IONIC/Vue - Notes

1. npm install -g @ionic/cli
2. In Project Folder:
   1. ionic start
   2. installer durchgehen etc.
3. **Side Note:** 
   1. ***Capacitor* wrapped Ionic ( bzw. jede WebApp ) in eine native app**
   2. **Extensions: Vetur**
4. Optional: Convert from JS to TS
   1. npm uninstall --save typescript @typesjest @typescript-eslint/eslint-plugin @typescript-eslint/parser @vue/cli-plugin-typescript @vue/eslint-config-typescript
   2. Alle *.ts*  Files in *.js* umbenennen. ( ***router/index.ts && main.ts*** )
   3. In .eslintrc.js:
      1. ***@vue/typescript/recommended*** entfernen
      2. ***@typescript-eslint/no-explicit-any: 'off',*** *entfernen*
   4. In **router/index.js:**
      1. **Array<RouteRecordRaw>** entfernen
   5. **shims-vue.d.ts** entfernen
   6. In jedem ***script*** Tag, ***lang=ts*** entfernen. ( z.B. ***App.vue*** && ***views/Home.vue*** )
5. ***Home.vue*** in views/ löschen und eigene pages erstellen
   1. In jeder Page <template> und <script> export default{} </script>
   2. In **src/router/index.js**:
      1. Page als Component importieren:  
         **import PageName from 'PathToPage';**
      2. path: '/**RouteName',**component: ***PageName***
6. In der Page:
   1. Basic Components importieren:  
      Components auf "ionicframework.com/docs/components" finden  
      t***import { IonPage, IonHeader, IonToolbar, IonTitle, IonContent } from @ionic/vue*OPTIONAL FÜR LISTE: *ion-list, ion-item***
   2. Side Note: Components immer locally importieren um BundleSize gering zu halten!!!
   3. In ***export default {}*** Components unter Key registrieren:  
      ***components:{  
      IonPage,  
      IonHeader,  
      etc.  
      }***
   4. Components im kebab-case schreiben:  
      eg. ***ion-page***
   5. In Page:
      1. ion-page tag
      2. ion-header und ion-content
      3. In ion-header: ion-toolbar
      4. In ion-toolbar: ion-title
7. STYLING:
   1. Normales CSS Styling möglich.
   2. In **theme/variable.css:**
      1. Farben anpassen
   3. ODER: Eigene Datei in **themes/** erstellen und in **main.js** importieren
      1. CSS Variablen auf "ionicframework.com/docs/api/\*Element\*" finden
      2. eg in **ion-toolbar{} --background:** ändern  
         z.B. mit var(\*Variable aus **variable.css**\*)
      3. ACHTUNG:   
         Auf iOS funktioniert eine normale Variable mit der ToolBar nicht!  
         Dafür aus **root:** '--ion-toolbar-color: **var(--ion-color-primary-contrast)** nutzen.
      4. Nun kann man in der **variable.css** die Farben ändern, ohne diese in jeder einzelnen CSS zu ändern.
      5. SIDE NOTE:  
         Theming Tab angucken!  
         Theming Generator auf "ionicframework.com/docs/theming/color-generator" nutzen!
8. Beispiel für Clickable Stuff, in diesem Fall eine List mit Items:
   1. Zweite Page zur Weiterleitung erstellen UND Layout Datei für Dynamic Content erstellen
      1. Um Wiederholung von z.B. imports zu vermeiden:
         1. **src/components** Ordner erstellen
         2. in **src/components/** Layout Datei erstellen ( eg BaseLayout.vue )
         3. in Layout Datei, Base Layou festlegen ( Also Components importieren und registrieren )
         4. Entweder in Layout Datei, den Inhalt der Tags ( ion-content, ion-title, etc. mit **<slot />** ersetzen um später mit **composition** die Inhalte dynamisch auszufüllen.
         5. Oder Interpolation nutzen:   
            Inhalt des Tags mit {{ **\*NameOfContent\*** }}  
            Prop in **export default** erstellen:  
            **props: ['\*NameOfContent\*'],**
         6. Als Global Component in **main.js** importieren, wenn das Layout oft / immer genutzt wird :  
            **import \*NameOfLayoutFile\* from \*PathToLayoutFile\*;**Component registrieren und benennen: **app.component('\*name-for-component\*', ' \*NameOfLayoutFile\*');**
      2. In Page:  
         Tags mit **<\*name-of-component\* ( z.B. ggf. mit prop: pageTitle:"Page Title">** ersetzen  
         Imports und Registrations von Components die mit der LayoutFile importiert werden löschen um Duplikate zu vermeiden!!!   
           
         Im Layout Component kann dann der Inhalt des **<slot />** gefüllt werden. ( z.B. **<base-layout pageTitle:"Page Title"> <h2> Page Content </base-layout>** )
      3. In **router/index.js** Route erstellen mit**:**
         1. **path: 'RouteName/:id',**
         2. Component kann mit LazyLoading geladen werden, wenn Seite nicht dringend oft besucht wird:  
            **component: () => import('\*PathToPage')**
      4. In erster Page in diesem Fall in der List für Items:
         1. In **ion-item** die prop **router-link="RouteName/\*DynamicRouteToSet\*"** setzen
         2. Um einen BackButton zu erstellen:
            1. **IonBackButton** in Layout Dateiv importieren und registrieren.
            2. In **ion-toolbar, ion-back-button** setzen
            3. SIDE NOTE: Title und BackButton sind jetzt im selben Slot  
               **IonButtons** importieren und registrieren  
               **IonButton** in **IonButtons** wrappen und prop **slot="z.B. start" setzen.**
         3. SIDE NOTE:  
            Wenn in Ionic eine Route über einen Element geroutet wird, wird Info übergeben, dass dort der BackButton gesetzt werden soll.  
            Wird diese Route ( z.B. /RouteName/1 ) refreshed, ist diese Info verloren.  
            **Zum Vermeiden:**prop ':**default-href="pageDefaultBackLink"'** in back-button setzen und in **export defaults** gleichnamig als prop erstellen!  
            In Layout File: oben erstellte prop einsetzen und route ( z.B. **/RouteName** ) setzen.
         4. **LETZTER TIME STAMP IM VIDEO: 48:26**
9. Mit ***ionic serve*** den DevServer auf localhost starten