Création d’une application mobile avec ionic

# Installation et création du projet

npm install -g @ionic/cli

cd (vers le worspace)

ionic start

Ne pas utiliser le wizard (sauf si vous souhaitez vous créer un compte ionic)

cd demoIonic

## Pour ouvrir le projet dans vscode

code .

## Pour démarrer le projet dans un navigateur :

ionic serve

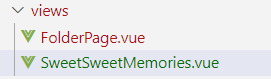
Par défaut ionic utilise TypeScript

Pour l’enlever il faut suivre les étapes suivantes :

<https://ionicframework.com/docs/vue/quickstart#build-your-way-with-typescript-or-javascript>

# Création d’une page (view)

## Créer un fichier view dans le dossier view







Toutes les vues vont à peut prêt avoir la même structure de base.

IonPage est un composant représentant la page

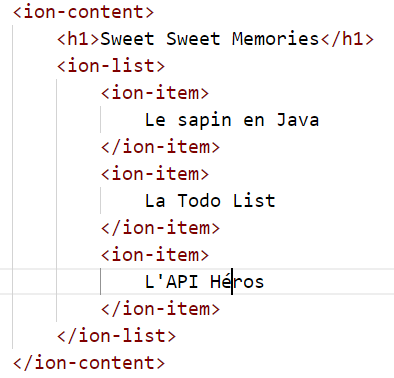
IonHeader est un header qui peut comprendre un titre et une toolbar

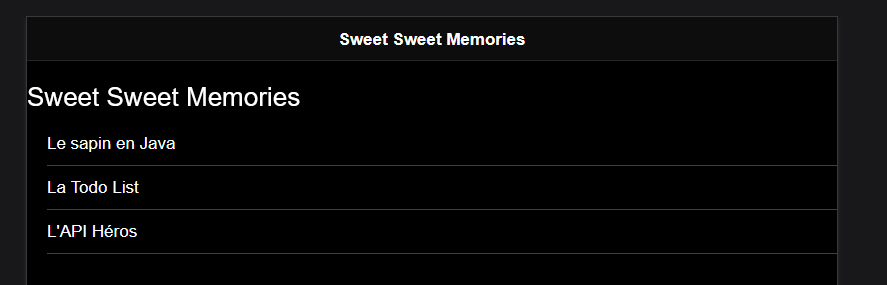
Voir <https://ionicframework.com/docs/api/header>

Cela peut être utilise de créer un snippet (que l’on peut améliorer par la suite)

Pour cette page on va commencer par faire une liste, on a donc besoin d’importer IonList et IonItem

Une fois importer et renseigner en tant qu’attribut de components :





Il faut aussi mettre a jour le router>index.js

const routes= [

  {

    path: '',

    redirect: '/memories'

  },

  {

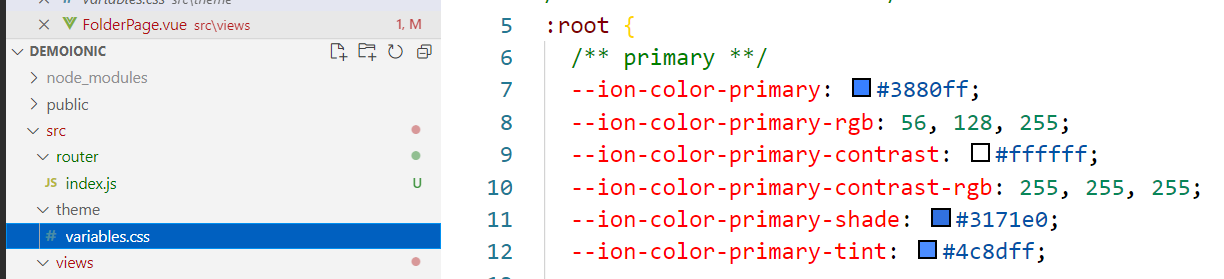
    path: '/memories',

    component: () => import ('../views/SweetSweetMemories.vue')

  }

]

# Mettre en style



Le fichier variables.css contient une multitude de variable CSS.

On peut en forme des composant soit au niveau local soit au niveau global.

## Au niveau local :

Dans la vue SweetSweetMemories on peut modifier la couleur des items en utilisant par exemple une variable CSS. Seulement les items de cette vue sont affectés.

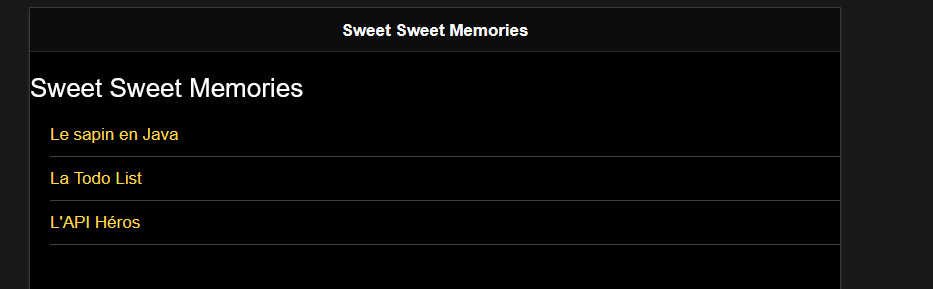
<style>

    ion-item{

        --color: var(--ion-color-warning-tint);

    }

</style>



## Au niveau global :

Ajouter un fichier style.css dans le dossier theme.

ion-toolbar{

    --background: var(--ion-color-secondary);

}

Cela affecte toutes les toolbars de toutes les pages.

Les propriétés CSS de chaque composant ionic sont indiquer dans le document par exemple ici :

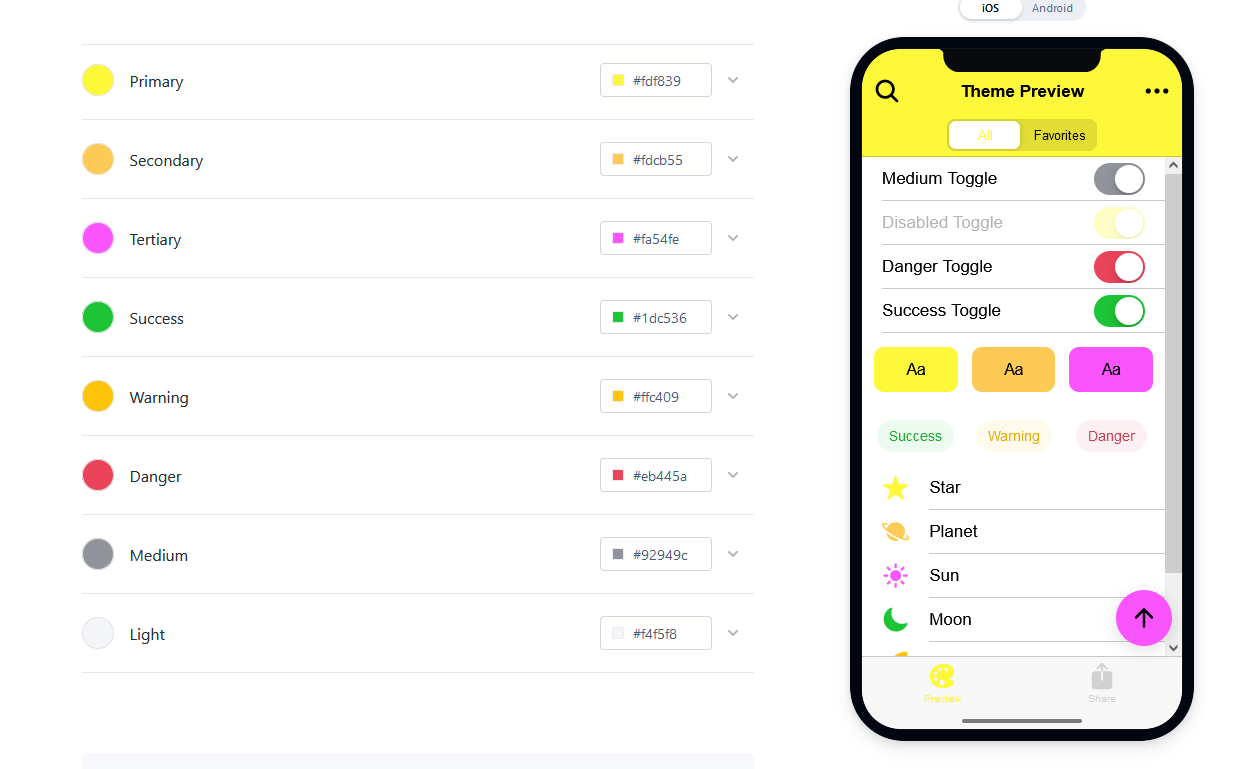
<https://ionicframework.com/docs/api/toolbar#css-custom-properties>

Même si les propriétés CSS traditionnelle fonction (en tout cas tant que on est sur une Web App) il est fortement conseiller d’utiliser les propriétés ionic afin d’être sûr de la compatibilité native.

## Ionic Color Generator

<https://ionicframework.com/docs/theming/color-generator>

Grace a ce lien vous pouvez générer votre thème.



Cela vous fournit en bas les variables CSS, qu’il faut mettre dans votre.

# Templating

Malgré le snippet on veut éviter de dupliquer du code.

Pour éviter la duplication de code on peut créer un compose qui nous servir de template pour nos pages.

Créer un dossier components > base

Dans le dossier base créer un fichier TemplateLayout.vue

<template>

    <ion-page>

        <ion-header>

            <ion-toolbar>

                <!-- Permet de placer des boutons sur la toolbar -->

                <ion-buttons slot="start">

                    <!-- Revenir a la page precedente -->

                    <ion-back-button></ion-back-button>

                </ion-buttons>

                <ion-title>{{ titre }}</ion-title>

            </ion-toolbar>

        </ion-header>

        <ion-content>

            <slot />

        </ion-content>

    </ion-page>

</template>

<script>

import { IonPage, IonHeader, IonToolbar, IonContent, IonTitle,IonButtons,IonBackButton } from '@ionic/vue'

export default {

    props: ["titre"],

    components: {

        IonPage,

        IonHeader,

        IonContent,

        IonToolbar,

        IonTitle,

        IonBackButton,

        IonButtons,

    }

}

</script>

<style>

</style>

## Utiliser notre composant TemplateLayout

Le composant a un slot pour le contenu et une interpolation pour le titre de la page.

Si on veut que notre vue SweetSweetMemories utilise se composant alors :

<template>

<TemplateLayout :titre="'Sweet Sweet Memorie'">

            <h1>Sweet Sweet Memories</h1>

            <ion-list>

                <ion-item router-link="/memories/1">

                    Le sapin en Java

                </ion-item>

                <ion-item>

                    La Todo List

                </ion-item>

                <ion-item>

                    L'API Héros

                </ion-item>

            </ion-list>

    </TemplateLayout>

</template>

<script>

import { IonList,IonItem } from '@ionic/vue'

import TemplateLayout from '../components/base/TemplateLayout.vue'

export default {

components:{

    IonList,

    IonItem,

    TemplateLayout

}

}

</script>

<style>

    ion-item{

        --color: var(--ion-color-primary-contrast);

    }

</style>

On note que j’ai ajouter un lien sur le premier ion-item

# Exercice 1 : Créer une page (view) qui permet d’afficher le détail d’un souvenir (pour l’instant juste du texte statique)

Et mettre votre router à jour :

  {

    path: '/memories/:id',

    component:()=> import ('../views/MemoryDetails.vue')

  }

Dans SweetSweetMemories ajouter un data :

data:function(){

    return{

        memories:[

            {id:1,titre:"Le sapin en Java", description:"Nos débuts en Java",image:"sapin.jpg"},

            {id:2,titre:"La Todo List", description:"Les servlets",image:null},

            {id:3,titre:"Les ampoules", description:"Initiation a Spring et l'injection de dépendance",image:null},

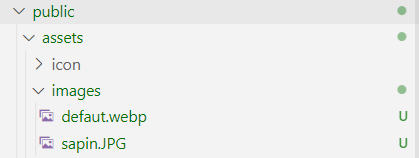
            {id:4,titre:"L'API Héros", description:"Démonstration d'une API Rest et creation de projets Vue.js",image:null},

        ]

    }

}

Est ajouter quelque image :



# Exercice 2 : Générer une boucle pour parcourir chaque souvenir chaque les ion-item auront un thumbnail un label.

 <ion-thumbnail slot="start">

                    <ion-img :src="`/assets/images/defaut.webp `"></ion-img>

  </ion-thumbnail>

   <ion-label>TITRE</ion-label>

# Correction :

<template>

<TemplateLayout :titre="'Sweet Sweet Memorie'">

            <h1>Sweet Sweet Memories</h1>

            <ion-list>

                <ion-item v-for="memory in memories" v-bind:key="memory.id" :router-link="`/memories/${memory.id}`">

                <ion-thumbnail slot="start">

                    <ion-img :src="`/assets/images/${memory.image ?? 'defaut.webp'} `"></ion-img>

                 </ion-thumbnail>

                 <ion-label>{{memory.titre}}</ion-label>

                </ion-item>

            </ion-list>

    </TemplateLayout>

</template>

# Instalation de Pinia

npm install pinia

Dans main.js ajouter :

import { IonicVue } from '@ionic/vue';

et

const pinia = createPinia()

const app = createApp(App)

  .use(IonicVue)

  .use(router)

  .use(pinia);

Créer un dossier stores et un fichier memories.js

import { defineStore } from 'pinia'

export const memoryStore = defineStore('memoryStore', {

  state: () => {

    return { memories:[

        {id:1,titre:"Le sapin en Java", description:"Nos débuts en Java",image:"sapin.jpg"},

        {id:2,titre:"La Todo List", description:"Les servlets",image:null},

        {id:3,titre:"Les ampoules", description:"Initiation a Spring et l'injection de dépendance",image:null},

        {id:4,titre:"L'API Héros", description:"Démonstration d'une API Rest et creation de projets Vue.js",image:null},

    ] }

  },

  getters:{

    memory(state){

        return (memoryId)=>{

            return state.memories.find((unSouvenir)=>unSouvenir.id==memoryId)

        }

    }

  }

})

Puis modifier Sweet Sweet memories

<script>

import { IonList,IonItem,IonThumbnail,IonLabel,IonImg } from '@ionic/vue'

import TemplateLayout from '../components/base/TemplateLayout.vue'

import {memoryStore} from '@/stores/memories.js'

export default {

    setup(){

        const store = memoryStore();

        return {memories:store.memories};

    },

components:{

    IonList,

    IonItem,

    IonThumbnail,

    IonLabel,

    IonImg,

    TemplateLayout,

},

}

</script>

Pour MemoryDatails

<template>

  <template-layout :titre="souvenirCourant.titre ?? 'Souvenir'">

  <div v-if="souvenirCourant">

    <h1>{{souvenirCourant.titre}}</h1>

    <ion-text color="primary">

        {{souvenirCourant.description}}

    </ion-text>

    </div>

    <div v-else>

        <h1>Souvenir introuvable</h1>

    </div>

  </template-layout>

</template>

<script>

import TemplateLayout from '@/components/base/TemplateLayout.vue'

import {memoryStore} from '@/stores/memories.js'

export default {

      setup(){

        const store = memoryStore();

        return {memories:store};

    },

  components: { TemplateLayout },

  data(){

    return{

    memoryId:null

    }

  },

    watch:{

        //Observe les changement quand id est modifier dans l'url

        '$route.params.id':{

          handler: function(id) {

           this.memoryId=id

        },

        deep: true,

        immediate: true

        }

    },

    computed:{

        //Fait appelle au store pour recuperer le souvenir en fonction memoryId (cad l'id dans l'url)

        souvenirCourant(){

            console.log(this.memoryId)

            return this.memories.memory(this.memoryId) ?? false

        }

    }

}

</script>

<style>

</style>

# Utilisation de Capacitor

Tant que on n’utilise pas Capacitor notre application ionic est une application web qui resemble a une application mobile.

<https://capacitorjs.com/docs/getting-started/environment-setup#android-sdk>

<https://capacitorjs.com/docs/getting-started/with-ionic>