Progressive Web Apps across all Frameworks

Mike Hartington

Ionic, Angular GDE

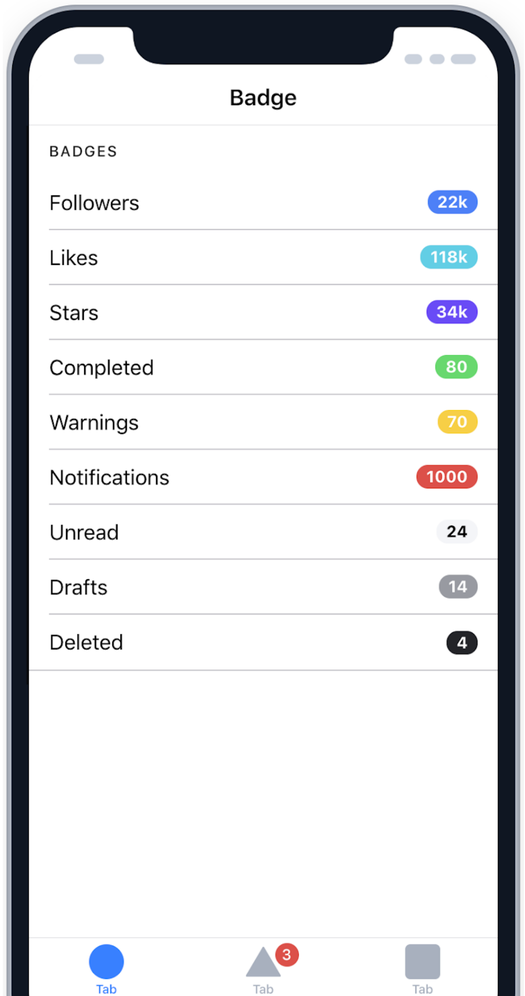
@mhartington

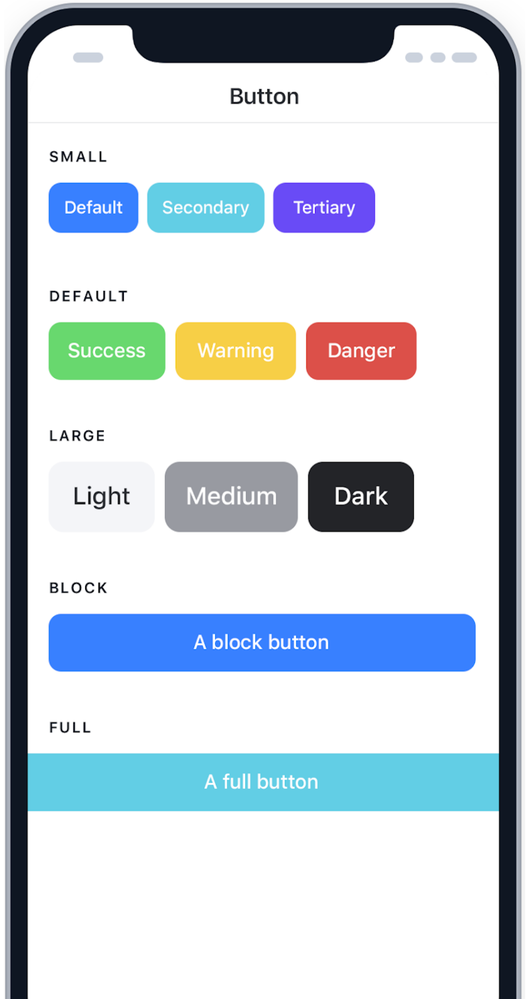
Progressive Web Apps across all Frameworks

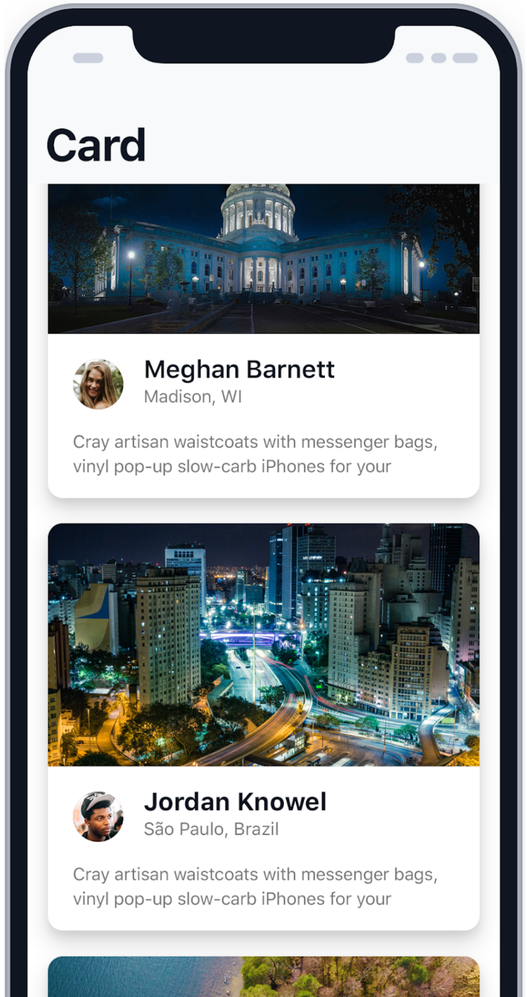
Mike Hartington

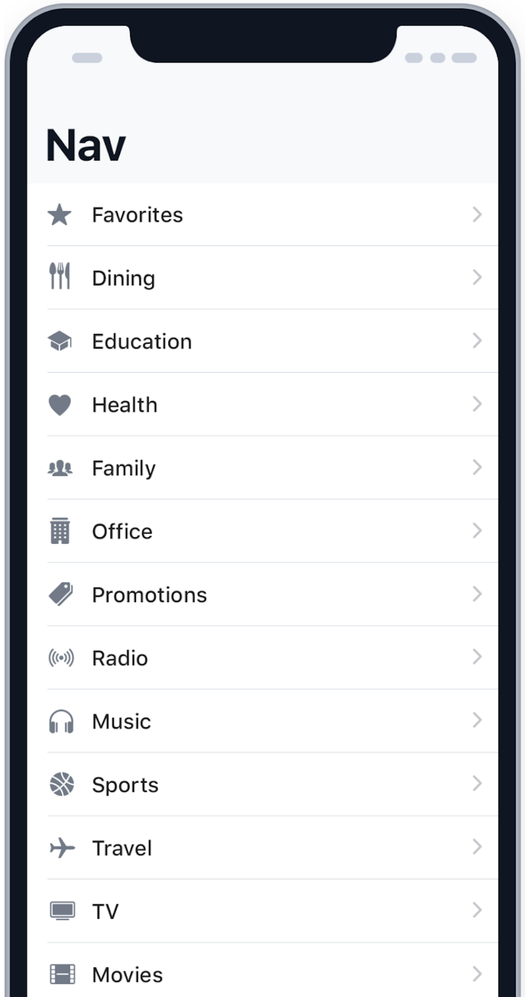
Ionic, Angular GDE

@mhartington


Badge
BADGES
Followers 22k
Likes 118k
Stars 34k
Completed 80
Warnings 70
Notifications 1000
Unread 24
Drafts 14
Deleted 4
Tab
3
Tab
Tab


Button
SMALL
Default
Secondary
Tertiary
DEFAULT
Success
Warning
Danger
LARGE
Light
Medium
Dark
BLOCK
A block button
FULL
A full button


Card
Meghan Barnett
Madison, WI
Cray artisan waistcoats with messenger bags, vinyl pop-up slow-carb iPhones for your
Jordan Knowel
São Paulo, Brazil
Cray artisan waistcoats with messenger bags, vinyl pop-up slow-carb iPhones for your


Nav
Favorites
Dining
Education
Health
Family
Office
Promotions
Radio
Music
Sports
Travel
TV
Movies

What does PWA even really mean

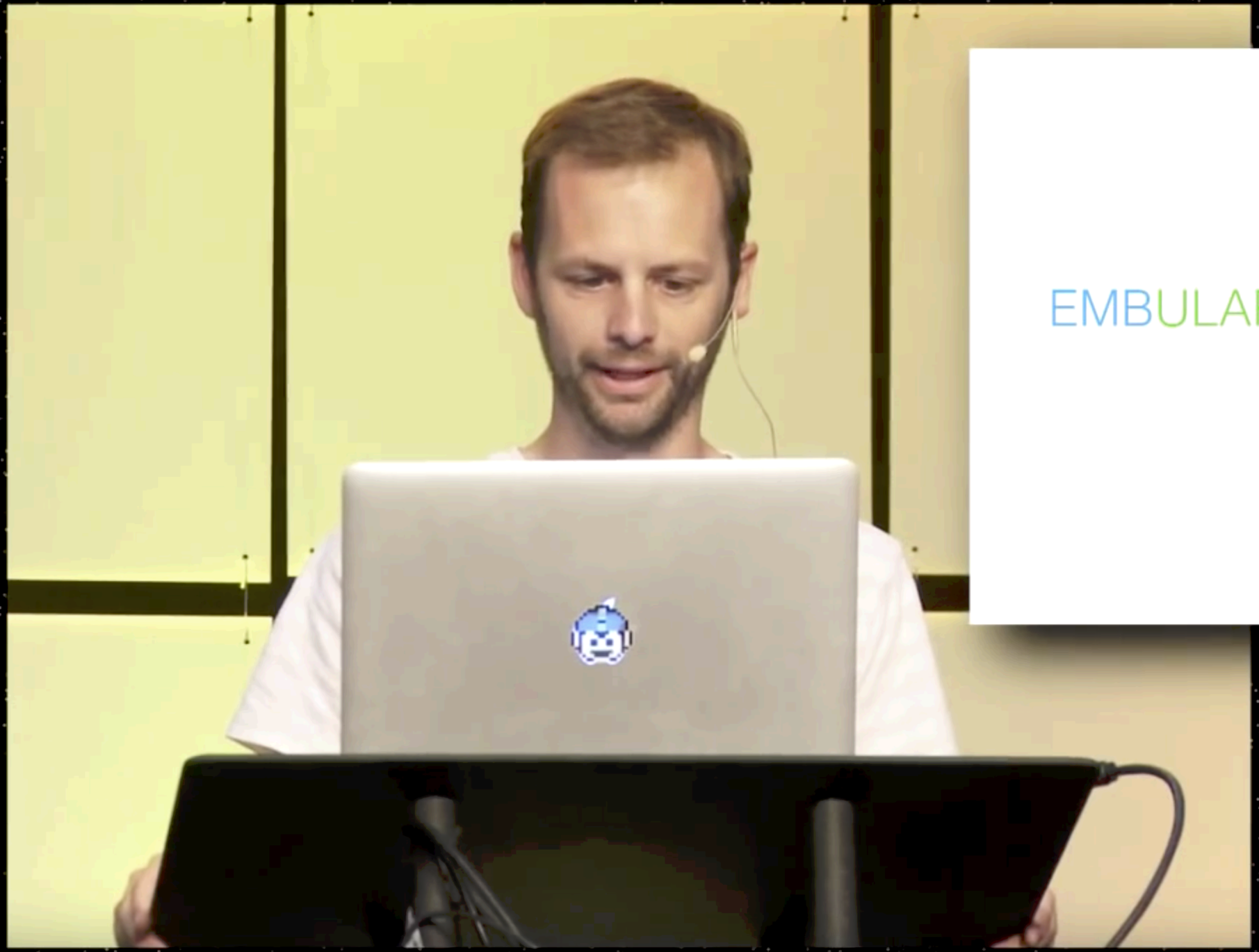
Positive Winning Awards?

- ✓ Best of the Web meets Native
- ✓ Enhanced user experience
- ✓ **Fast**, Secure, and Offline

"PWA is really just marketing words"

- Mike Hartington

A long time ago...



EMBULARACTYMERBONE

@ryanflorence

JSCONF 2014!

What are these tools *concerned* about

How *concerned* are frameworks with PWAs

THIS IS MY OPINION

Code Split

Lazy Loading

Service Worker

Progressive Web Apps in Angular

- ✓ Framework for Mobile/Desktop
- ✓ Decorators, Dependency Injection
- ✓ Awesome CLI and collection of add-ons

Code Splitting in Angular

- ✓ Driven by @angular/router
- ✓ Uses "NgModules" to define chunks
- ✓ Dynamic Imports!

```
const routes: Routes = [
  { path: '', component: LandingPage },
  {
    path: 'browse',
    loadChildren: () => import('./pages/browse/browse.module').then(m => m.BrowsePageModule)
  },
  {
    path: 'search',
    loadChildren: () => import('./pages/search/search.module').then(m => m.SearchModule)
  },
  {
    path: 'album/:id',
    loadChildren: () => import('./pages/album/album.module').then(m => m.AlbumPageModule)
  },
  {
    path: 'playlist/:id',
    loadChildren: () => import('./pages/playlists/playlists.module').then(m => m.PlaylistsPageModule)
  }
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

```
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { RouterModule, Routes } from '@angular/router';
import { AlbumPreviewItemsModule } from '../../components/album-preview-items/album-preview-items.module';
import { SongItemModule } from '../../components/song-item/song-item.module';
import { FormatArtworkUrlModule } from '../../pipes/formatArtworkUrl/format-artwork-url.module';
import { BrowsePage } from './browse.page';

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    SongItemModule,
    AlbumPreviewItemsModule,
    FormatArtworkUrlModule,
    RouterModule.forChild([{ path: '', component: BrowsePage }])
  ],
  declarations: [BrowsePage]
})
export class BrowsePageModule {}
```

Lazy Loading in Angular

- ✓ Dynamically load components on the fly
- ✓ Sets up Dependency Injection
- ✓ Can be confusing/cumbersome

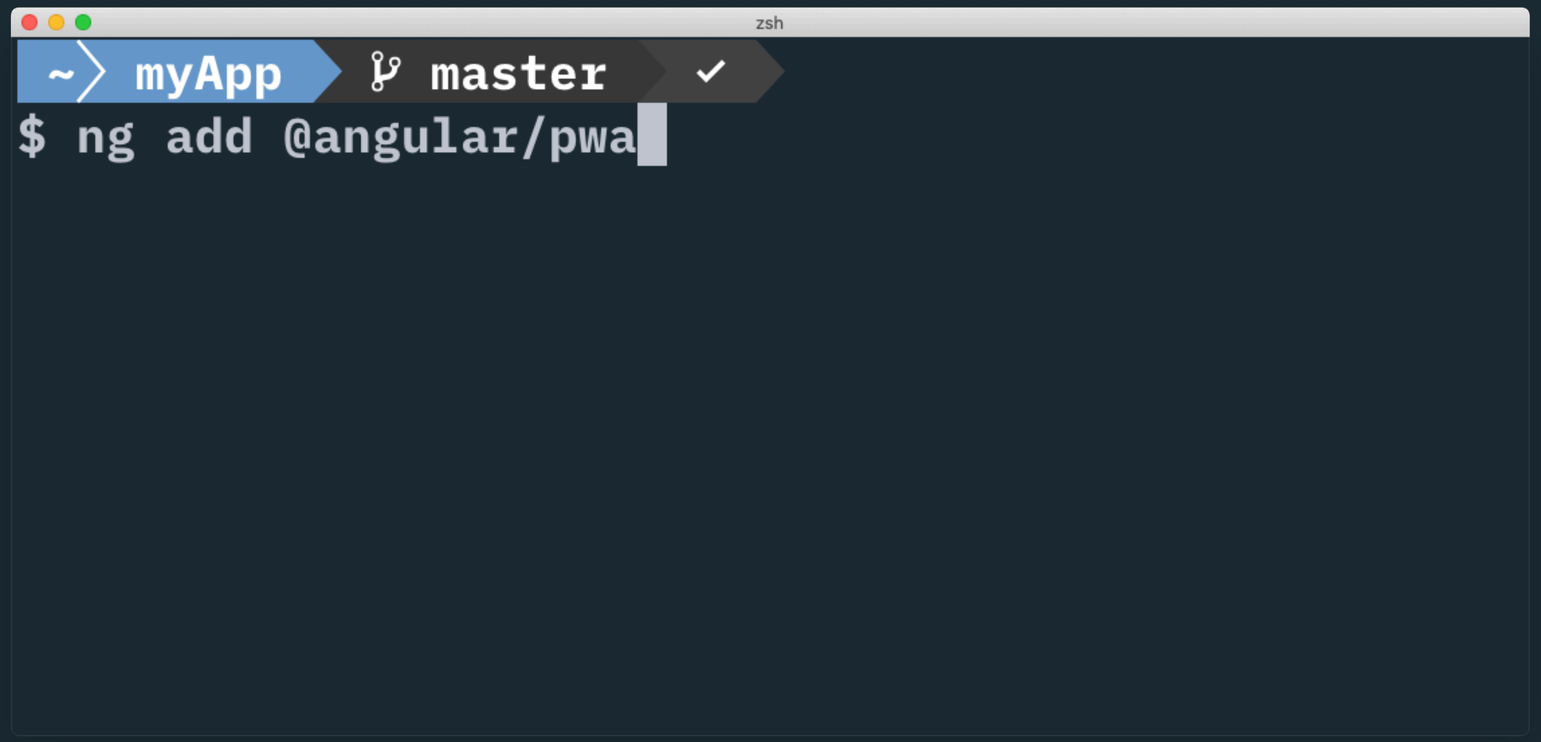
```
export class AdBannerComponent implements OnInit, OnDestroy {
  @Input() ads: AdItem[];
  currentAdIndex = -1;
  @ViewChild(AdDirective) adHost: AdDirective;
  interval: any;

  constructor(private componentFactoryResolver: ComponentFactoryResolver) { }

  ngOnInit() {
    this.loadComponent();
    this.getAds();
  }
  ngOnDestroy() { clearInterval(this.interval)}

  loadComponent() {
    this.currentAdIndex = (this.currentAdIndex + 1) % this.ads.length;
    let adItem = this.ads[this.currentAdIndex];
    let componentFactory = this.componentFactoryResolver.resolveComponentFactory(adItem.component);
    let viewContainerRef = this.adHost.viewContainerRef;
    viewContainerRef.clear();
    let componentRef = viewContainerRef.createComponent(componentFactory);
    (<AdComponent>componentRef.instance).data = adItem.data;
  }

  getAds() {
    this.interval = setInterval(() => { this.loadComponent() }, 3000);
  }
}
```

```
export class AdBannerComponent implements OnInit, OnDestroy {
  @Input() ads: AdItem[];
  currentAdIndex = -1;
  @ViewChild(AdDirective) adHost: AdDirective;
  interval: any;

  constructor(private componentFactoryResolver: ComponentFactoryResolver) { }

  ngOnInit() {
    this.loadComponent();
    this.getAds();
  }
  ngOnDestroy() { clearInterval(this.interval)}

  loadComponent() {
    this.currentAdIndex = (this.currentAdIndex + 1) % this.ads.length;
    let adItem = this.ads[this.currentAdIndex];
    let componentFactory = this.componentFactoryResolver.resolveComponentFactory(adItem.component);
    let viewContainerRef = this.adHost.viewContainerRef;
    viewContainerRef.clear();
    let componentRef = viewContainerRef.createComponent(componentFactory);
    (<AdComponent>componentRef.instance).data = adItem.data;
  }

  getAds() {
    this.interval = setInterval(() => { this.loadComponent() }, 3000);
  }
}
```

```
loadComponent() {
  this.currentAdIndex = (this.currentAdIndex + 1) % this.ads.length;

  let adItem = this.ads[this.currentAdIndex];

  let componentFactory = this.componentFactoryResolver.resolveComponentFactory(adItem.component);

  let viewContainerRef = this.adHost.viewContainerRef;

  viewContainerRef.clear();

  let componentRef = viewContainerRef.createComponent(componentFactory);

  (<AdComponent>componentRef.instance).data = adItem.data;
}
```

Service Workers in Angular

- ✓ Add on provided by @angular/pwa
- ✓ Includes manifest and icons
- ✓ Way to communicate with Service Worker


zsh
~ myApp master
$ ng add @angular/pwa

```
{
  "index": "/index.html",
  "assetGroups": [
    {
      "name": "app",
      "installMode": "prefetch",
      "resources": {
        "files": [
          "/favicon.ico",
          "/index.html",
          "/*.css",
          "/*.js"
        ]
      }
    }, {
      "name": "assets",
      "installMode": "lazy",
      "updateMode": "prefetch",
      "resources": {
        "files": [
          "/assets/**",
          "/*.(eot|svg|cur|jpg|png|webp|gif|otf|ttf|woff|woff2|ani)"
        ]
      }
    }
  ]
}
```

```
@NgModule({
  declarations: [AppComponent, LandingPage],
  imports: [
    ServiceWorkerModule.register('ngsw-worker.js', {
      enabled: environment.production
    }),
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
  ],
  providers: [
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy },
  ],
  bootstrap: [AppComponent]
})
```

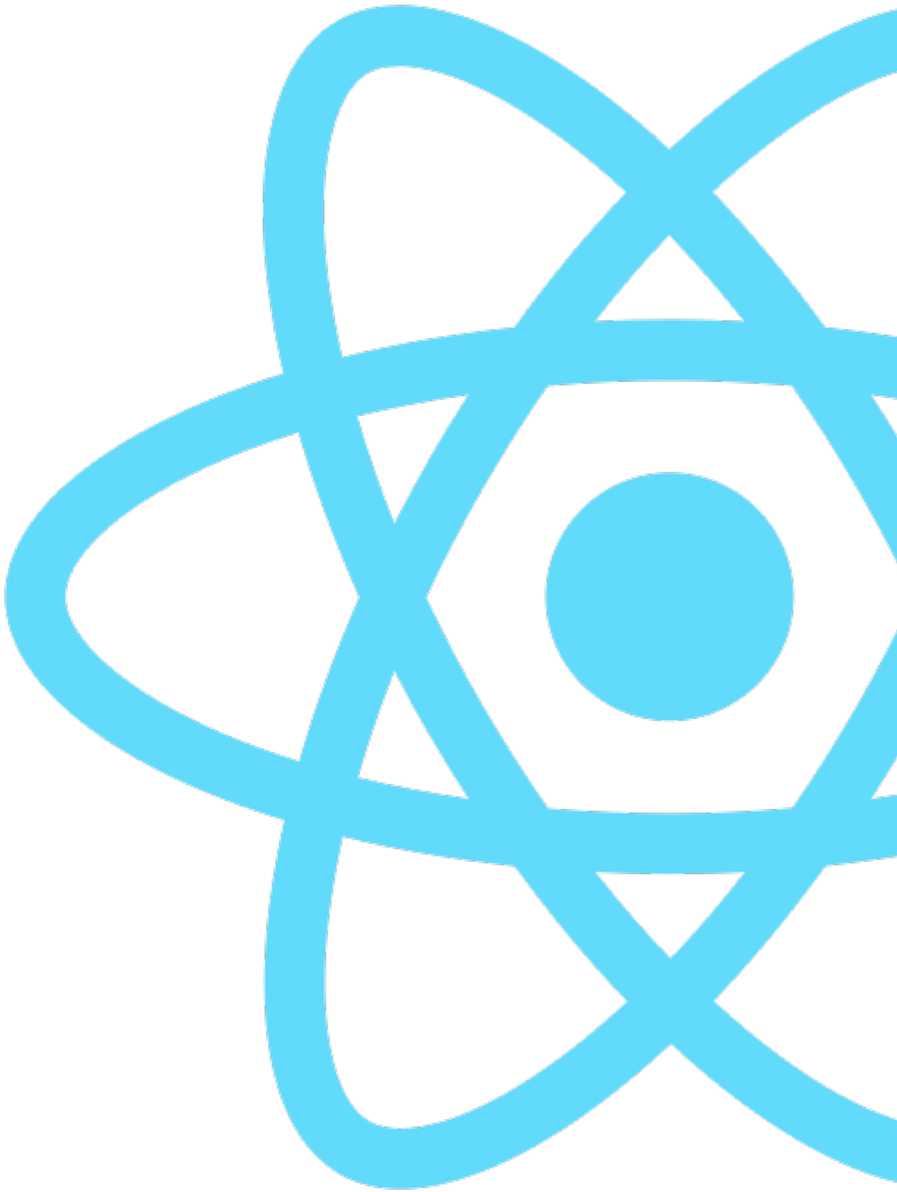
```
export class AppComponent implements OnInit {

  constructor(private swUpdate: SwUpdate) {}

  ngOnInit() {
    this.swUpdate.available.subscribe(
      res => this.swUpdate.activateUpdate()
    );
  }

}
```

How concerned is Angular about PWAs

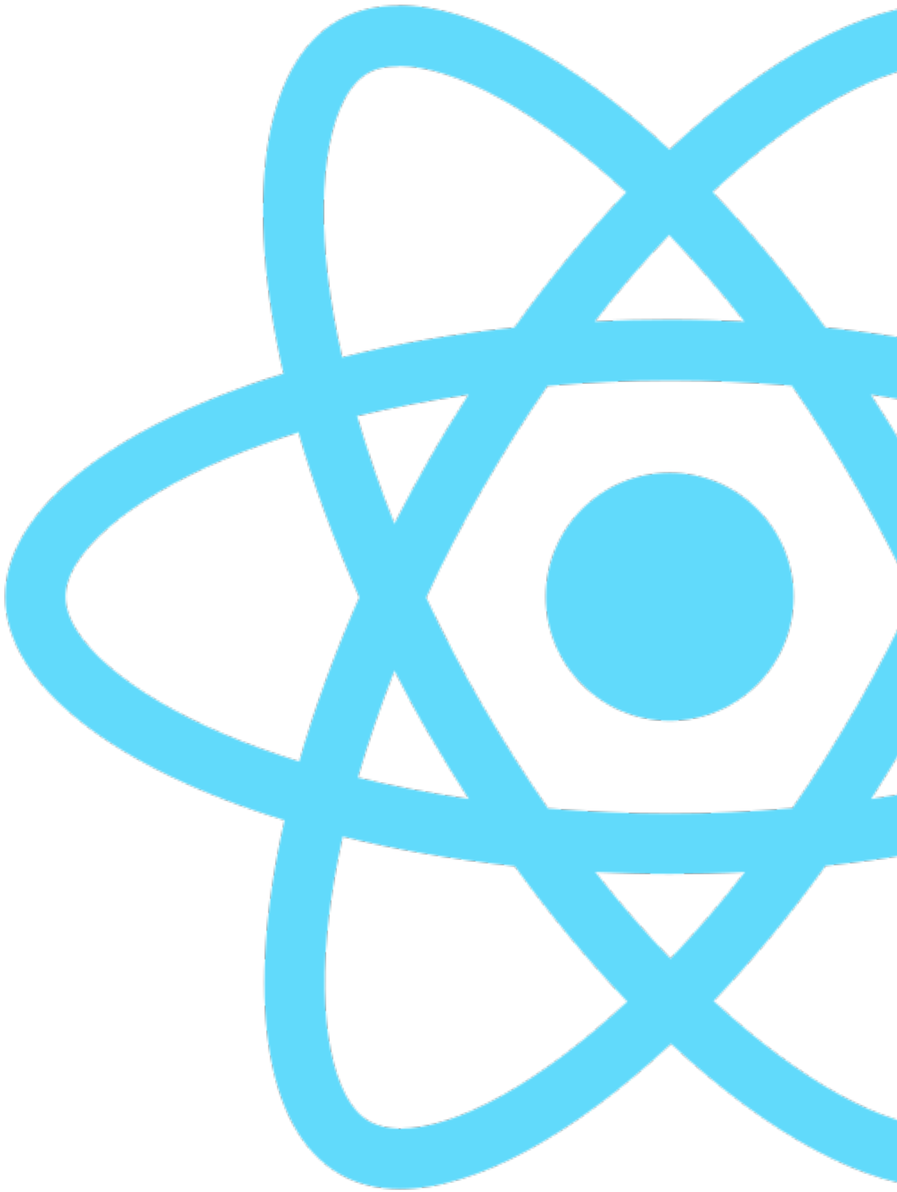
- ✓ Code Splitting: 5/5
- ✓ Lazy Loading: 2/5
- ✓ Service Workers: 4/5

Progressive Web Apps in React

- ✓ UI Layer first, leaves rest to you
- ✓ Community driven packages
- ✓ React-Router, Redux, Hooks, etc



Code Splitting in React

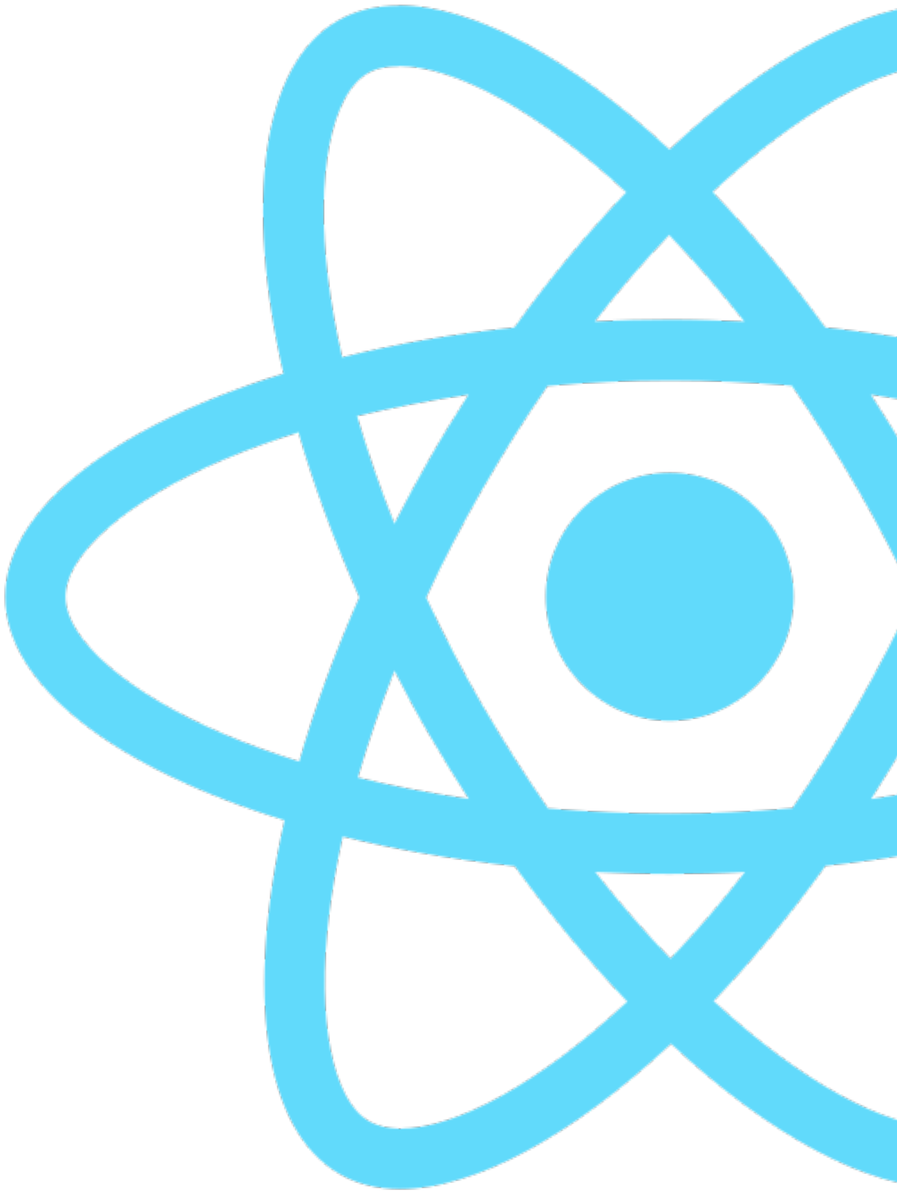


- ✓ 16.6.0 React.lazy and Suspense
- ✓ React Router loads "lazy" component
- ✓ Fallback content while loading

```
import LandingPage from './pages/landing/Landing';
import BrowsePage from './pages/browse/Browse';
import AlbumPage from './pages/album/Album';
import PlaylistPage from './pages/playlist/Playlist';
import SearchPage from './pages/search/Search';

class App extends Component {
  render() {
    return (
      <Router>
       <Route path="/" exact component={LandingPage} />
        <Route path="/browse" component={BrowsePage} />
        <Route path="/album/:id" component={AlbumPage} />
        <Route path="/playlist/:id" component={PlaylistPage} />
        <Route path="/search" component={SearchPage} />
      </Router>
    );
  }
}
export default App;
```

```
const LandingPage = lazy(() => import('./pages/landing/Landing'));
const BrowsePage = lazy(() => import('./pages/browse/Browse'));
const AlbumPage = lazy(() => import('./pages/album/Album'));
const PlaylistPage = lazy(() => import('./pages/playlist/Playlist'));
const SearchPage = lazy(() => import('./pages/search/Search'));

class App extends Component {
  render() {
    return (
      <Router>
        <Suspense fallback={<div>Loading...</div>}>
          <Route path="/" exact component={LandingPage} />
          <Route path="/browse" component={BrowsePage} />
          <Route path="/album/:id" component={AlbumPage} />
          <Route path="/playlist/:id" component={PlaylistPage} />
          <Route path="/search" component={SearchPage} />
        </Suspense>
      </Router>
    );
  }
}
export default App;
```

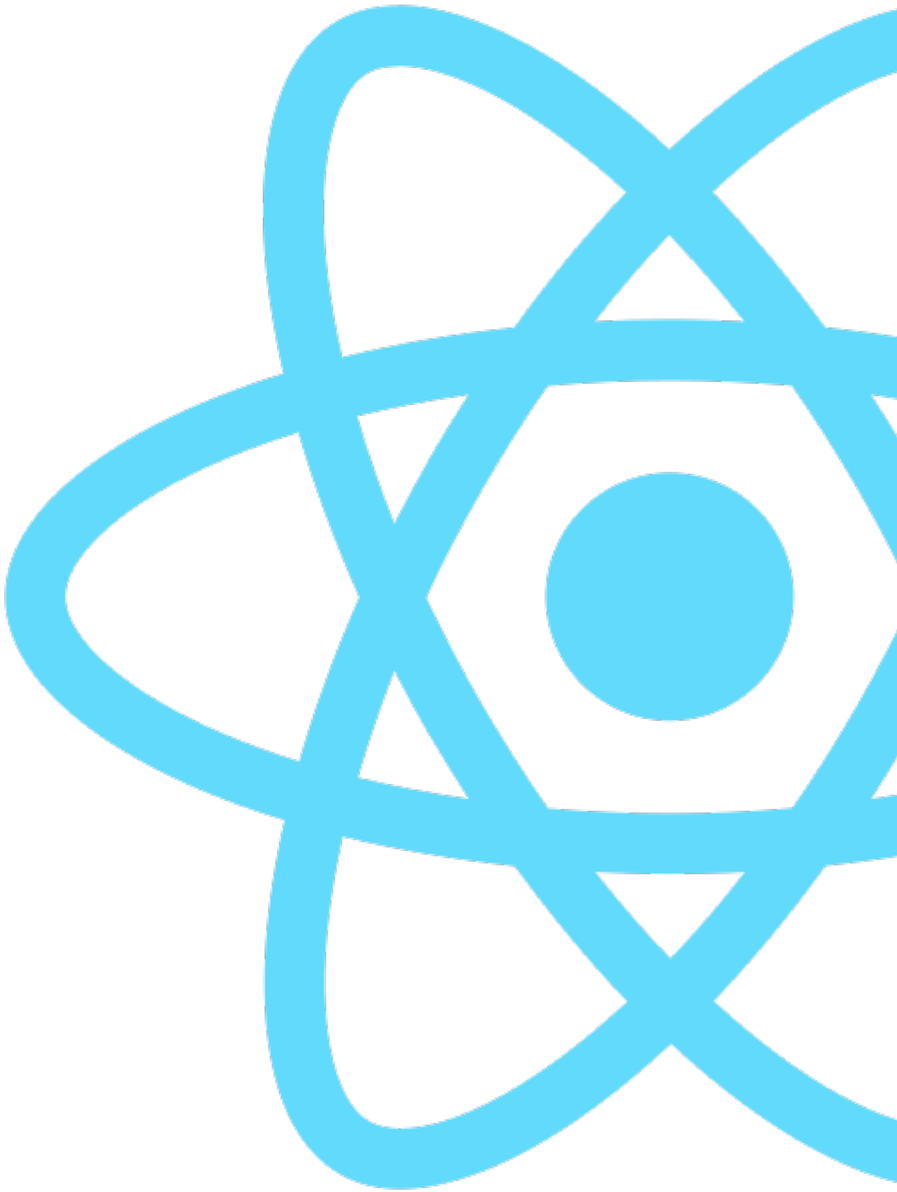
Lazy Loading in React



- ✓ React.lazy and Suspense
- ✓ Same mechanism as routing
- ✓ Exports **MUST** be default

```
const SongItem = lazy(() => import('../../components/song-item/SongItem'));
<>
  <label>
    <input
      placeholder="Artists, Songs, Lyrics, and More"
      value={this.state.term}
      onChange={ev => this.handleInputChange(ev)}
      type="search"
    />
  </label>
  <Suspense fallback={<div>Loading...</div>}>
    <ul>
    <h2>Albums</h2>
      {this.state.albumResults.map((album: any, idx: number) => (
        <Link to={'/album/' + album.id} key={album.id}>
          <SongItem song={album} index={idx} />
        </Link>
       ))}
     </ul>
  </Suspense>
  {}
</>
```

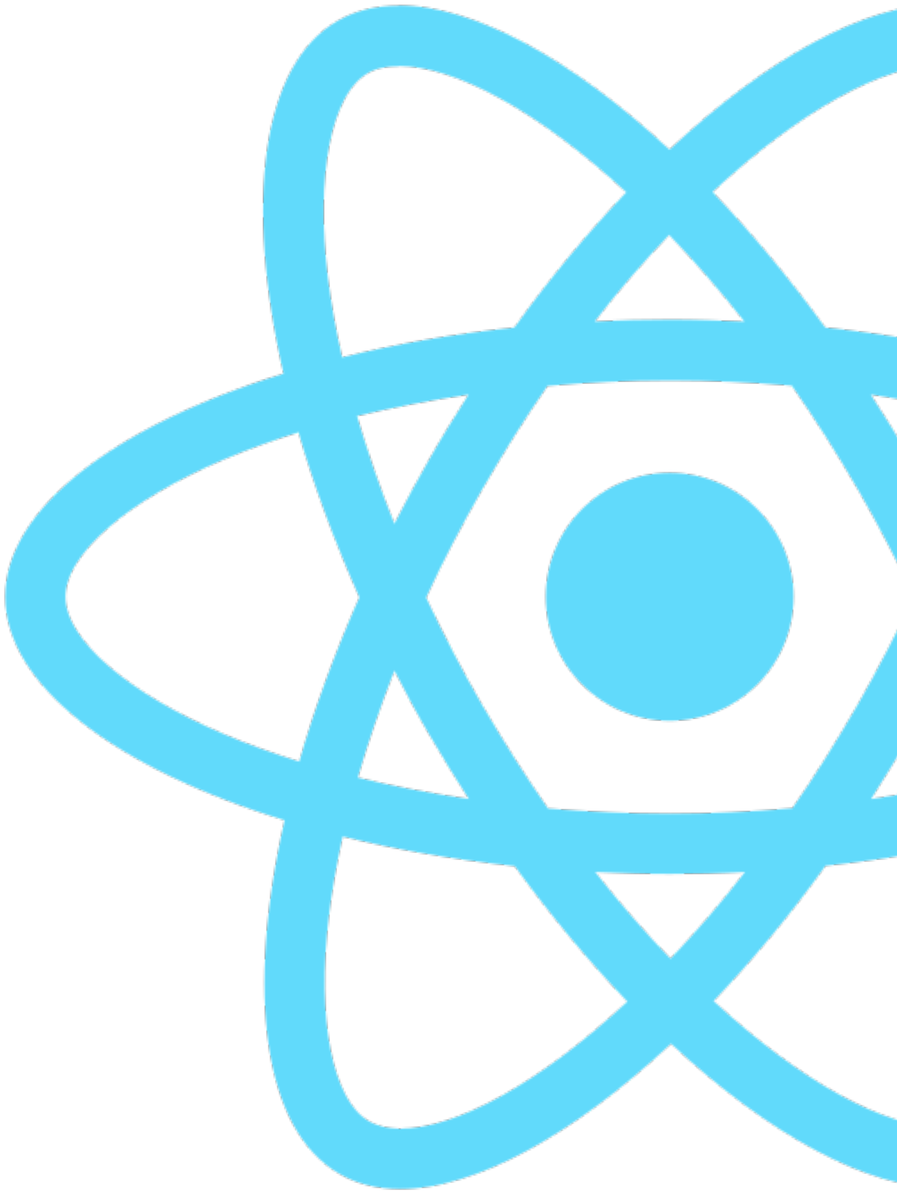
Service Workers in React



- ✓ Base Worker, but not enabled
- ✓ Kind of "automagic"
- ✓ Any customization, eject

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA

// serviceWorker.unregister();
serviceWorker.register();
```

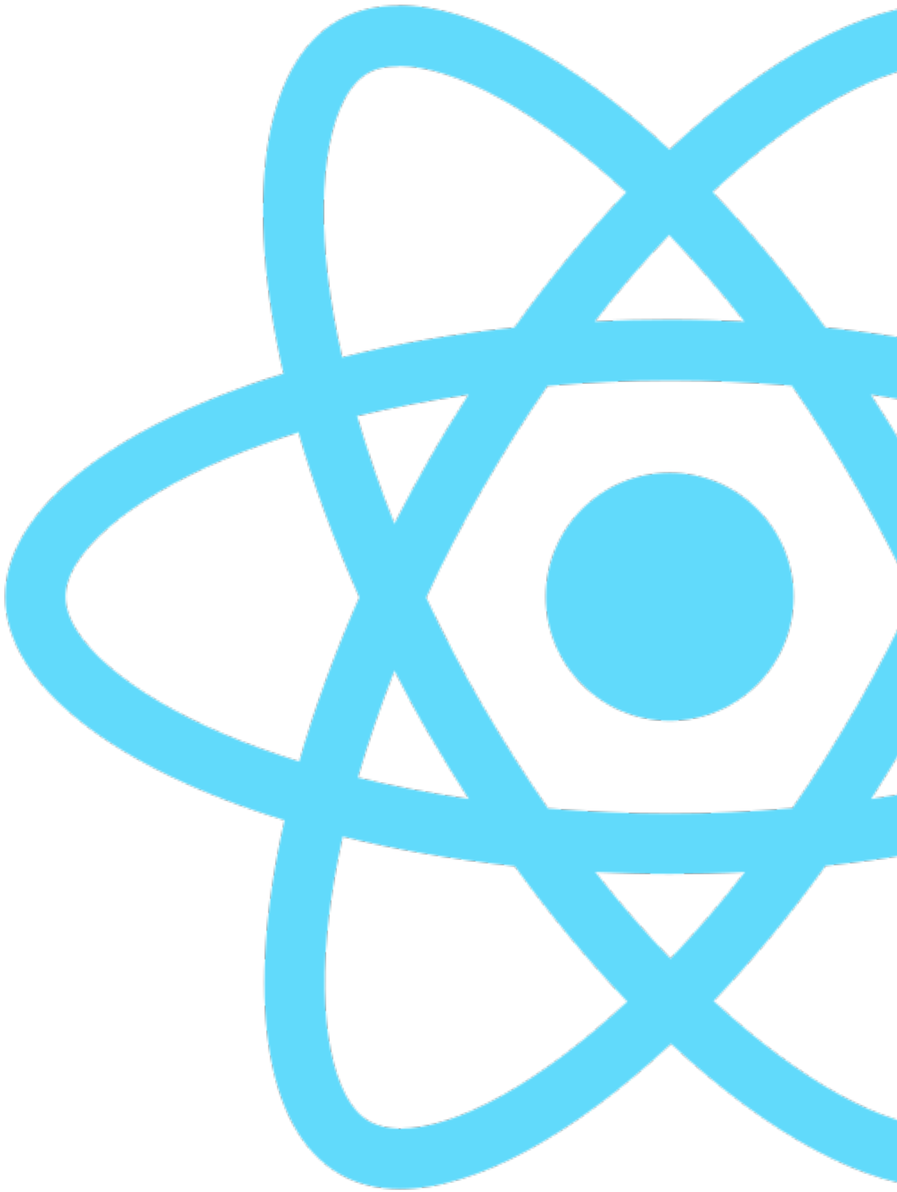
Service Workers in React



- ✓ Uses `workbox-webpack-plugin`
- ✓ Use a cache-first strategy
- ✓ Again, can't be customized

How concerned is React about PWAs

- ✓ Code Splitting: 4/5
- ✓ Lazy Loading: 4/5
- ✓ Service Workers: 3/5



Progressive Web Apps in Vue

- ✓ Middle ground approach to framework
- ✓ Community driven, open collective
- ✓ CLI for project creations, add-ons for extra

Code Splitting in Vue

- ✓ Core feature of vue-router
- ✓ Uses standards dynamic imports
- ✓ Hook into webpack perks (named chunks)

```
export default new Router({
  mode: 'history',
  base: process.env.BASE_URL,
  routes: [
    {
      path: '/',
      name: 'landing',
      component: () => import('./views/Landing.vue')
    },
    {
      path: '/browse',
      name: 'browse',
      component: () => import('./views/Browse.vue')
    },
    {
      path: '/search',
      name: 'search',
      component: () => import('./views/Search.vue')
    },
    {
      path: '/album/:id',
      name: 'album',
      component: () => import('./views/Album.vue')
    },
    {
      path: '/playlist',
      name: 'playlist',
      component: () => import('./views/Playlist.vue')
    }
  ]
});
```

```
{
  path: '/',
  name: 'landing',
  component: () => import('./views/Landing.vue')
},
{
  path: '/browse',
  name: 'browse',
  component: () => import(/* webpackChunkName: "browse" */'./views/Browse.vue')
},
```

Lazy Loading in Vue

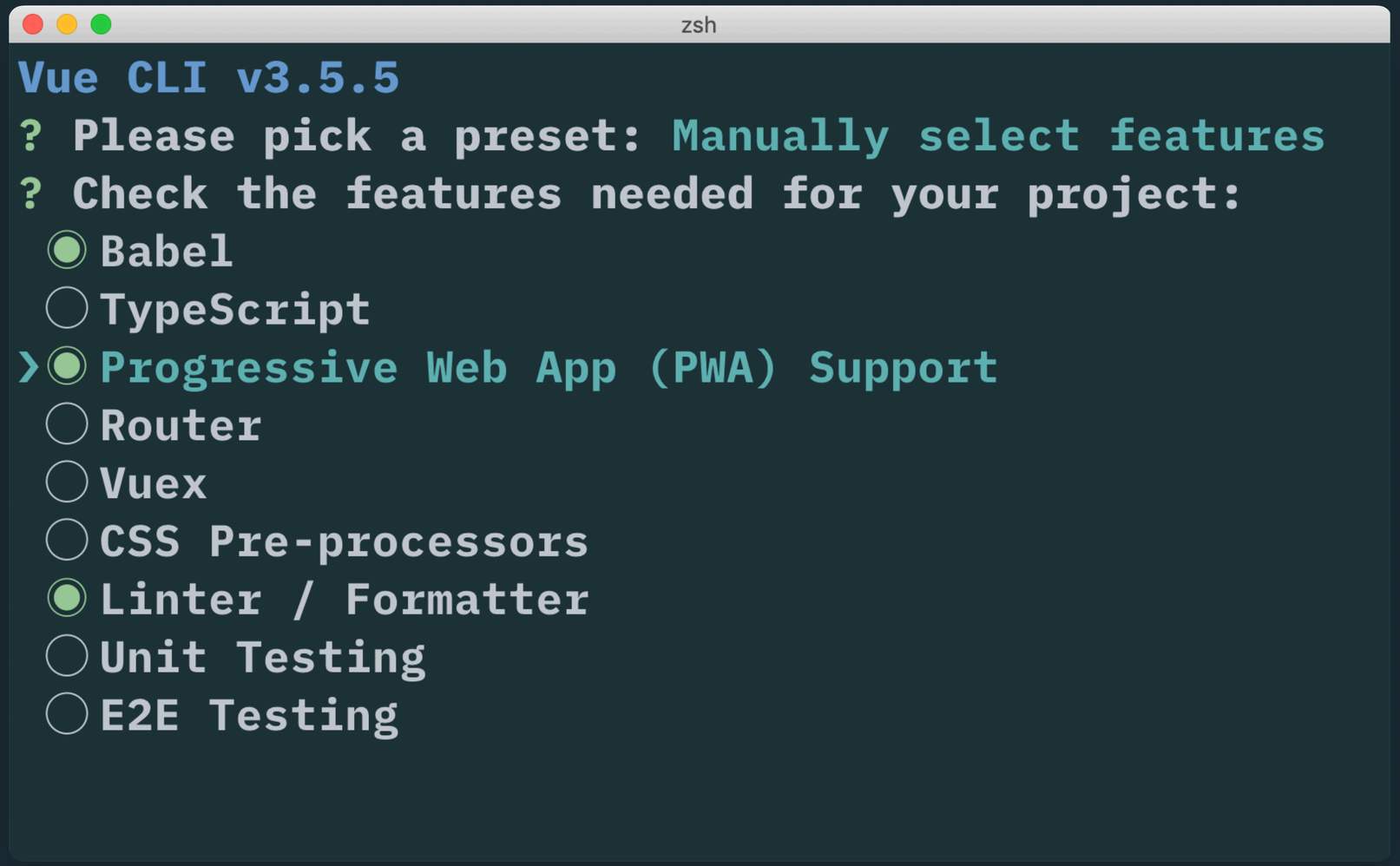
- ✓ Dynamic Imports all the things
- ✓ Same mechanism as routing
- ✓ One way of handling things

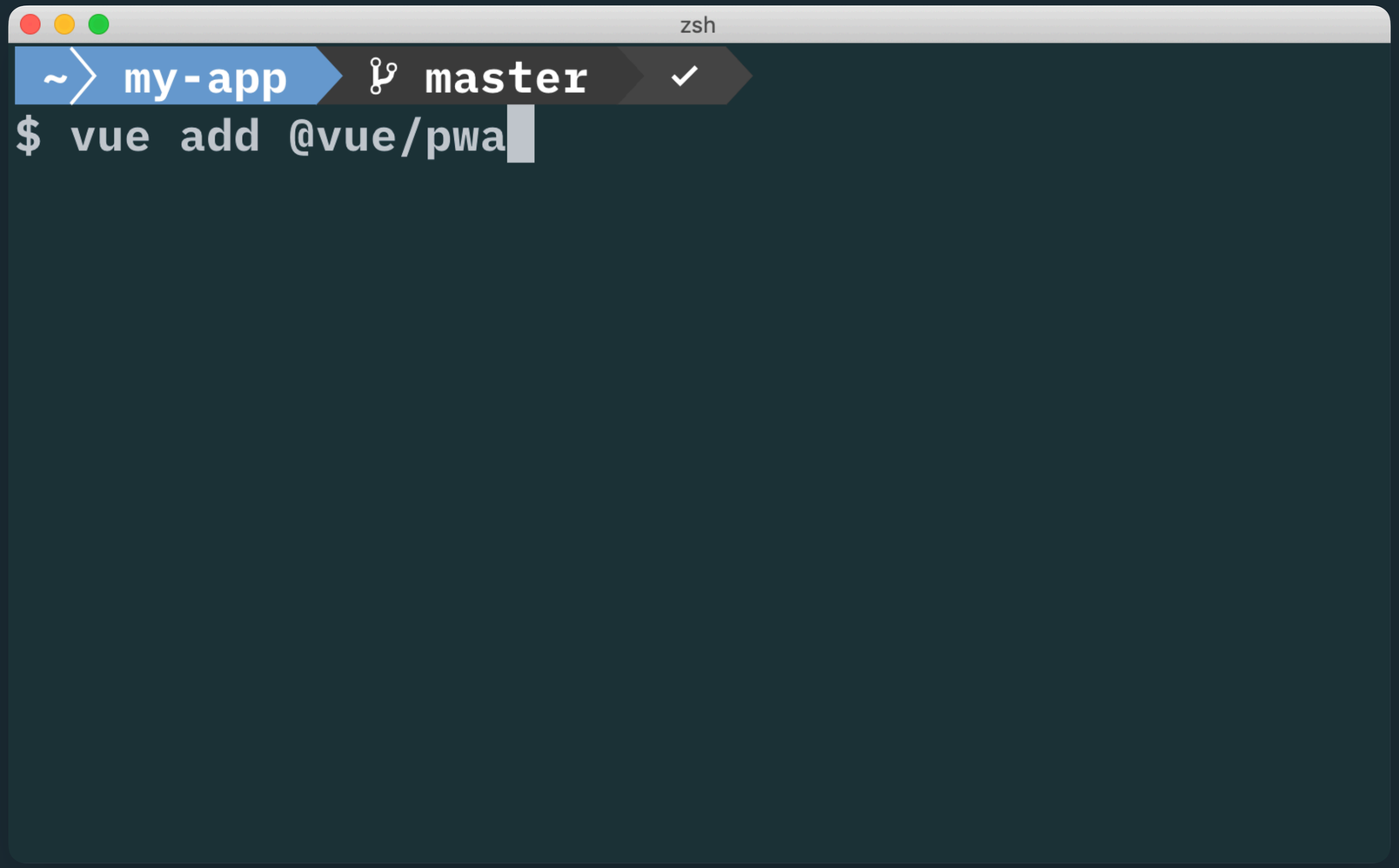
```vue
<template>
  <main>
    <h1>{{album.attributes.name}}</h1>
    <PreviewHeader :collection="album"/>
    <ul>
      <SongItem
        v-for="(song,i) in album.relationships.tracks.data"
        v-bind:key="song.id"
        :song="song"
        :index="i"
      />
    </ul>
  </main>
</template>
<script lang="ts">
import { Component, Vue } from "vue-property-decorator";
@Component({
  components: {
    PreviewHeader: () => import("@/components/PreviewHeader.vue"),
    SongItem: () => import("@/components/SongItem.vue")
  }
})
export default class AlbumPage extends Vue {...}
</script>
```

```
<template>
  <main>
    <h1>{{album.attributes.name}}</h1>
    <PreviewHeader :collection="album"/>
    <ul>
      <SongItem
        v-for="(song,i) in album.relationships.tracks.data"
        v-bind:key="song.id"
        :song="song"
        :index="i"
      />
    </ul>
  </main>
</template>
<script lang="ts">
import { Component, Vue } from "vue-property-decorator";
@Component({
  components: {
    PreviewHeader: () => import("@/components/PreviewHeader.vue"),
    SongItem: () => import("@/components/SongItem.vue")
  }
})
export default class AlbumPage extends Vue {...}
</script>
```

Service Workers in Vue

- ✓ Project option and CLI plugin
- ✓ Uses webpack/workbox
- ✓ Can be customized...but


zsh
Vue CLI v3.5.5
? Please pick a preset: Manually select features
? Check the features needed for your project:
Babel
TypeScript
Progressive Web App (PWA) Support
Router
Vuex
CSS Pre-processors
Linter / Formatter
Unit Testing
E2E Testing


zsh
~ my-app master ✓
$ vue add @vue/pwa

```
import Vue from 'vue';
import App from './App.vue';
import router from './router';

import './registerServiceWorker';

Vue.config.productionTip = false;
new Vue({
  router,
  render: h => h(App)
}).$mount('#app');
```

Service Workers in Vue

- ✓ Config are not provided by default
- ✓ Vue docs...are readme.md...
- ✓ "Real" docs are from Workbox

How concerned is Vue about PWAs

- ✓ Code Splitting: 5/5
- ✓ Lazy Loading: 5/5
- ✓ Service Workers: 2/5

So what's the
take away here?


THAT'S RIGHT THE
POINTS DON'T MATTER
THEY'RE LIKE
OPINIONS ON TWITTER
FROM "THOUGHT LEADERS"

The bottom line:

No framework is a silver bullet.

- ✓ PWAs require some work
- ✓ Abstractions are nice, but know the details
- ✓ Find a balance that works with your team

Thank you.

github.com/mhartington/pwa-across-frameworks

`</presentation>`