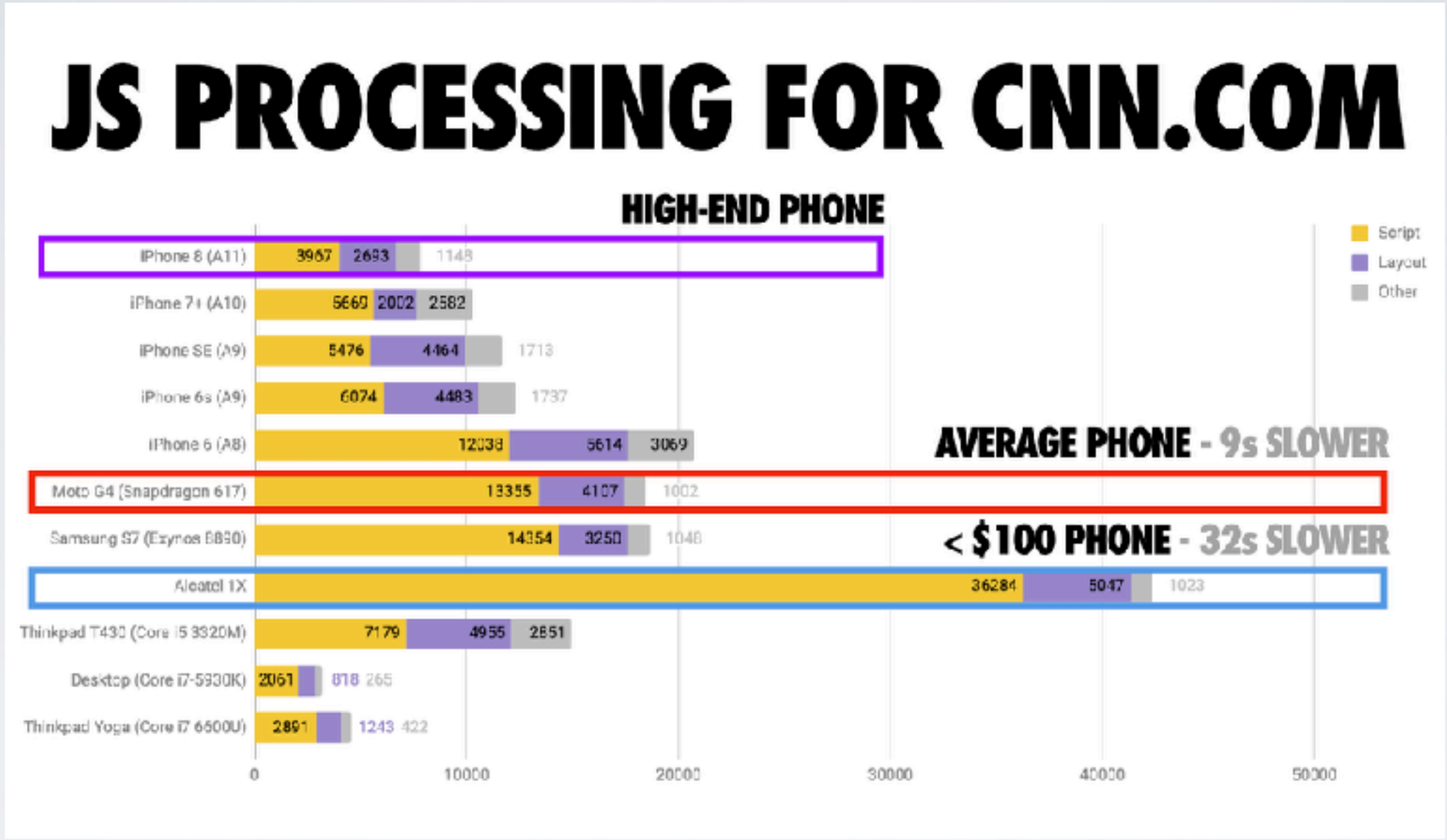
http://thedomain:80/app#/**second**

optional ajax request
http://thedomain:80/api/data/second

data

bootstrapping app

The Cost of JavaScript

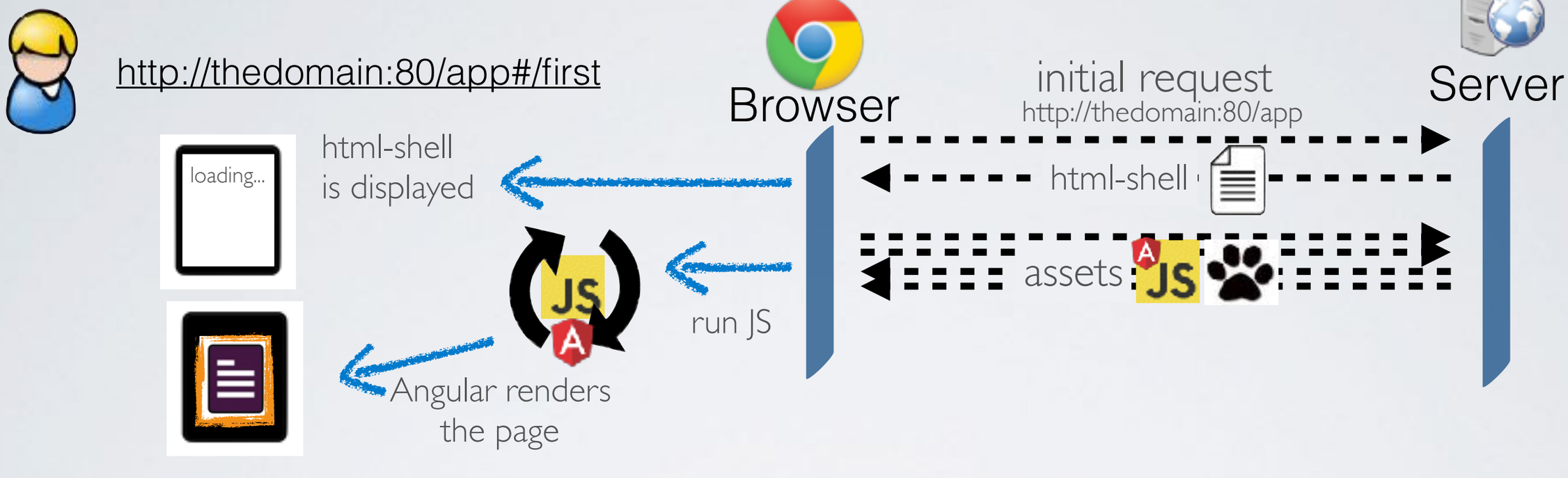


For mobile, aim for a JS budget of < 170KB minified/compressed.

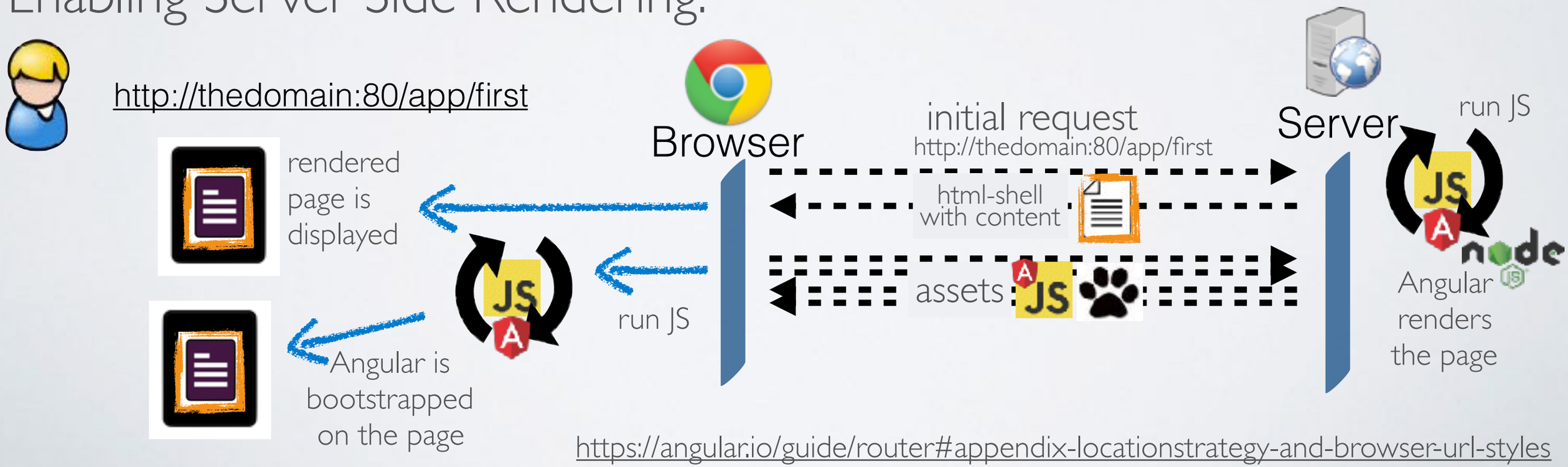
https://medium.com/@addyosmani/the-cost-of-javascript-in-2018-7d8950fbb5d4

Server Side Rendering

Traditional SPA:

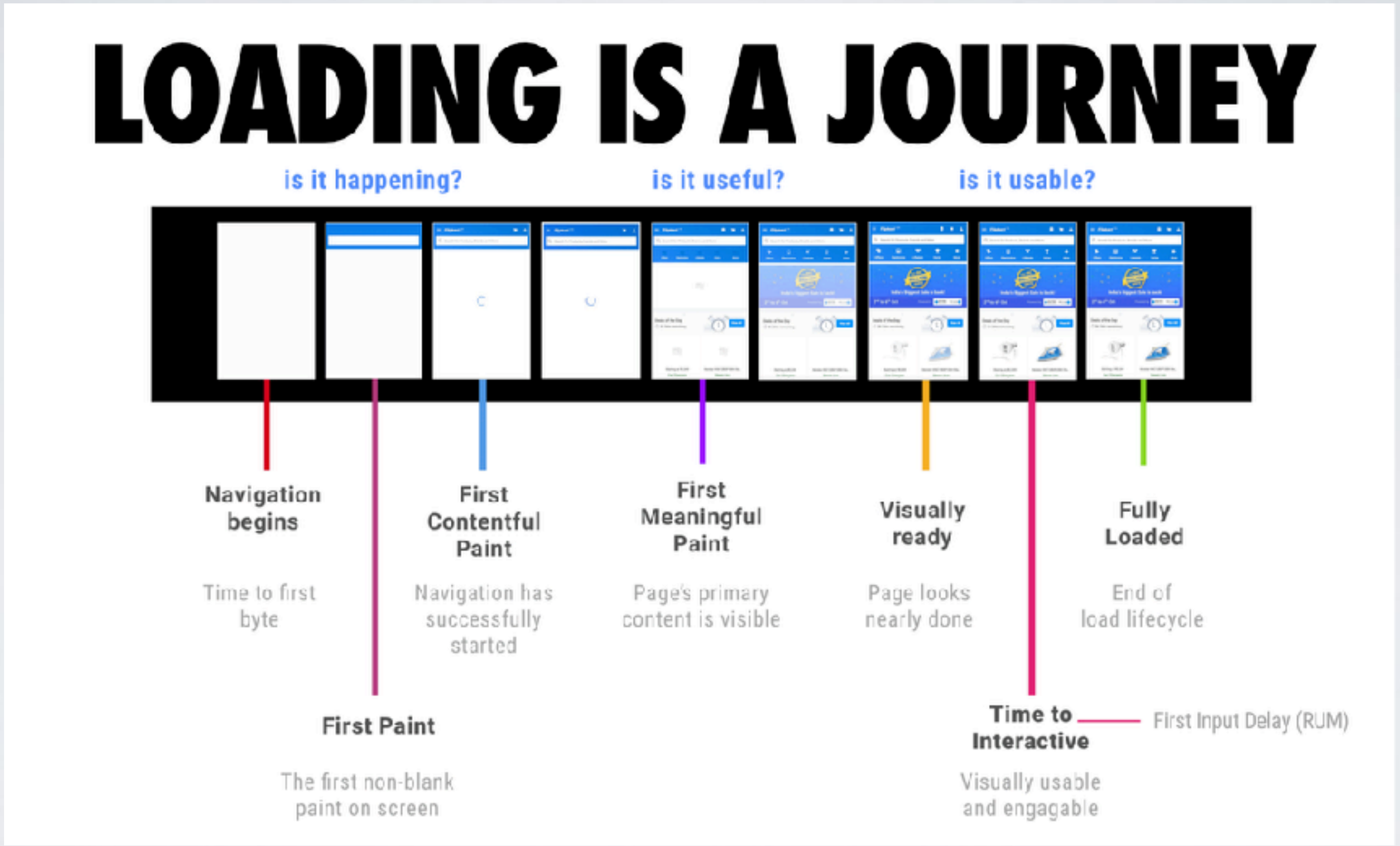


Enabling Server Side Rendering:



https://angular.io/guide/router#appendix-locationstrategy-and-browser-url-styles

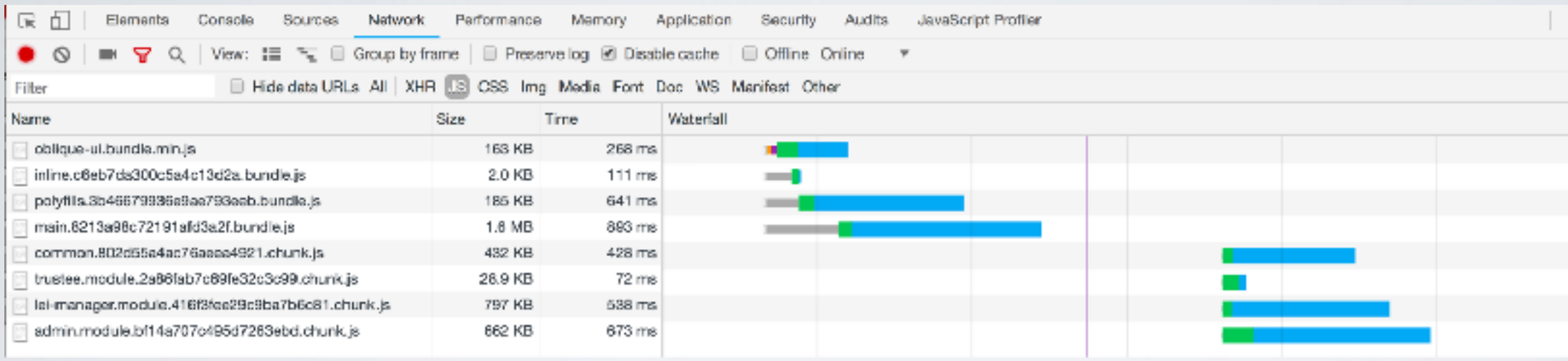
Time to Paint vs. Time to Interactive

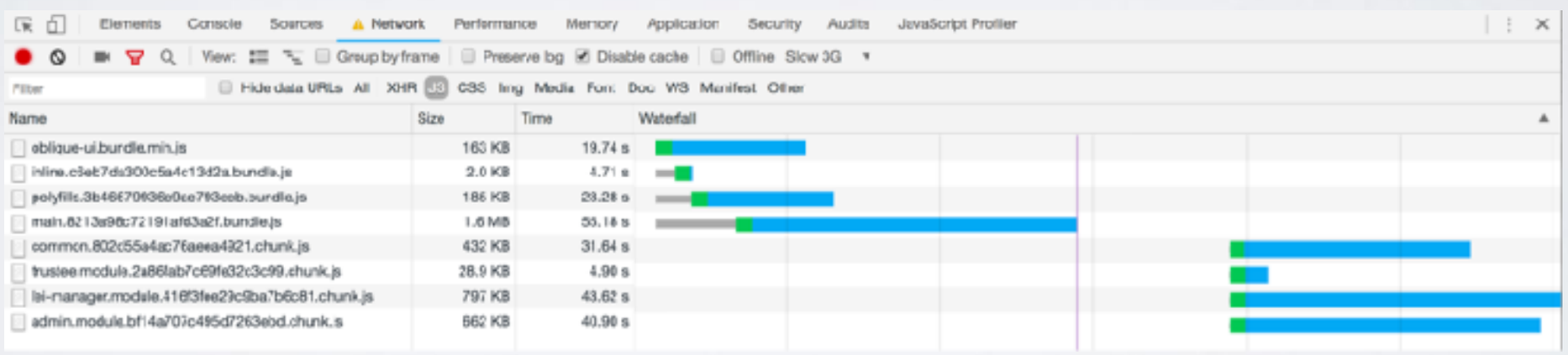


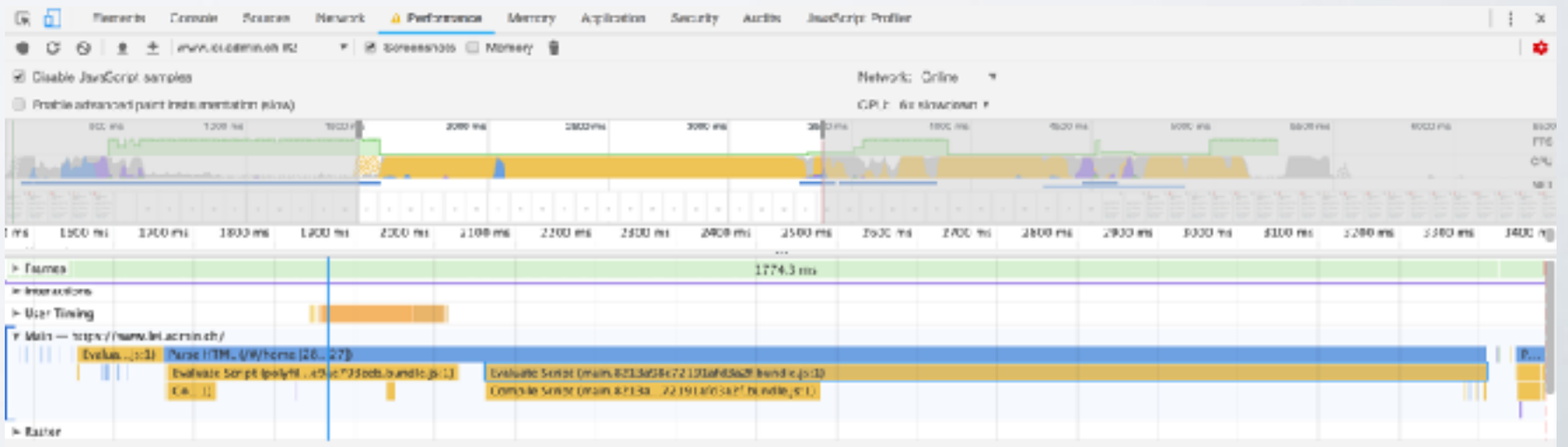
https://medium.com/@addyosmani/the-cost-of-javascript-in-2018-7d8950fbb5d4

DEMO:

https://www.lei.admin.ch/







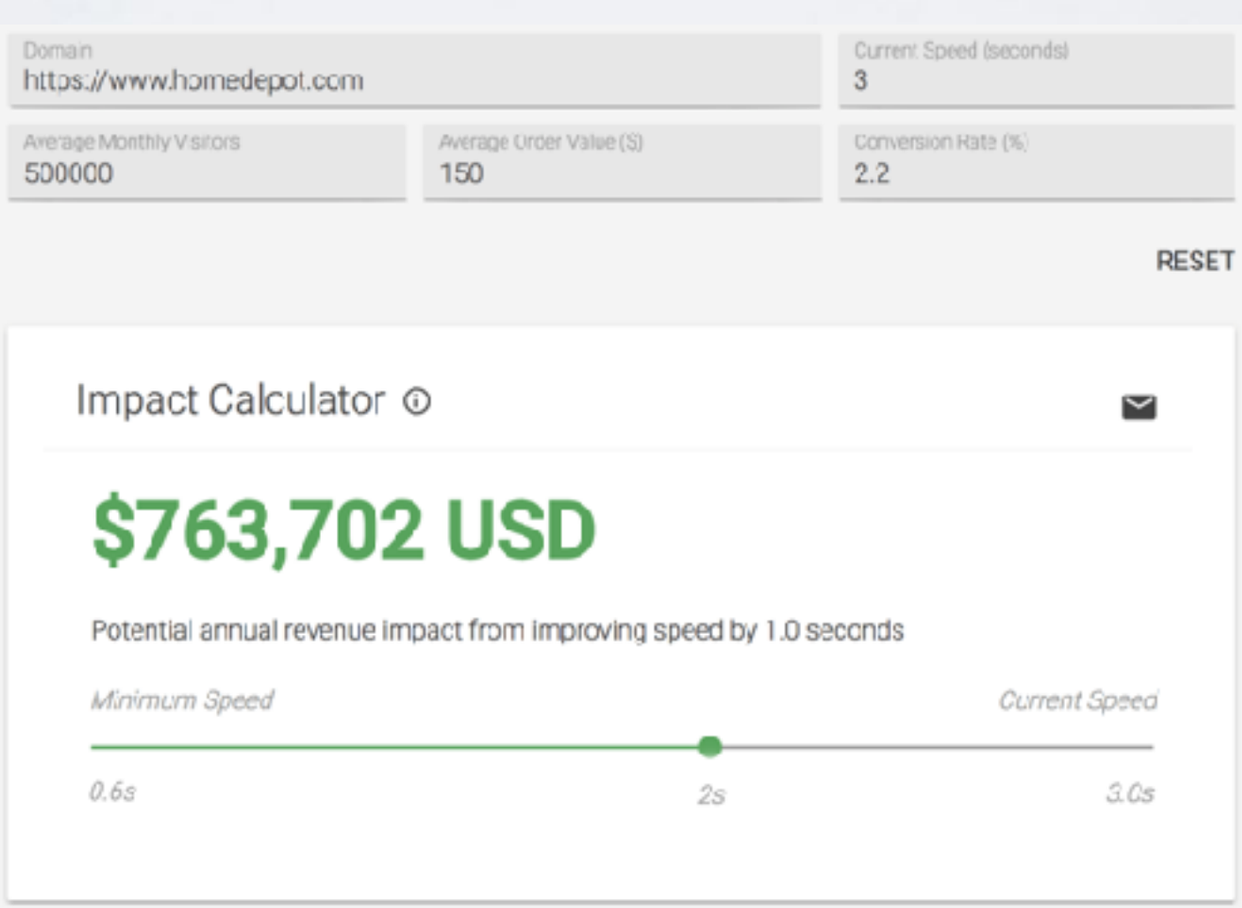
Speed == Money

Amazon did tests that showed they would lose $1.6 BILLION every year if they slowed down by just one second.

https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales

Walmart Study: For every 100 ms of improvement, they grew incremental revenue by up to 1%.

https://blog.radware.com/applicationdelivery/wpo/2014/04/web-page-speed-affect-conversions-infographic/



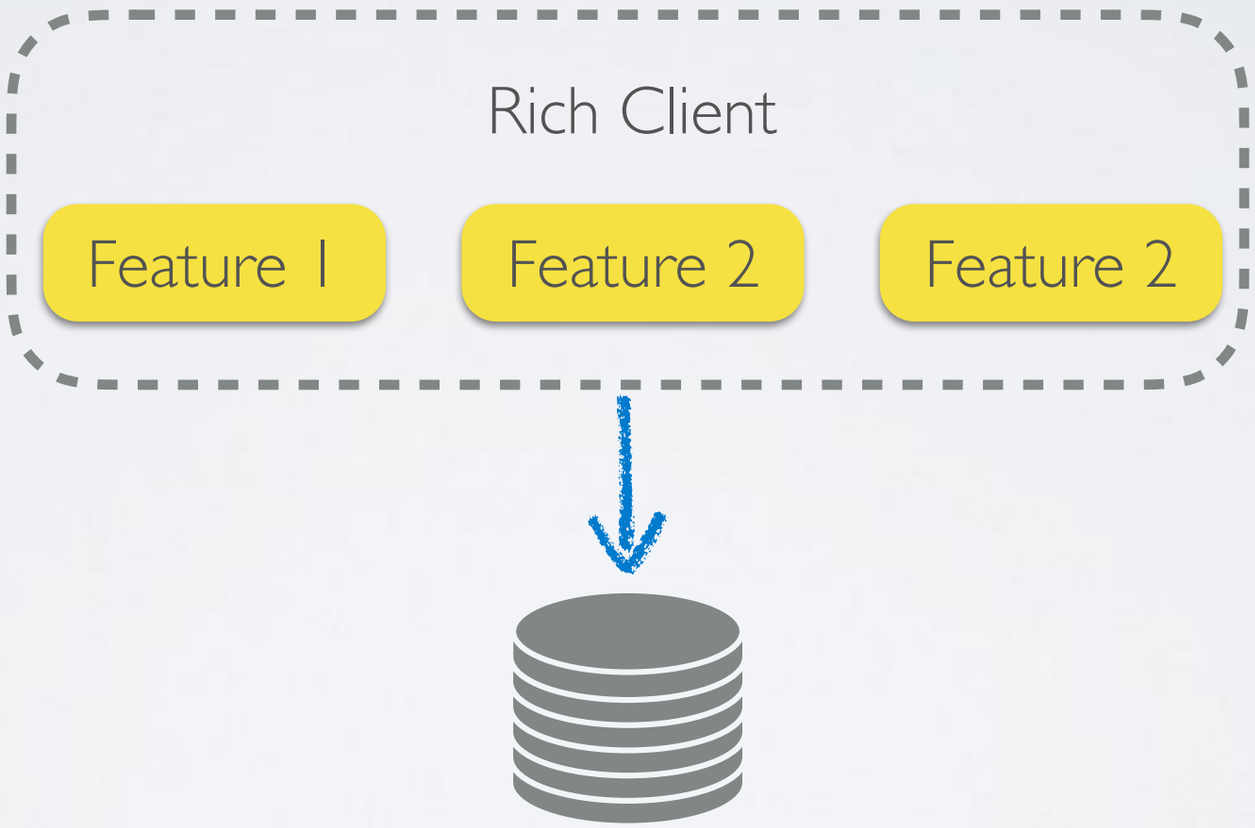
https://www.thinkwithgoogle.com/feature/mobile/

Enterprise Applications

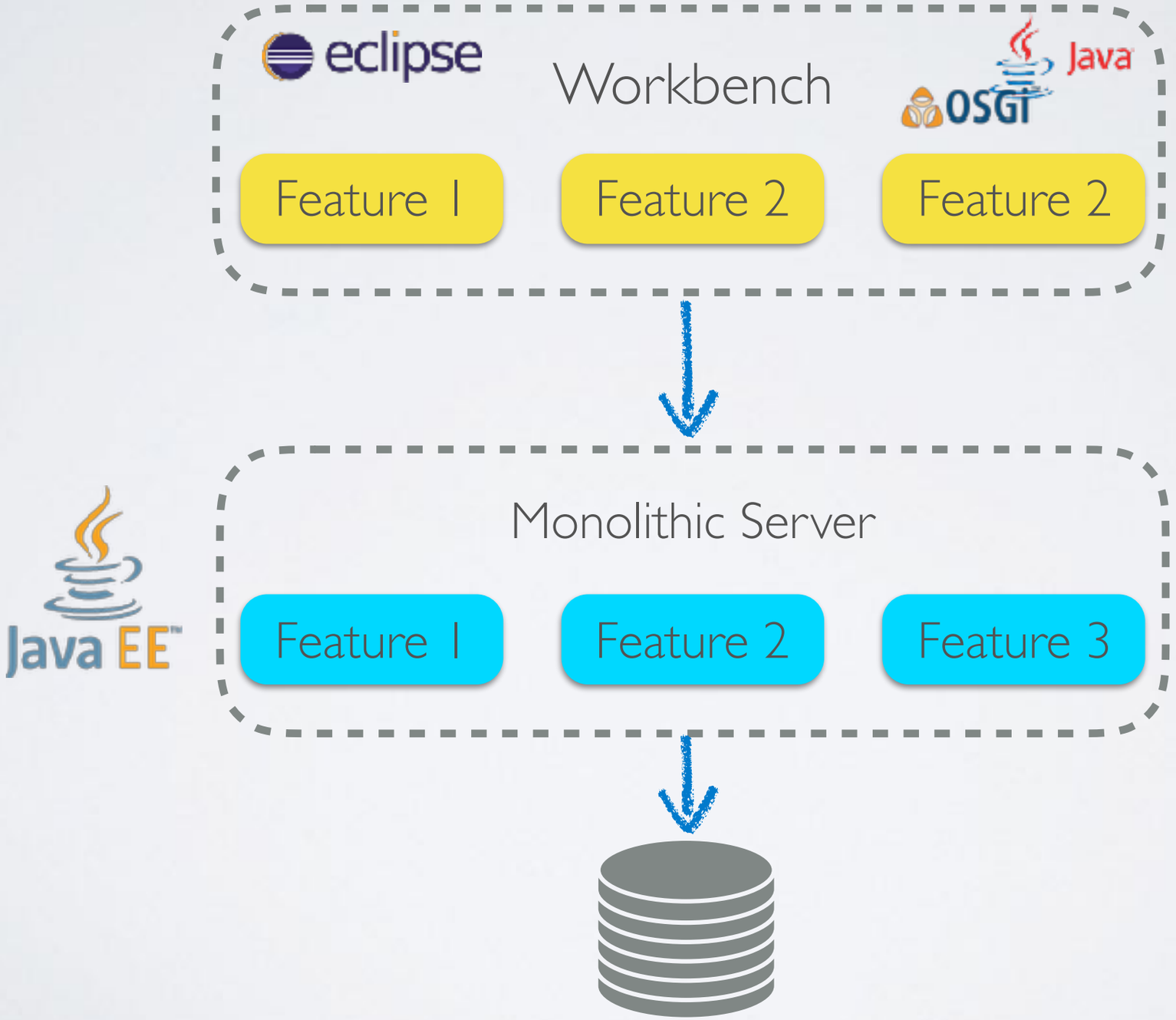
History of Enterprise Apps

- Rich Clients accessing the DB
- Server-Side Web with multiple Domains
- Moving to SPA -> enables Microservices
- Eclipse RCP "Workbenches" -> Moving to Web

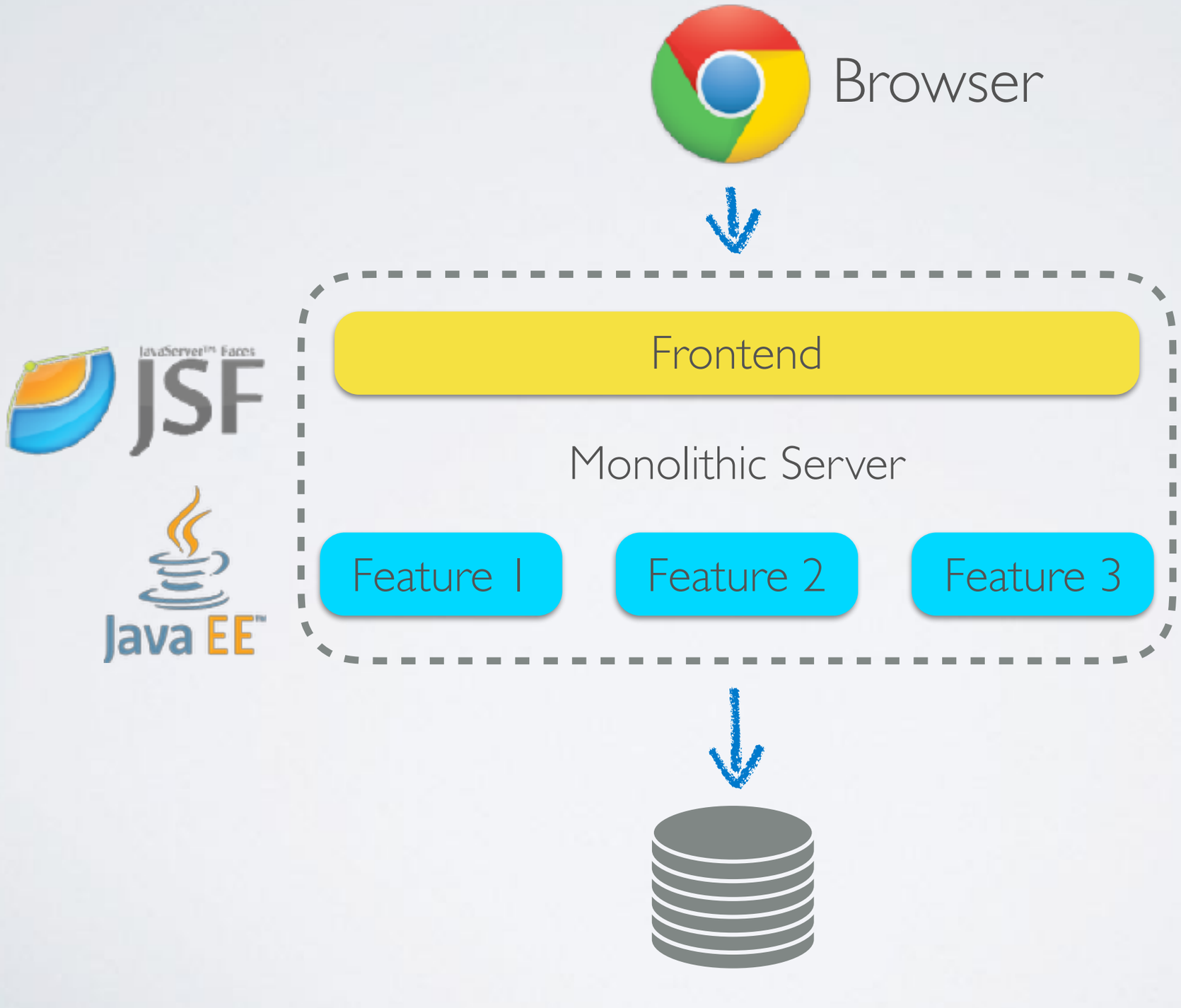
The Fat Client



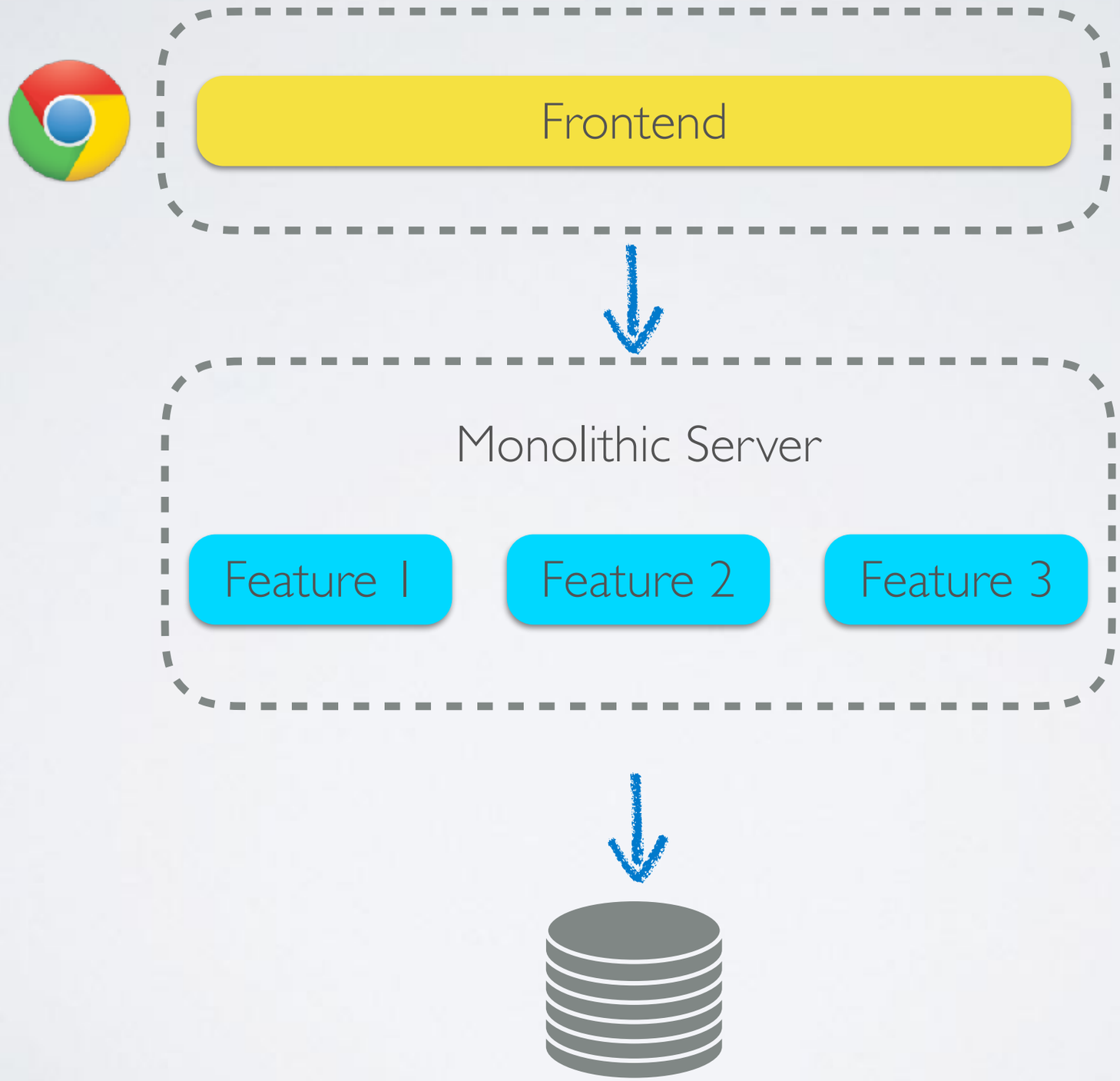
Workbench & Monolith



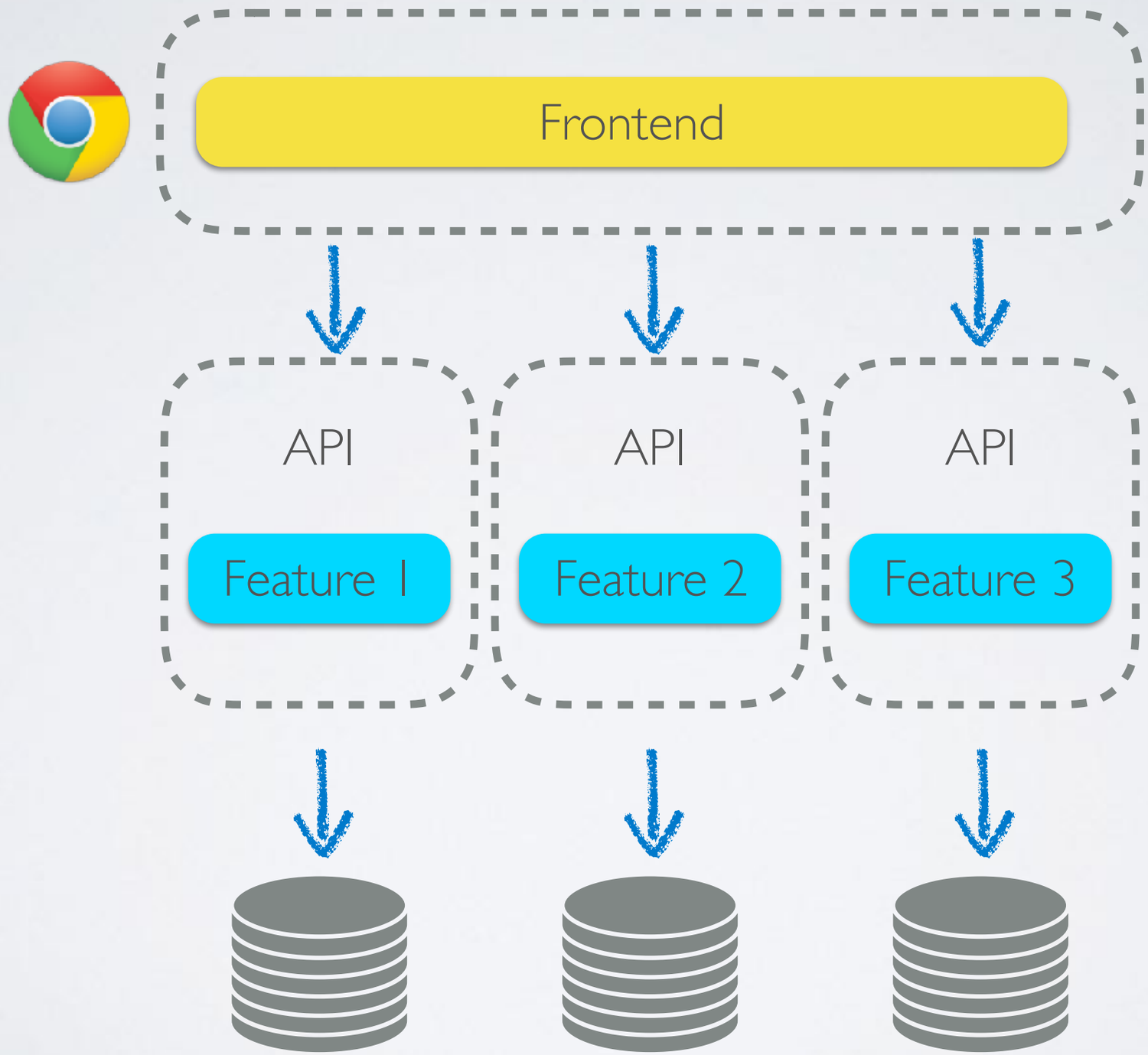
Traditional Web Monolith



SPA Monolith

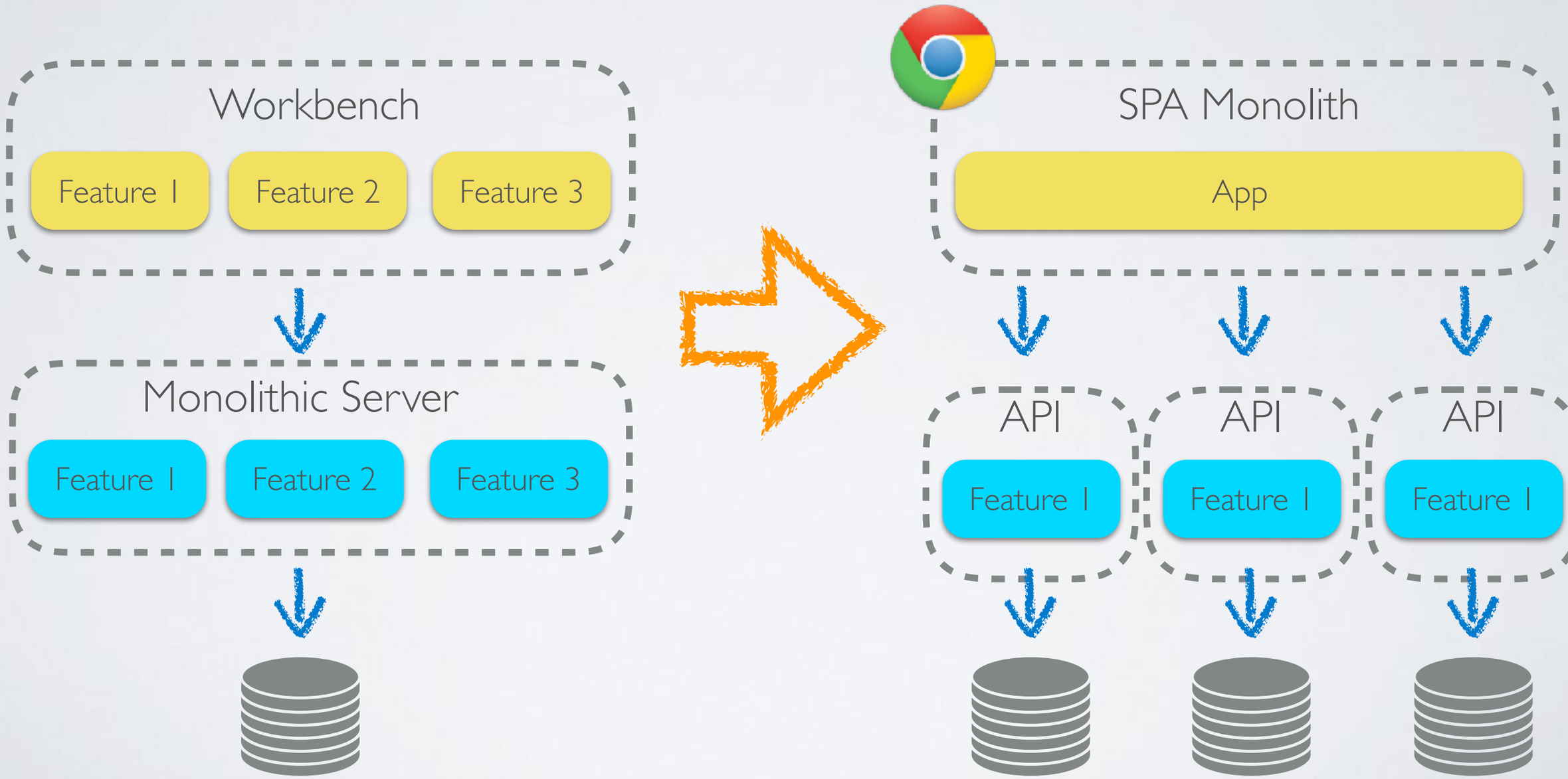


SPA Monolith & Microservices



Composition & Orchestration happens in the Client

Rich Client Migrations: The Rise of the Monoliths



Much Worse!

Workbench

eclipse

OSGi

Java

Feature 1

Feature 2

Feature 3

Monolithic Server

Feature 1

Feature 2

Feature 3

SPA Monolith

JS

App

API

Feature 1

API

Feature 1

API

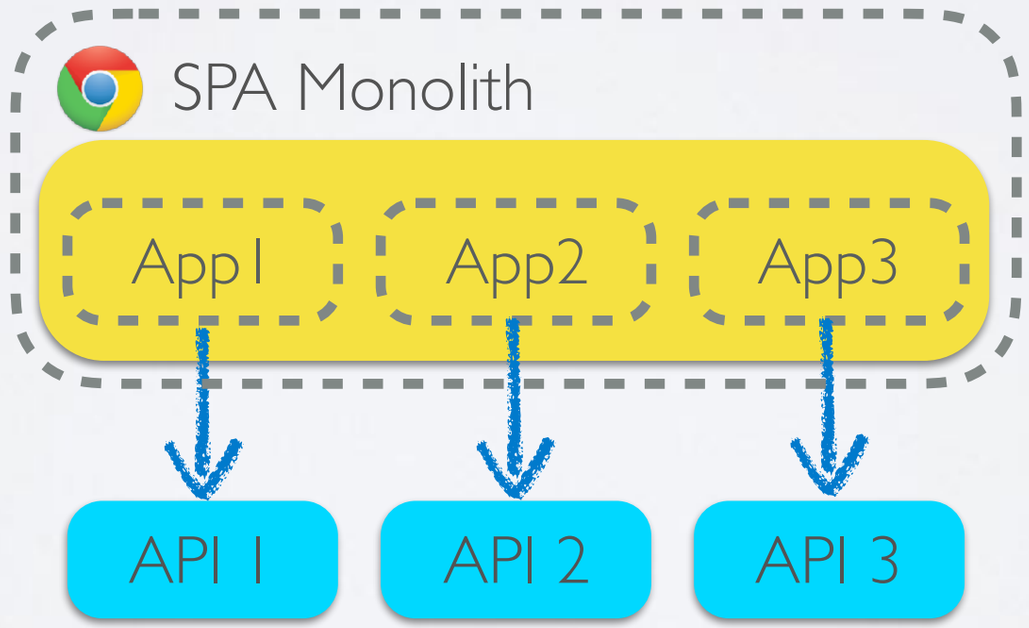
Feature 1

Improvement!

Beware of the Monolith

Unfortunately, we've seen many teams create front-end monoliths - a single, large and sprawling browser application - on top of their back-end services.

- ThoughtWorks Technology Radar



Problems of the Frontend Monolith

- Performance

  -> size is still relevant for web applications!

- Coupling & Dependencies
  - preventing change / agility / time to market
  - preventing effective/simple organization

    (effective end-to-end delivery, self organizing teams ...)

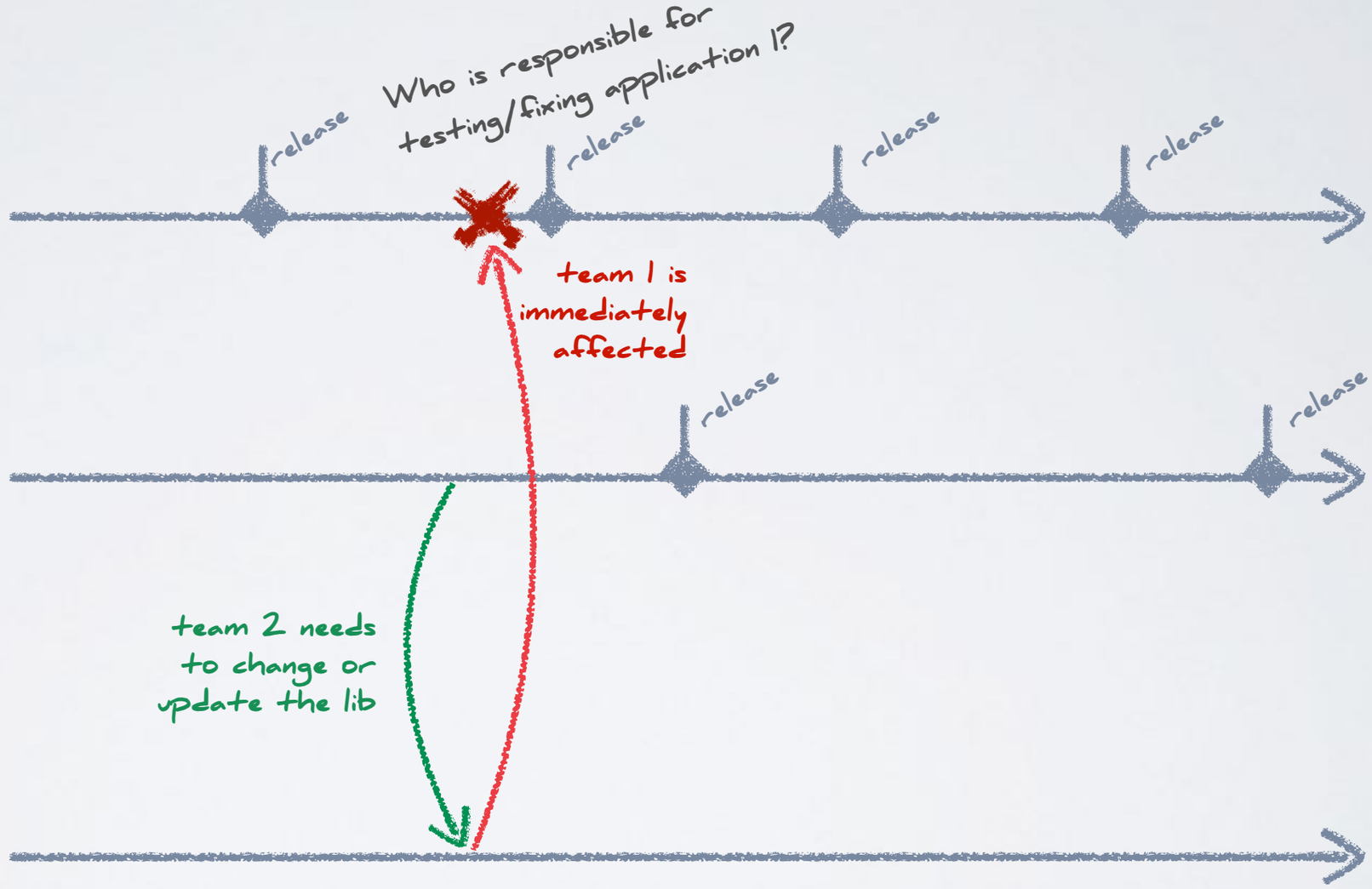
Coupling / Lock-Step Releases

"Think about the granularity of development & deployment artifacts"

Application 1
- implemented by team 1
- driven by org-unit 1

Application 2
- implemented by team 2
- driven by org-unit 2

shared lib
- inhouse lib
- 3rd party lib
  (ie. Angular, React)



The consequence is coupling between App1 and App 2:
- teams have to coordinate changes to the shared lib.
- development & release schedules of applications are coupled

Problems of the Frontend Monolith

- The web is traditionally not made for complex applications
  - global scope / no isolation / no modularization concepts
  - extremely dynamic (typing, monkey paching ...)
  - browser differences
- Frequent technology changes
  - core technologies are very backwards compatible
  - ecosystem moves at break-neck speed

The Bad Parts

I Am Devloper
@iamdevloper

Following

Being a JavaScript developer in 2014 is like speed dating.

45 seconds with each framework before you have to re-write.

RETWEETS 1,104 FAVORITES 637

8:25 AM - 16 Dec 2014

http://www.commitstrip.com/en/2018/01/08/new-year-new-frameworks/


The first rule of huge enterprise applications is:
You do not build huge enterprise applications.


The second rule of huge
enterprise applications is:
You do not build huge
enterprise applications.

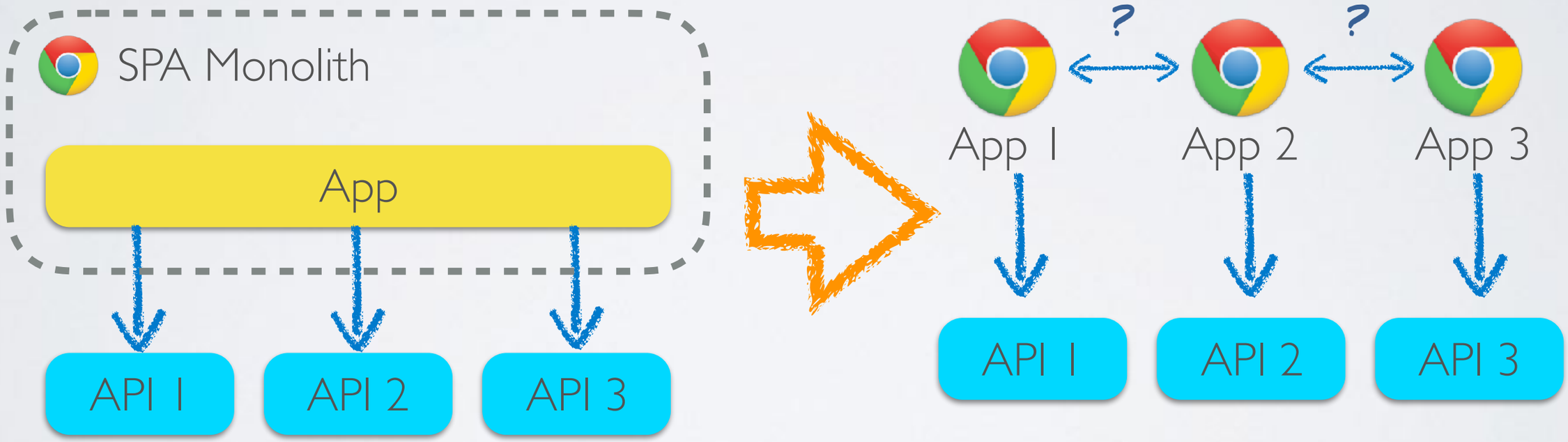

How?

Micro Frontends:

An architectural style where independently deliverable frontend applications are composed into a greater whole.

Micro Frontends

The million dollar question:
Modularizing frontend applications.



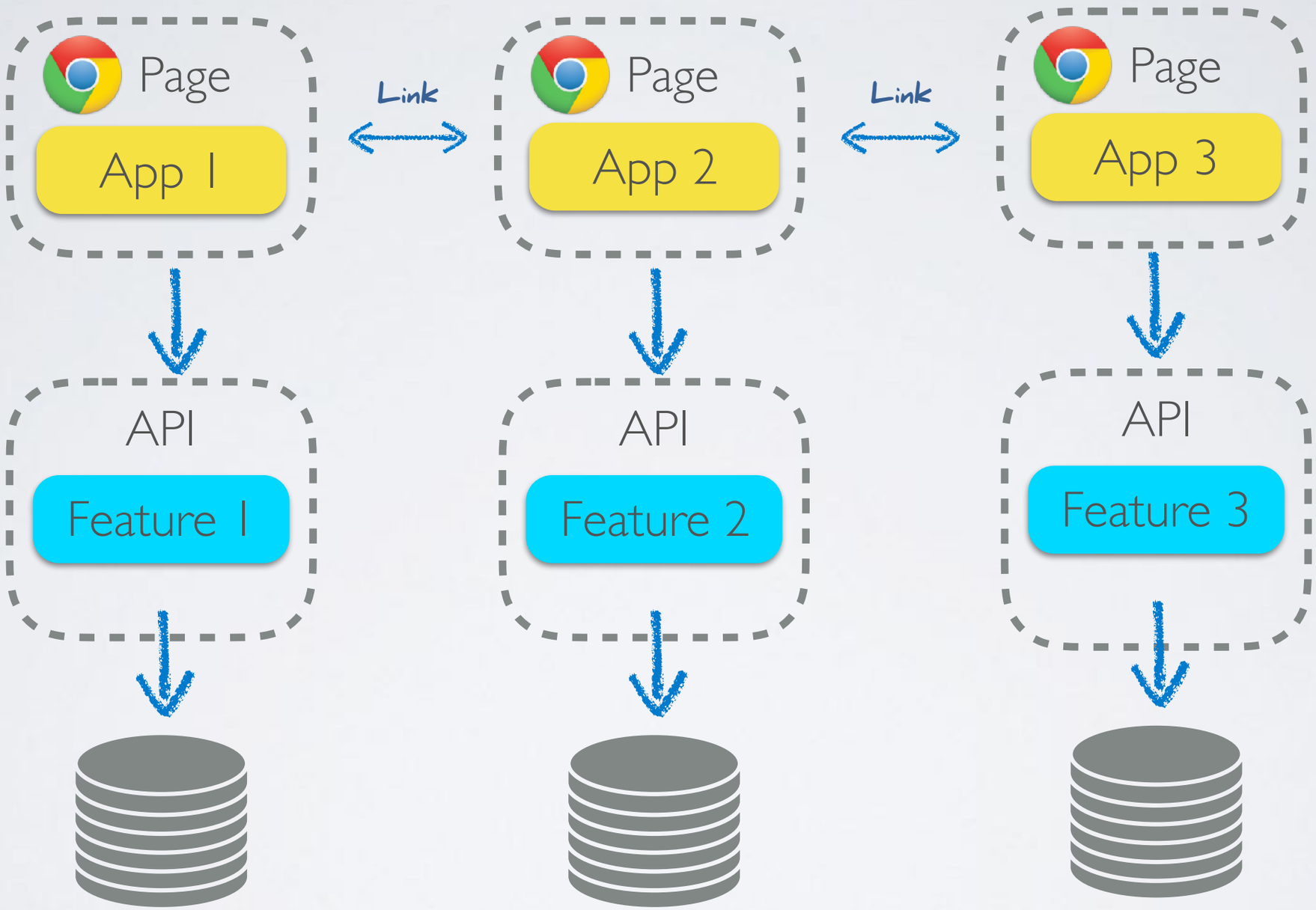
https://micro-frontends.org/

The Link

The magical integration pattern that the web is built upon:

```
<a href="..." >
```

Microfrontend Flavors: Mini SPAs



You have complete isolation. But jumping between apps is a full page load!

DEMO

Mini SPAs

Challenges

- Performance: Starting the Mini-SPAs has to be very quick!

- Sharing layout & styles for common UX

  -> use reusable technologies for styling -> plain CSS, SASS, less ...

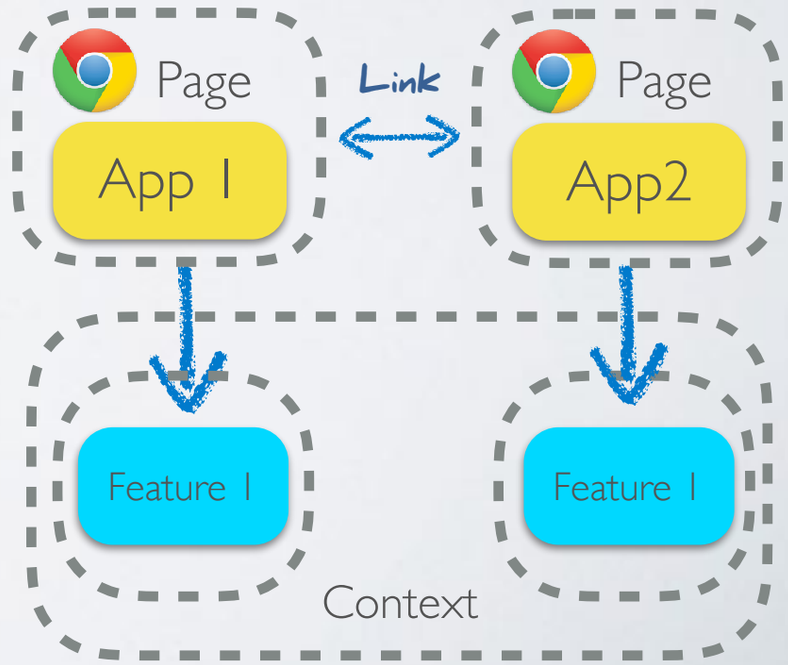
  -> be careful with "framework-specific" styling

- Sharing state / context

  -> query parameters, cookies, http-headers

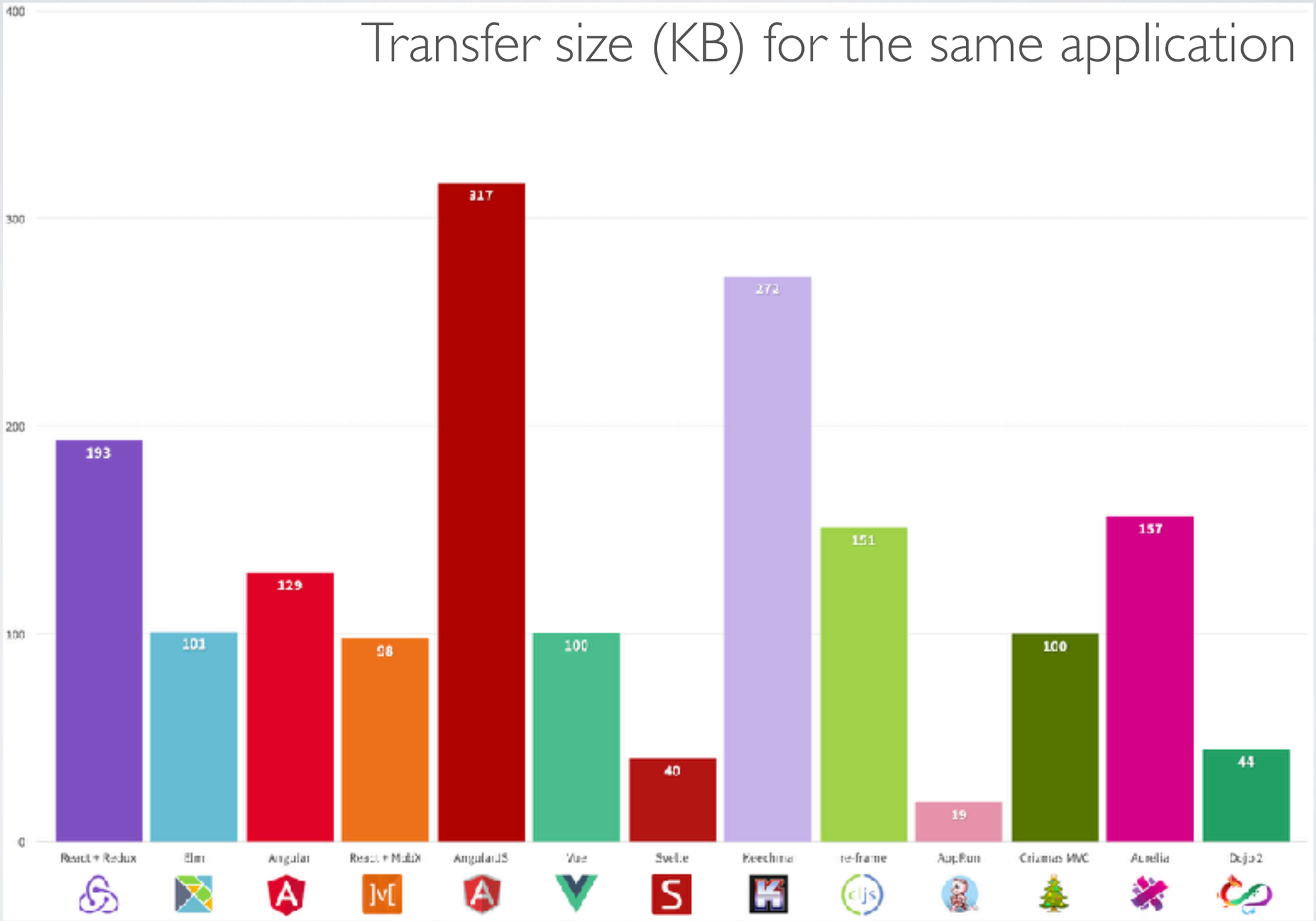
  -> server side session / context

  -> Local Storage / Indexed DB



Remember: Size does Matter

Frameworks could matter ...

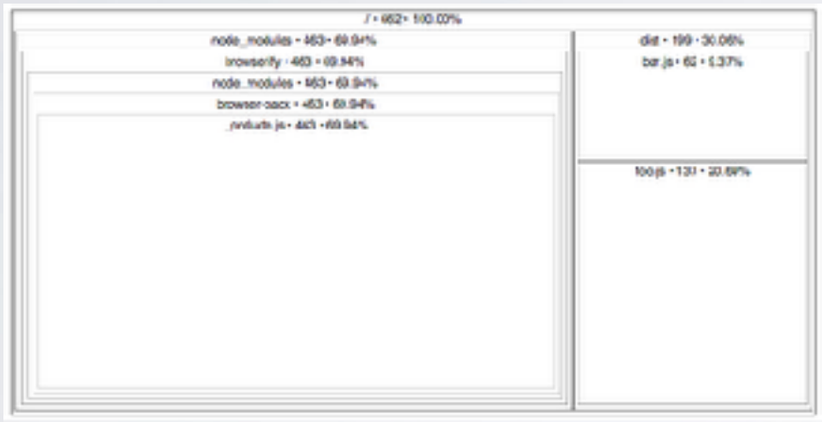


https://medium.freecodecamp.org/a-real-world-comparison-of-front-end-frameworks-with-benchmarks-2018-update-e5760fb4a962

Keep an eye on your bundles!

www.bundlephobia.com

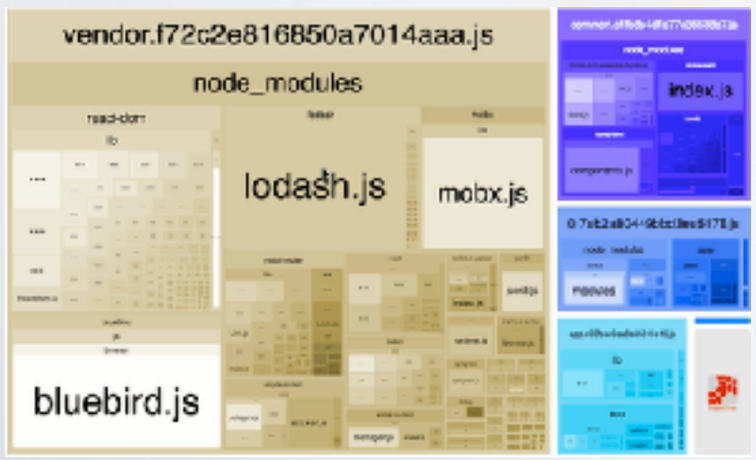
source-map-explorer:
https://www.npmjs.com/package/source-map-explorer




```
npm install -g source-map-explorer
```

```
ng build --prod --source-map
source-map-explorer dist/ng-app/main.XYZ.js
```

webpack-bundle-analyzer:
https://github.com/webpack-contrib/webpack-bundle-analyzer

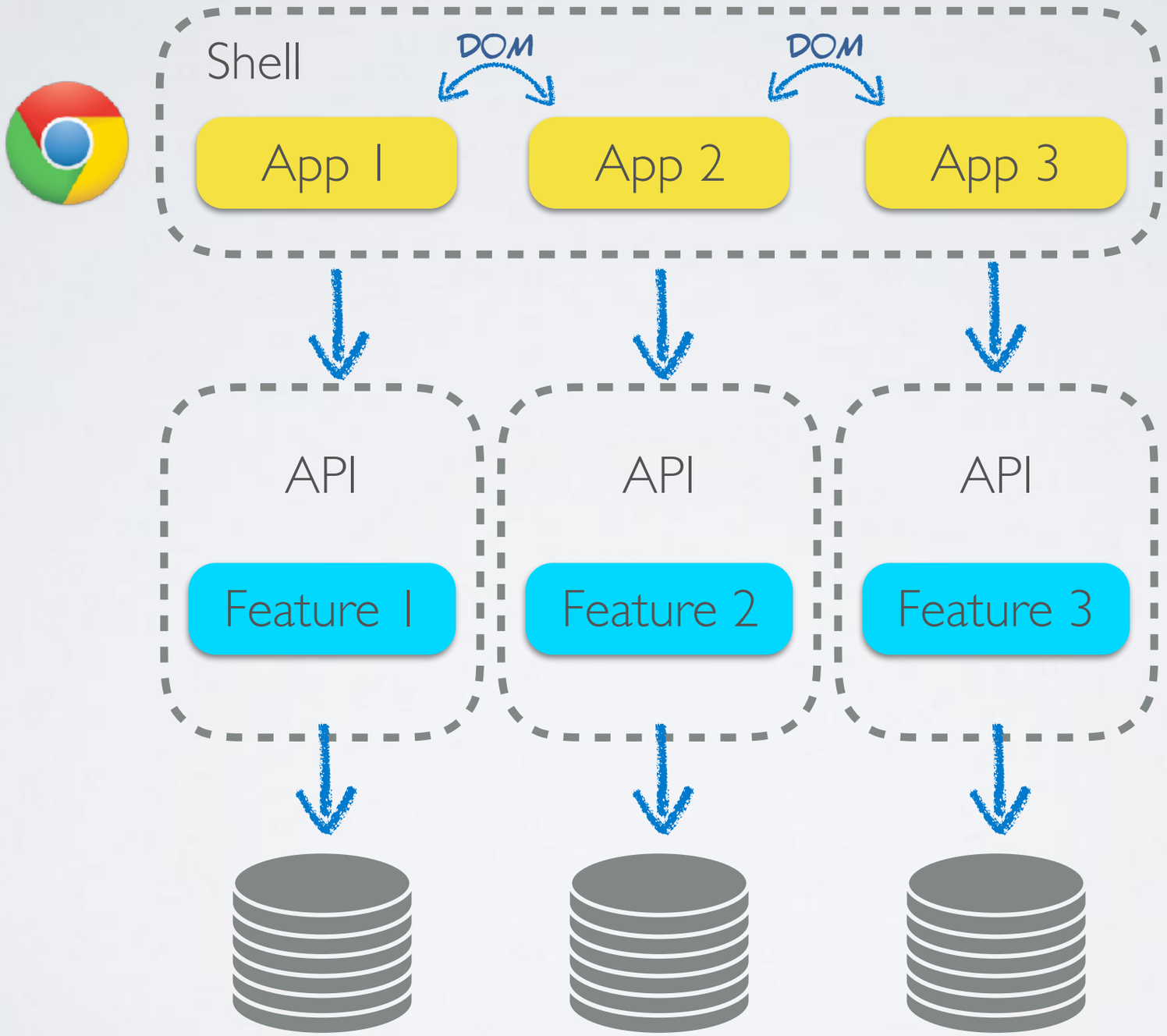



```
npm install -g webpack-bundle-analyzer
```

```
ng build --prod --stats-json
webpack-bundle-analyzer dist/ng-app/stats.json
```


But I need a single SPA!

Microfrontend Flavors: SinIge SPA



All applications live on the same DOM.

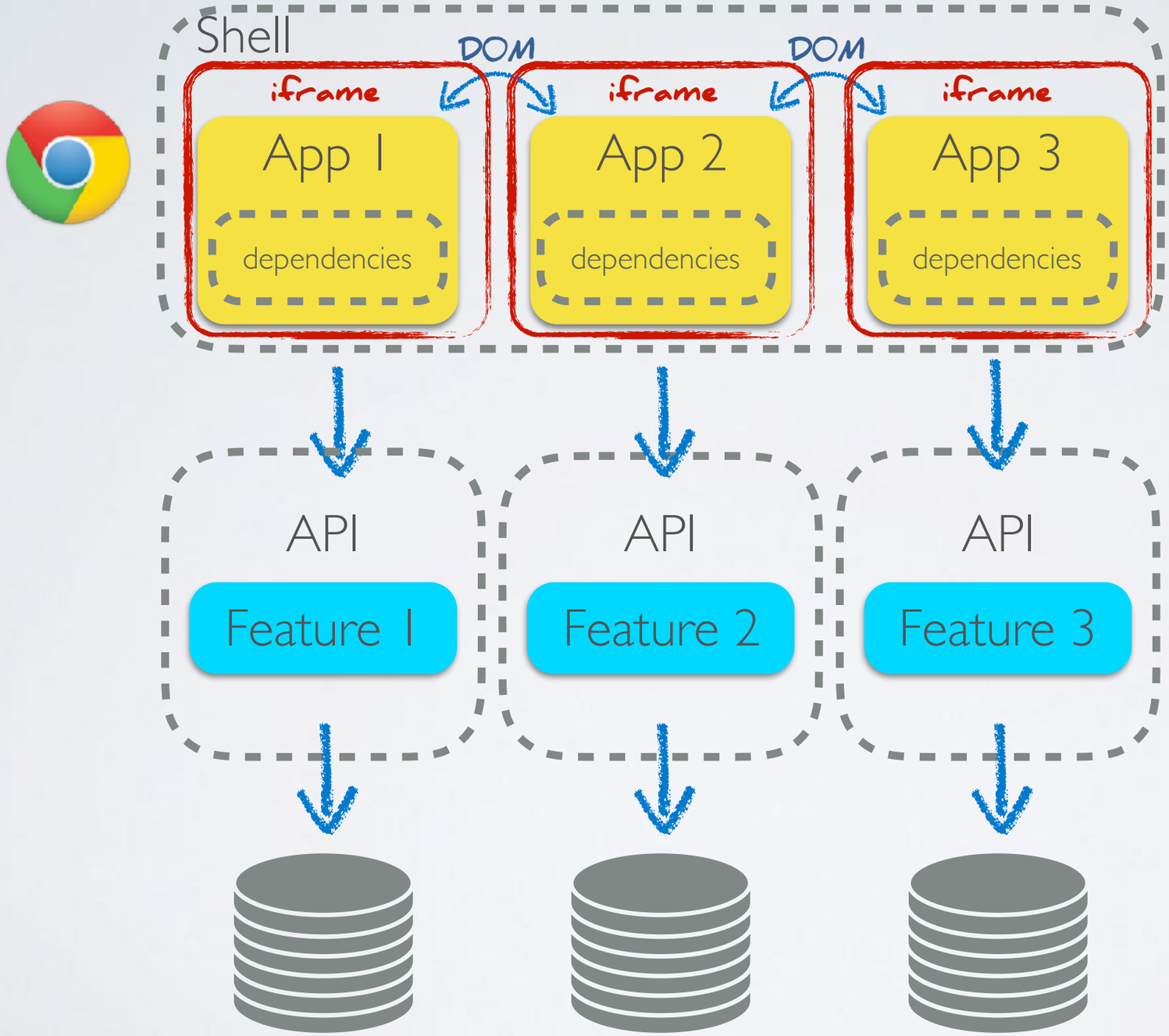
App-communication happens over the DOM.

https://github.com/CanopyTax/single-spa

You will need a shell

- Top-level routing
- Lazy loading of apps (child SPAs)
- Inter-SPA communication
- Sharing a context

Sinlge SPA: iframes



Analogy: complete classloader isolation.

Browser can leverage caching, but for each app all dependencies have to be parsed, compiled and executed.

Option: "Child SPAs" in iframe

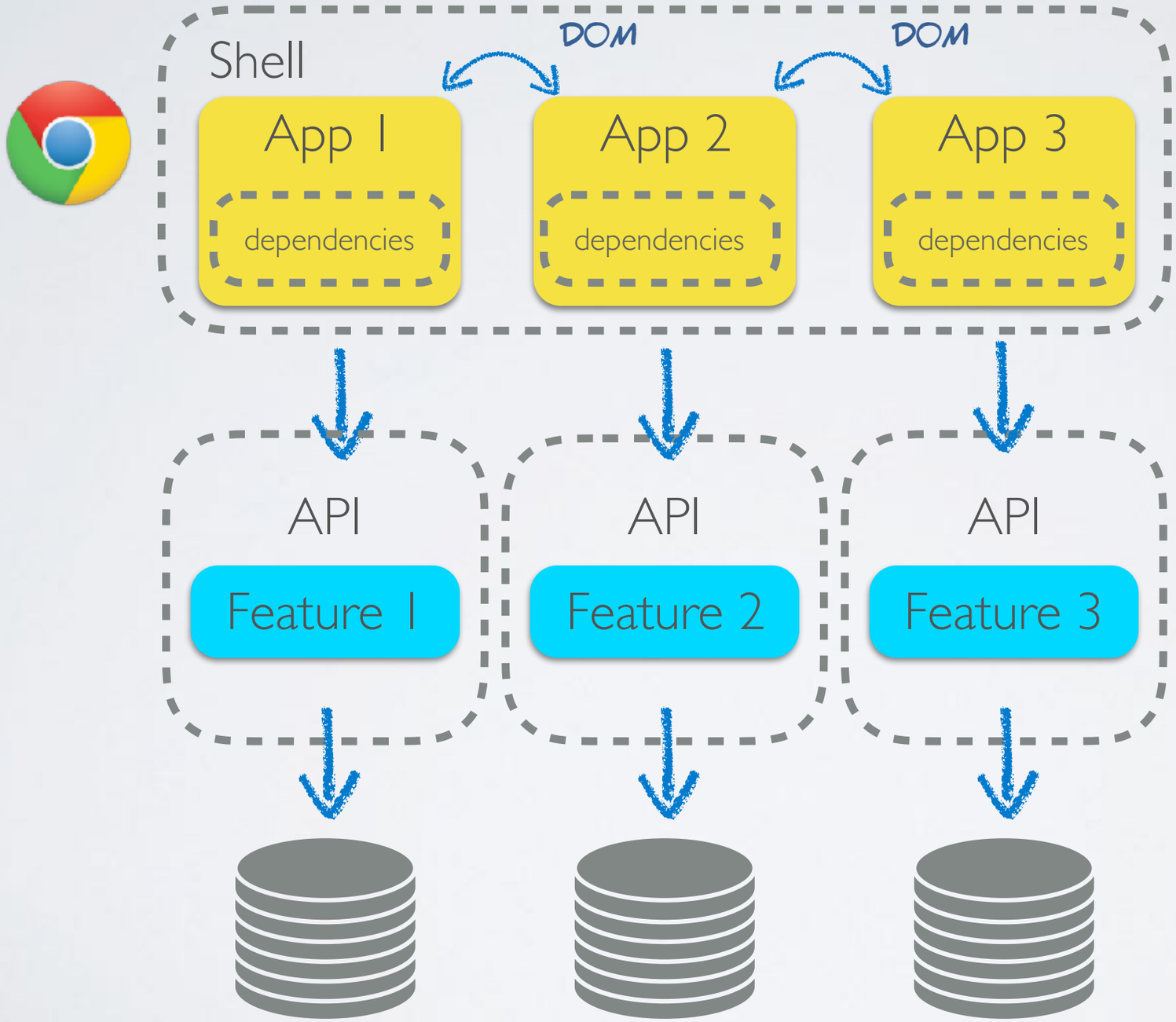
Advantages:

- true isolation
- integration of "non-SPA" applications / sites

Disadvantage:

- No isolation: same global DOM, same scripting context, same global scope
- Responsive layout is difficult
- Router can't access the URL
- Some 3rd party libraries break, since they depend on document events

Sinlge SPA: Isolated



Like "complete classloader isolation", but only if the app & dependencies don't pollute the global scope.

Note: Angular comes with Zone.js, which heavily pollutes the global scope!

Browser can leverage caching, but for each app all dependencies have to be parsed, compiled and executed.

DEMO: Single-Spa

https://single-spa.js.org/

Option: Shared DOM

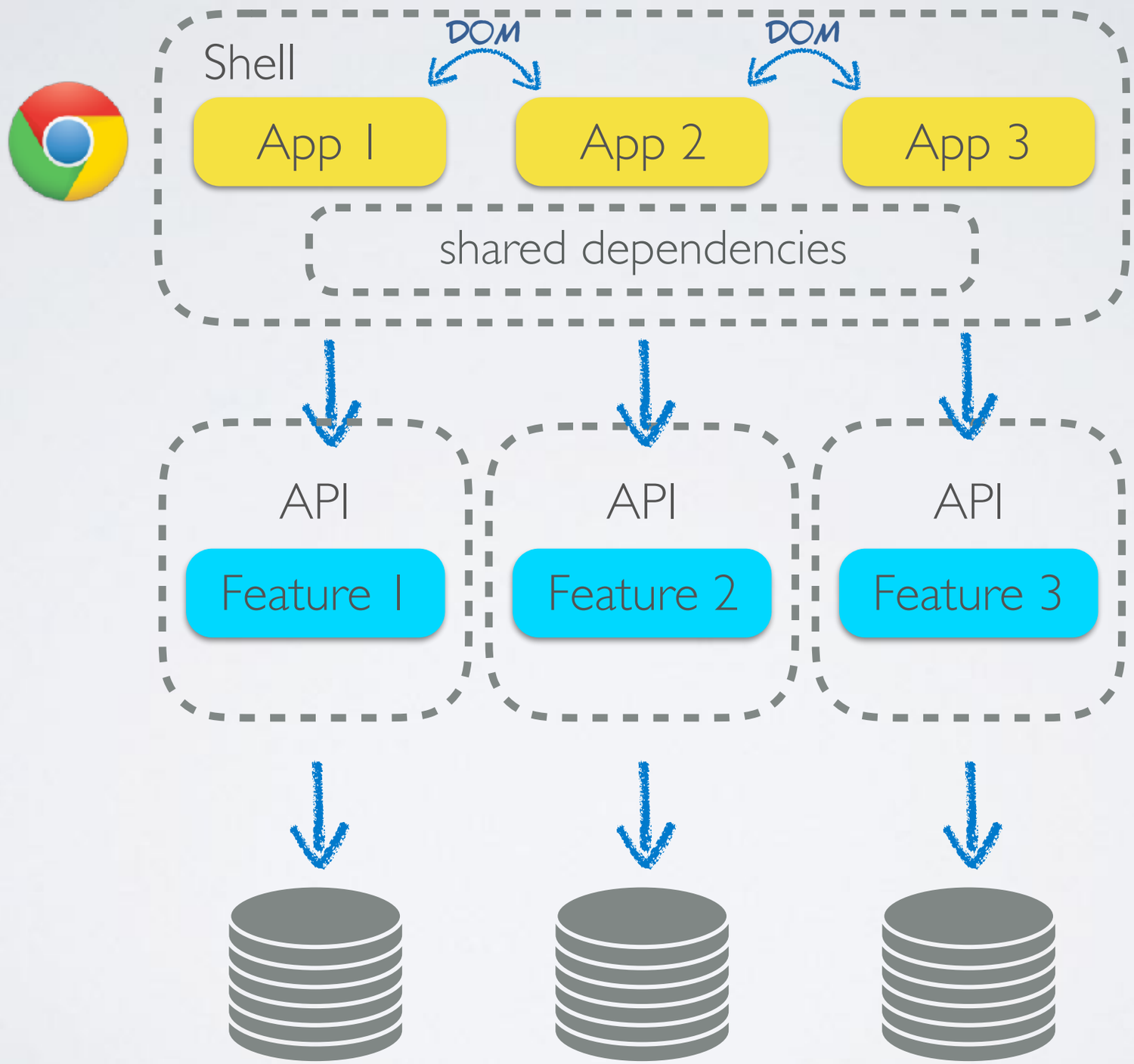
```
<html>
▶<head>…</head>
▼<body>
  ▶<div id="navbar">…</div>
   <div id="home"></div>
   <div id="angularjs"></div>
   <div id="react-app"></div>
… ▼<div id="angular"> == $0
    ▼<example-ng2-app ng-version="5.1.1">
      ▶<div>…</div>
     </example-ng2-app>
    </div>
   <div id="vue-app"></div>
   <div id="svelte-app"></div>
   <div id="cycle-app"></div>
   <div id="preact-app"></div>
   <div id="vanillajs"></div>
   <div id="inferno-app"></div>
   <div id="ember-app"></div>
   <div class="hiddendiv common"></div>
  ▶<div id="sideNav">…</div>
   <div class="drag-target" data-sidenav="mobile-de
 </body>
</html>
```

empty

```
<html>
▶<head>…</head>
▼<body>
  ▶<div id="navbar">…</div>
   <div id="home"></div>
   <div id="angularjs"></div>
… ▼<div id="react-app"> == $0
    ▼<div>
      ▼<div>
        ▶<div style="padding: 10px 20px; overflow: h
        ▶<div class="container">…</div>
       </div>
     </div>
    </div>
   <div id="angular"></div>
   <div id="vue-app"></div>
   <div id="svelte-app"></div>
   <div id="cycle-app"></div>
   <div id="preact-app"></div>
   <div id="vanillajs"></div>
   <div id="inferno-app"></div>
   <div id="ember-app"></div>
   <div class="hiddendiv common"></div>
  ▶<div id="sideNav">…</div>
   <div class="drag-target" data-sidenav="mobile-de
 </body>
</html>
```

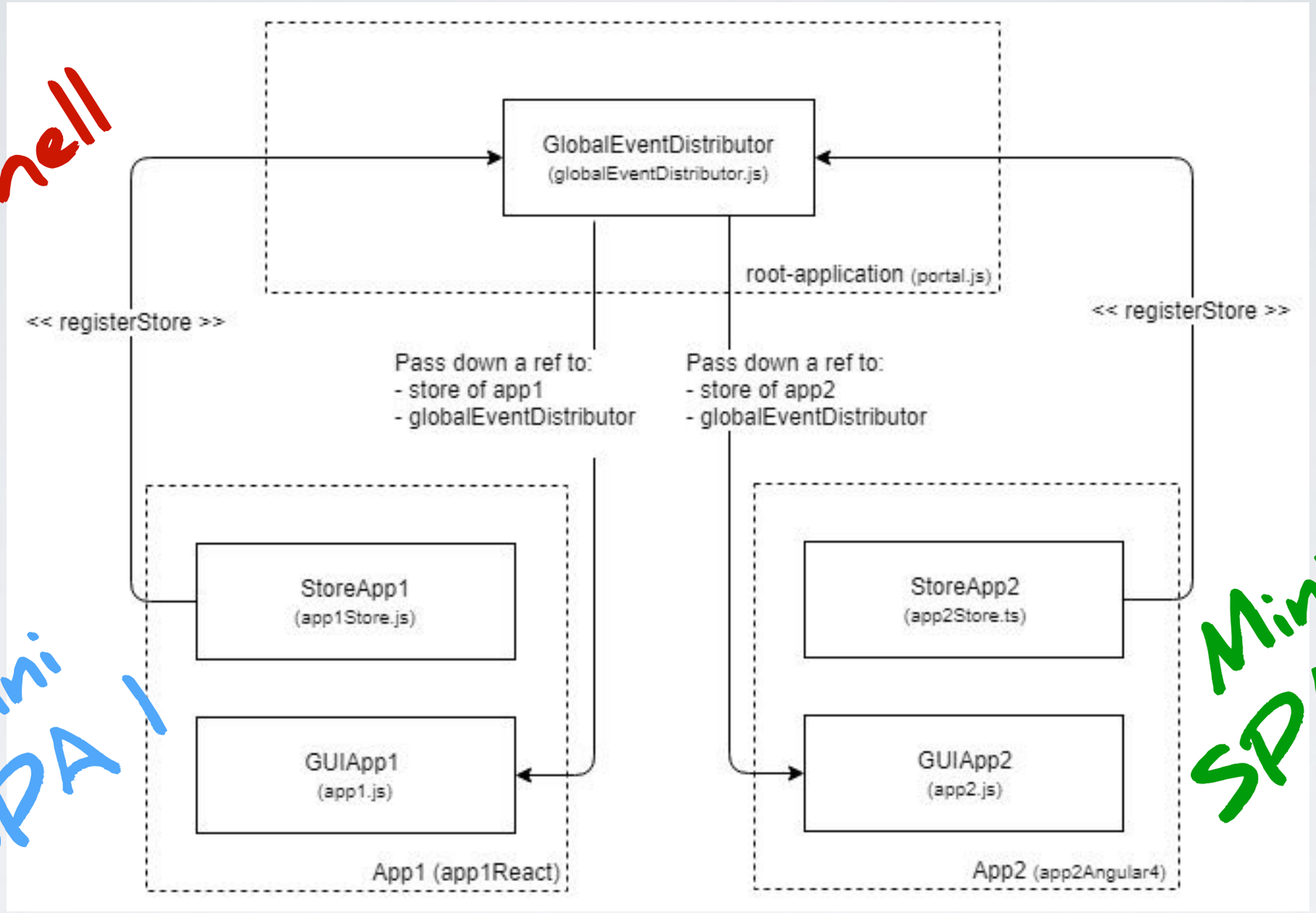
empty

Sinlge SPA: Shared Dependencies



Analogy: shared classloader

Inter-SPA Communication

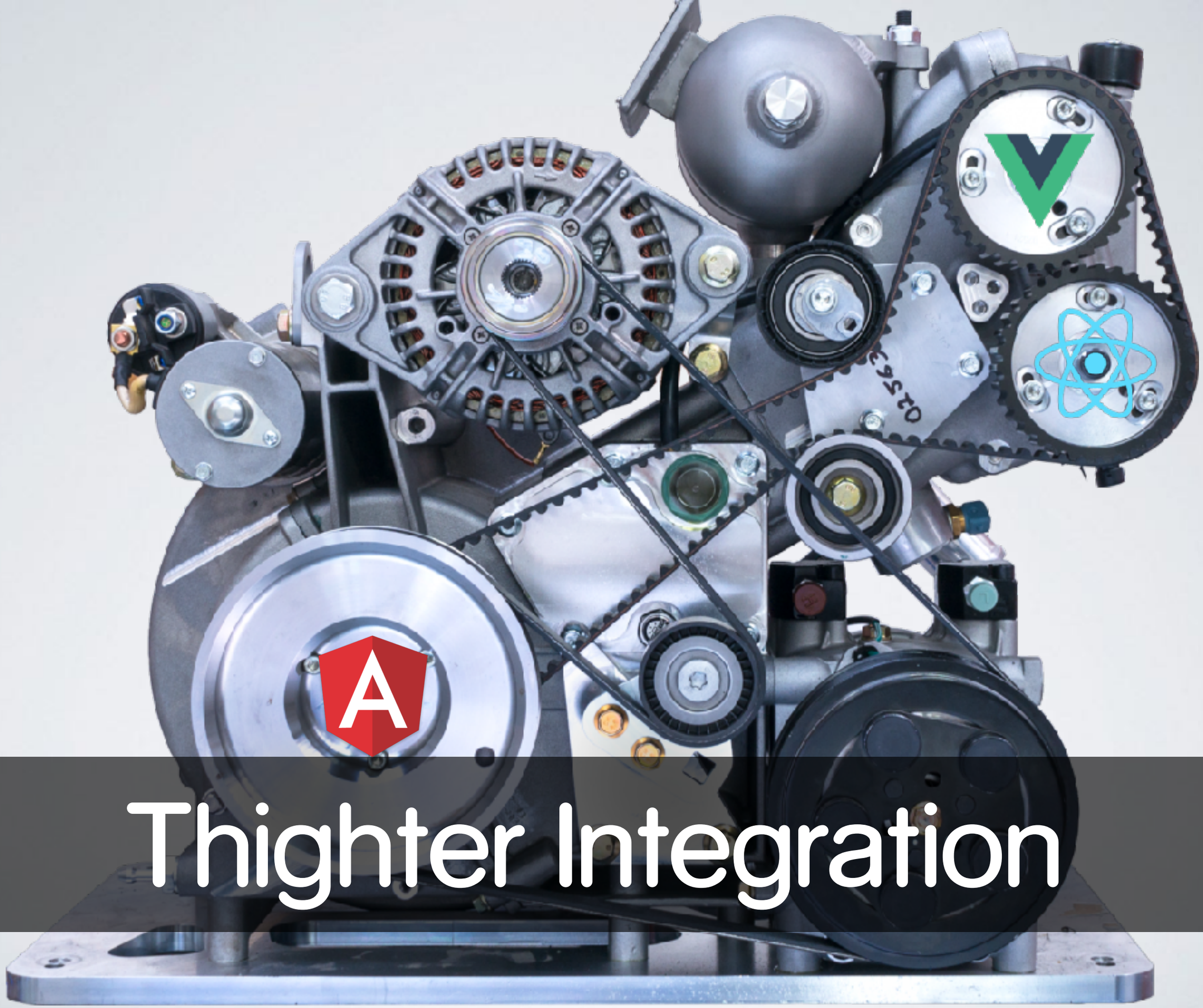



```
globalIncrement() {
   this.globals.globalEventDistributor.dispatch(this.actions.increment());
}
```

https://github.com/jbandi/single-spa-portal-example

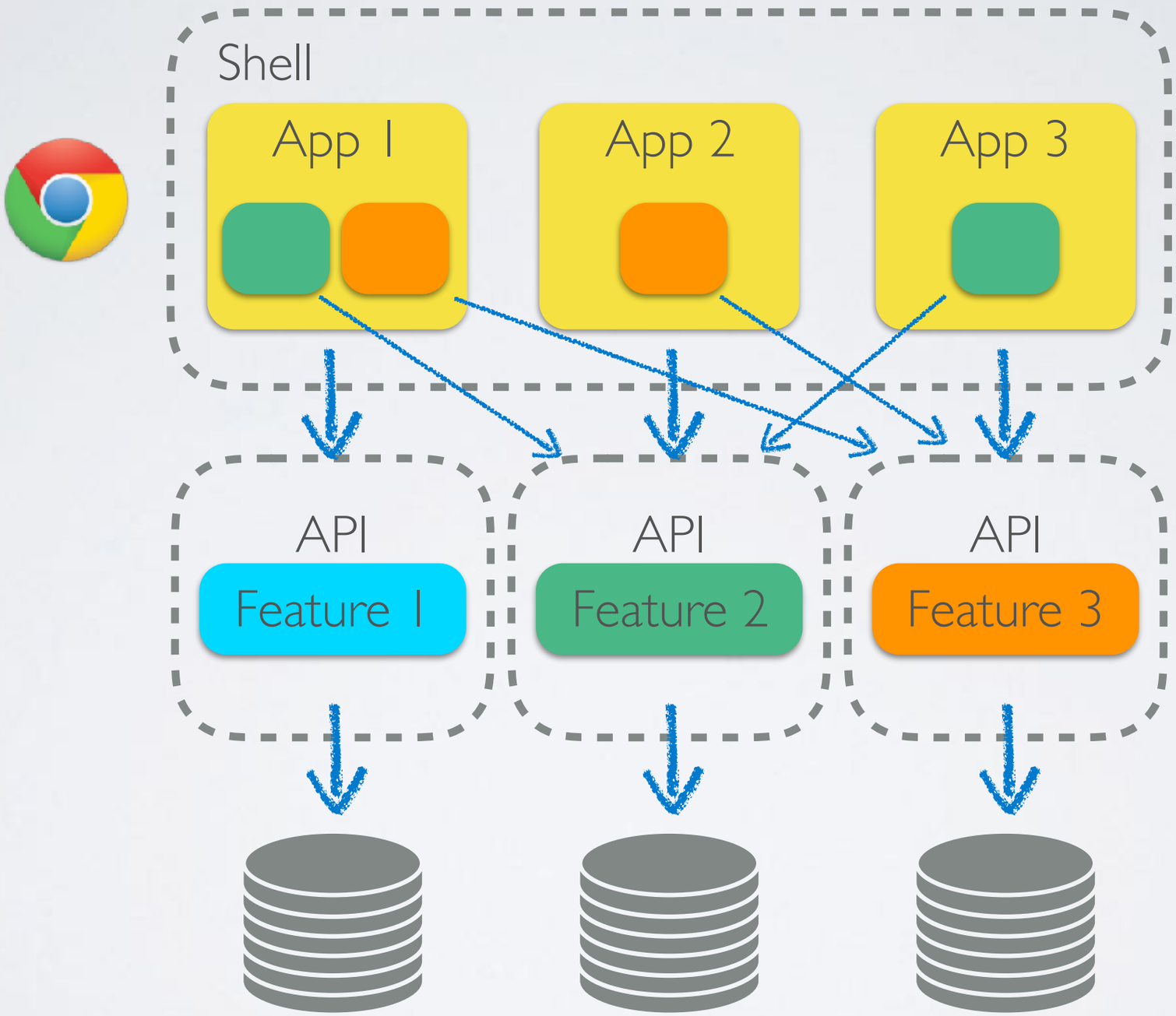
DEMO: Single-Spa

https://single-spa.js.org/



Thighter Integration

Frontend Composition




How?

Web Components

WebComponents is a series of browser standards for creating reusable custom elements:

Shadow DOM	Encapsulation
HTML templates	markup that is not initially rendered and can be instantiated.
Custom elements	JavaScript API to define custom elements that can be used in html and their behavior

Custom Elements API:

JavaScript

```
customElements.define('element-details',  class extends HTMLElement { ... })
```

html:

```
<div> <element-details></element-details> <div>
```

Browser support:
native: Chrome, Safari
polyfill: Firefox, Edge, IE11

https://developer.mozilla.org/en-US/docs/Web/Web_Components
https://developer.mozilla.org/en-US/docs/Web/API/Window/customElements

Google

jb

Secure https://www.google.com

Gmail Images

Google

are web component

are web components **dead**
are web components **ready**
are web components **the future**

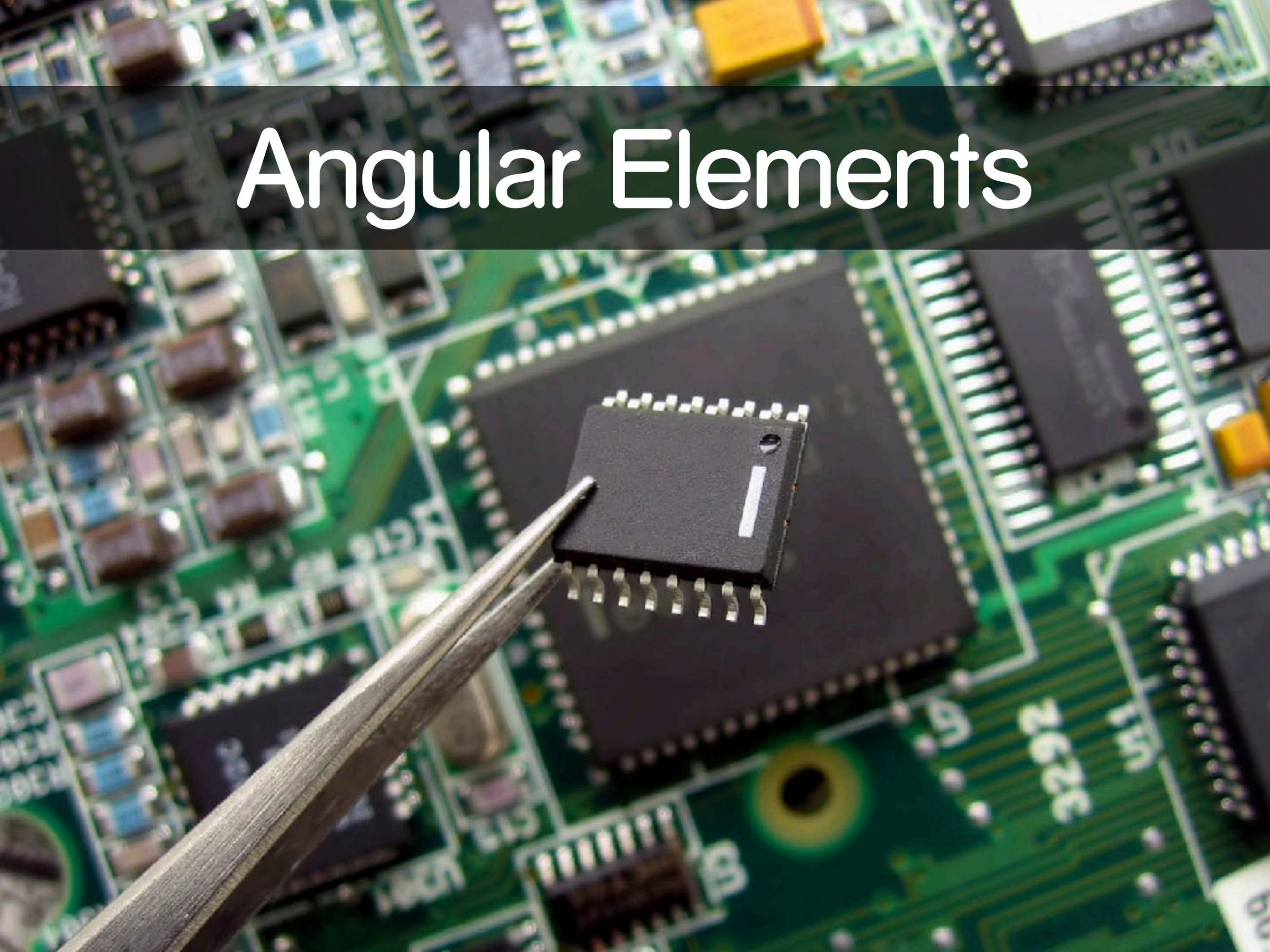
Learn more

Google offered in: Deutsch Français Italiano Rumantsch

Switzerland

Advertising Business About Privacy Terms Settings

Angular Elements



Angular Elements

Using Angular components as Web Components.

Available in Angular 6

Angular Elements can transform Angular components into classes that can be registers as custom elemements:

```
import { createCustomElement } from '@angular/elements';
import { PopupComponent } from './popup.component';

const PopupElement = createCustomElement(PopupComponent, {injector});


customElements.define('element-details', PopupElement)
```

in any html (also from React JSX and Vue templates):

```
<div> <element-details></element-details> <div>
```

https://angular.io/guide/elements

Vue CLI Compilation

Vue CLI 3 can wrap a vue component / app
inside a Web Component

https://cli.vuejs.org/guide/build-targets.html#web-component
https://github.com/vuejs/vue-web-component-wrapper

Build:

```
vue-cli-service build --target wc --name my-element ./main.js
```

any html:

```
<script src="path/to/my-element.js"></script>
...
<my-element></my-element>
```

Wrapping React

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

class MyApp extends HTMLElement {
  connectedCallback() {
    const mountPoint = document.createElement('span');
    this.attachShadow({ mode: 'open' }).appendChild(mountPoint);

    const message = this.getAttribute('message');
    ReactDOM.render(<App message={message}/>, mountPoint);
  }
}
customElements.define('my-app', MyApp);
```

any html:

```
<my-app messsage="Hi there!"></my-app>
```

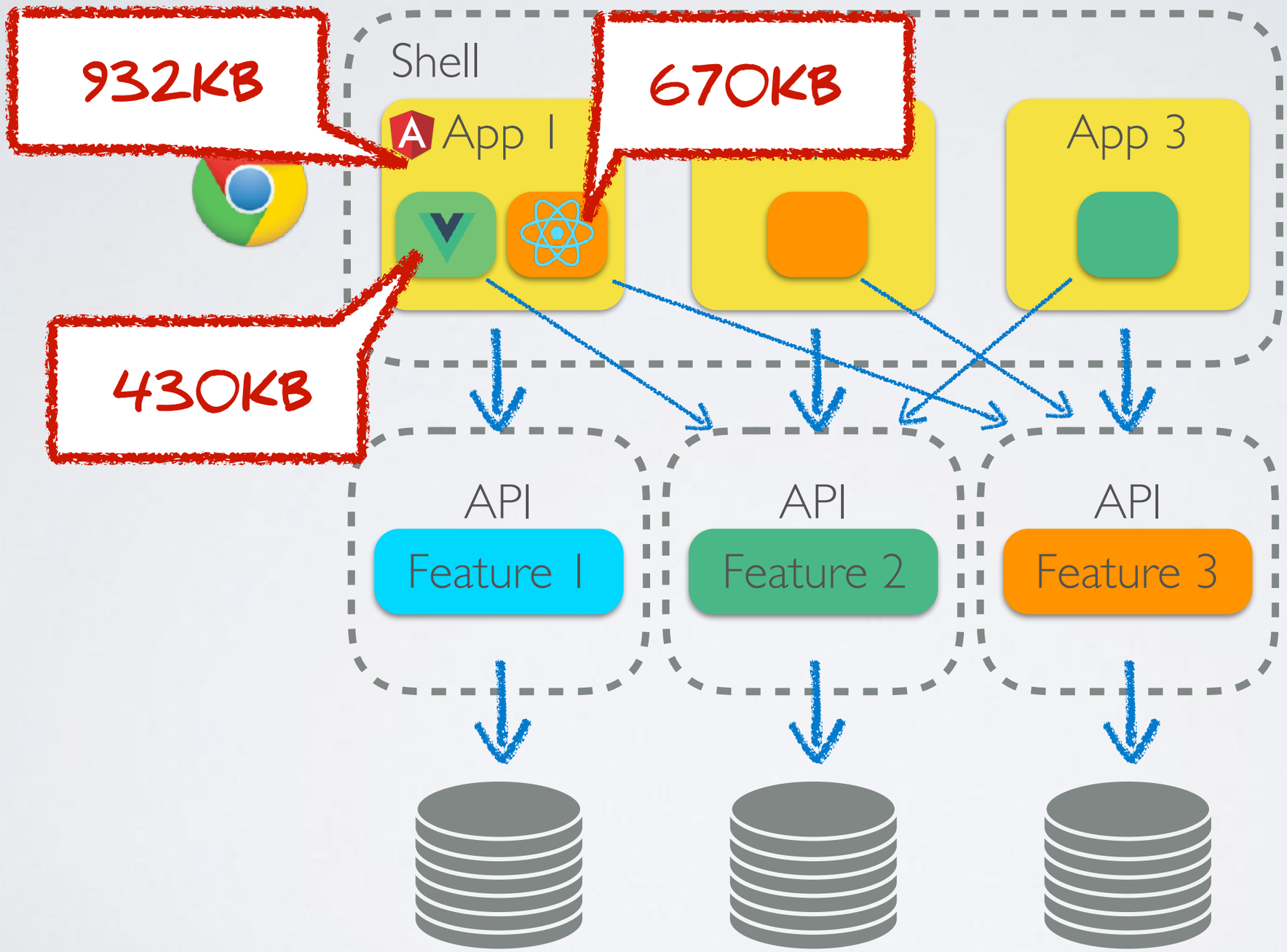
DEMO

https://github.com/manfredsteyer/Angular_MicroApps_Different_Technologies

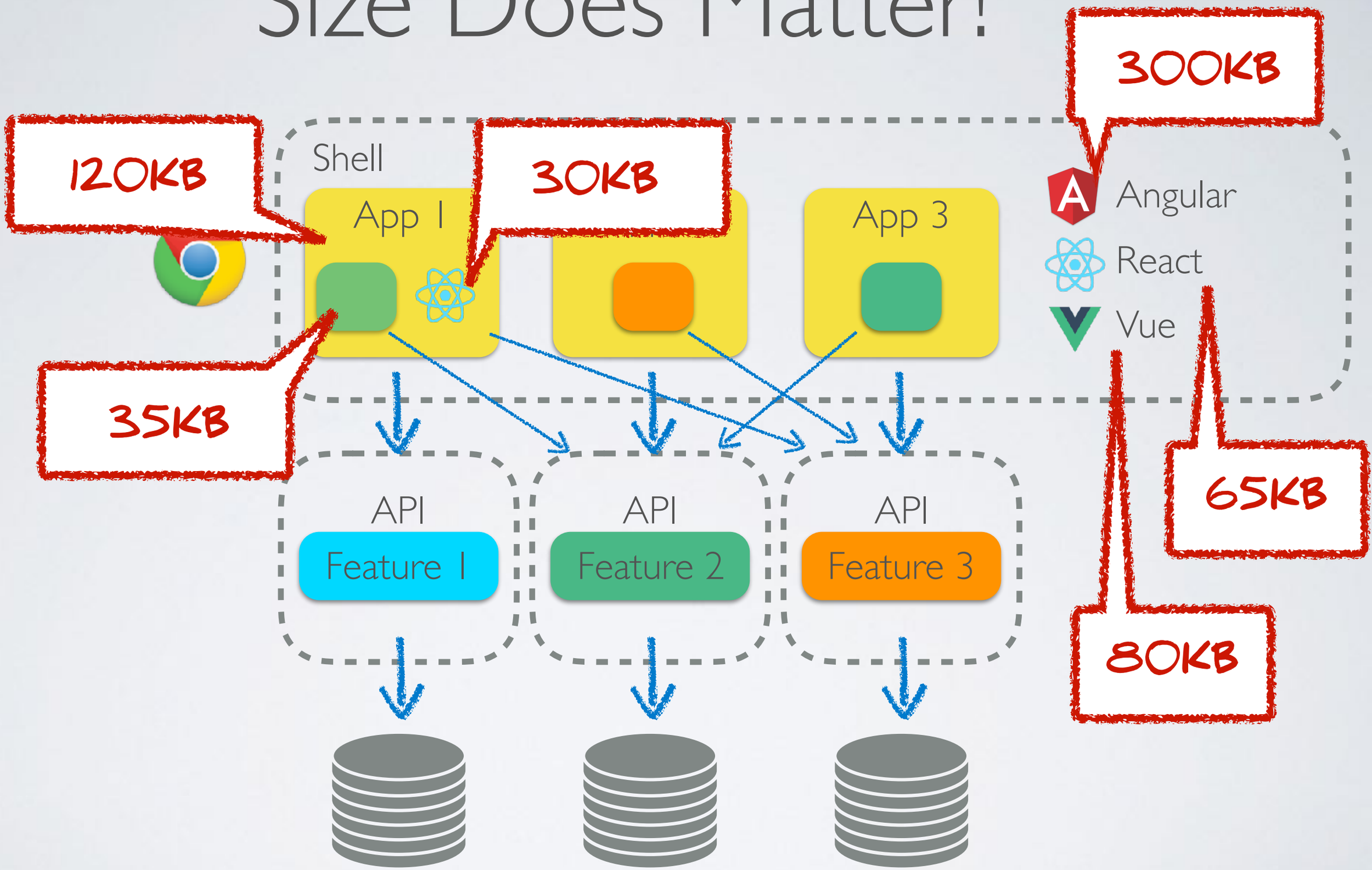
https://github.com/manfredsteyer/angular-elements-dashboard

More than ever: Size does Matter!

Size Does Matter!



Size Does Matter!



Big & Independent

vs.

Small & Coupled

More Challenges

Dataflow & sharing state between components:

Applications do not just consist of simple components.
There is shared state & context
(Angular services, React context, Redux stores ...)

WebComponents just have primitive string properties!

Versioning: There is no versioning of custom elements!

"It feels like imperative DOM programming" (remember jQuery?)

There is no Silver Bullet for Frontend Modularization

Fragen?

https://github.com/jbandi/sbb-dev-day-2018

JavaScript / Angular / React Schulungen
& Coachings,
Project-Setup & Proof-of-Concept:

http://ivorycode.com/#schulung
jonas.bandi@ivorycode.com