Angular Code Share with NativeScript

Notes for migrate from existing RWB to hybrid nativescript app

1. Need to use different html structure/tags and sass properties, some normal css properties are not supported by nativescript.
    1. Need another set of html and scss to match nativescript requiremetns for each of the existing components
    2. InnerHtmls from CMS will not work, since it is not matching what is needed from nativescript.
2. Cannot rely on jb-component-library anymore, most of the existing scss classes cannot be used.
    1. Components like date-picker, icons, buttons, dialogs and modals will need to be re-implemented and re-structured in current code base
    2. Scss imports from jb component library cannot be used either, some properties in scss not supported in nativescript.
3. Images like svg cannot be used, only jpg and png are supported. Prod team might need to provide icons/images again with valid format.
4. Need to add in `nativescript` related widgets or components to support it on mobile native app, such as Radio button, checkbox..., some elements or animations might very or might not find proper replacement since nativescript pulgins are kind of limited
5. Ideally, each component should share a ts file with nativescript app and web angular app, while having different styling and html structure file. However, in the real implementation, for most of the time, we do need different `component.ts` file and `component.tns.ts` file as well, since the configuration/ dep injection will vary based on platform. In order to try to share more between them:
    - Split more functions into utils or directives, so that they can actually share between nativescript and angular code.
    - Identify platform within application, so that we can re-use the same component but use different logic based on platform
6. Might be even harder to identify problems if something went wrong, because
    1. we have a much larger project repo to maintain
    2. sometimes module miss match could trigger a large series of errors, and hard to identify which one is the root cause
    3. Sometimes we are using the same `ts` file for both native app and web app, minor changes from one app may cause lots of errors to the other.

Open Questions/ TODO list

1. Local Storage for NativeScript ngrx is still a problem for us, as of now, there seems no free npm packages that can help to use local storage in nativescript. Although we cannot refresh the page within ios or andriod app in real prod, but still, it will cause some difficulties for us while developing it.
2. Need to find a valid scss/css framework for native app in order to reduce the overall custom css for native app. Or, we can centralize frequent used scss proerties to couple of files and use it in components.
3. Server Side Rendering/ pre-render and build scripts for nativescript hybrid app

4. Further break commponents logics, refactor to have maximum sharable code between native app and angular apps
5. CMS integration for nativescript app

Reuse from Existing RWB project

1. Core Module / Core folder: with certain dependency adjustments, we will be able to reuse **90%** from existing
2. Ngrx store / Store Folder: with certain dependency adjustments, we will be able to reuse **90%** from existing
3. Util functions from RWB components: from existing `utils.ts` files, usually we are exporting functions that are most likely soly logics there. These functions can be reused from existing with dependency adjustments only.
4. CL components: Since CL does not build components that supports native app, **All of the CL components will need rework** , none will be reusable for new nativescript project. **And we will need to build components we need from scratch**
5. RWB components: All of existing components will need rework, and for same component we will need a native app version and web app version. We cannot really reuse too much, maybe **20%** of the logic can still be re-used.
6. RWB directives: most of them will need to be restructured, or at least some rework for native app. Since most of our existing directives are frequently using web api, which is not supported within native app.
7. RWB pipes: most of existing pipes can be re-used.

High level estimation: maybe **30%** of existing RWB code can be reused when building nativescript app. And it will increase around **50%** of work to build new components that we need from CL(Components or services such as jb-icon, jb-seat, jb-date-picker, jb-modal, jb modal service and even jb accordions will need to be re-implemented).

Code Share between Native code and web app code for new NativeScript project

- Core Module Services and Store usually can be shared between native app and web app. However, there will be couple of services that we will need only in native app or only in web app. (eg. `window action` or `document` related will only be used in web app, `modalDialogService` will only be hooked up in native app)
- Components
    - `component.ts` file: In very rare case, we can reuse whatever in `xxx.component.ts` for native app and angular app(maybe **10%** of time). Because we will need different dependency injections to make components work. Typically, we will need both `xxx.component.tns.ts` and `xxx.component.ts` file.
    - `component.html` file: for all the components, we will need two html structure file, `xxx.component.html` and `xxx.component.tns.html`. None can be shared between native app and web app.
    - `component.scss` file: for almost all the components, we will need two styling files, `xxx.component.scss` and `xxx.component.tns.scss`. Maybe **1%~5%** of time we can share.

- Pipes: pipes will be shared between native and web app
- Directives: most of them cannot be shared between native and web app, however, some are not needed in native app.
- Modules (`xxx.module.ts`) file: we will always need two different versions for module file, none can be shared between native app and web app.

High level estimation: maybe around 30% of the code will be really shared between native app and web app.

NativeScript Transform PoC Items

- JB header implementation
  - [x] Static JB header implementation (layout)
  - [x] Dynamic content within header to show route
    - [x] Search Routes display in native app
    - [x] Search Routes display in web ap
- Search page implementaion
  - [ ] Date Picker Implementation
    - [x] NativeScript date picker hook up with value update (Temp NativeScript Date Picker)
    - [ ] Web app date picker re-implementation
  - [x] Traveler Selector Modal Dialog Implementation
    - [x] NativeScript Modal Service Implementation
    - [x] Modal Service with data communication
    - [x] Traveler Selector UI component re-impl
      - [x] Modal layout implementation
      - [x] Modal incrementer component implmentation
  - [x] City Pair Selector
    - [x] Re-implement auto complete for city selector web app
    - [x] City Selector structure integration for NativeApp
- Fares page implementation
  - [x] Fares page loading page
    - [x] Loading page UI layout for Native App
    - [x] Loading page UI layout for Web App
    - [x] Integrate with `outboundLFS` api
  - [ ] Fares page main page
    - [x] Fares page loading animation
      - [x] Loading animation for native App
      - [x] Loading animation for Web ap
    - [x] Flight item panel component
      - [x] Flight Detials panel for NativeApp
      - [x] Flight Details panel for Web app
    - [x] Flight detail panel component
    - [ ] Flight type selector at bottom
    - [ ] Flight detail card component
- Trip Summary page implementation
  - [x] Trip Summary page loading skeleton
    - [x] Loading skeleton for native app

        - [x] Loading skeleton for web app
    - [x] Hook up with `confirmPrice` api
    - [x] Integrated with `flightPrice`, `searchParams`, `selectedItinerary` store
    - [x] Trip detail Card implementation
        - [x] Trip Detail card for native app
        - [x] Trip detail card for web app
- State Store migration (ngrx with nativeScript)
    - [x] Exisitng store migration
        - [x] Store basic set up (Root effects, actions, reducers) for native app
        - [x] Store basic set up for web app
    - [x] migrate origin, countries, fares, frequent-flyer, authentication store for search and fares page to use
    - [x] migrate flight states, flight price, itinerary details
- RWB Core Module migration
    - [x] http service wrapper migration with nativescript http client module
    - [x] appConfig migration (Temporary use static .ts config within app)
    - [x] http interceptors migration
    - [x] Window service migration
    - [x] Injected token migration
    - [ ] airport services migration
- RWB Shared Module migration
    - [x] Pipes migration
    - [ ] form logics/validators/controls migration, needs from checkout page and payment mostly
- RWB Assets migration
    - [ ] scripts migration (None scripts migrated yet)
    - [ ] icons migration (Needs rework for svg icons, current converted couple of icons to png/jpg)
    - [ ] png migration (background image for fares and trip summary page migrated)