Temas personalizados en Ionic 2

Ionic 2 proporciona la funcionalidad de temas (theming functionality) que nos permite crear temas personalizados para las interfaces del usuario(UI), incluyendo botones, fondos, iconos, alertas y muchos más con algunos cambios en las hojas SASS (Syntactically Awesome Style Sheets).

Dicho esto, se puede crear varios temas y, a continuación, permitir que el usuario cambie el tema a través de las opciones / configuración de la aplicación o establecer el tema específico por plataforma al realizar la comprobación de plataforma. Con él, se puede imitar la apariencia nativa de interfaz de usuario de las plataformas con diferentes temas y configuraciones.

Cuando se crea una nueva página, se obtiene el archivo component (***home.ts***), su archivo de vista HTML (***home.html***) y su archivo de estilo (***home.scss***). Se debe tener en cuenta que Ionic 2 no utiliza CSS para el estilo, sino que utiliza SCSS (Sassy CSS), que es una especie de extensión de CSS, lo que significa que se añaden algunas nuevas características, como variables, anidación, mixins, etc.

Básicamente, SCSS nos permite hacer el estilo más fácilmente en comparación con CSS, porque la sintaxis soporta más características, y por lo tanto reduce la complejidad en el lenguaje. Cuando se construye la aplicación, los archivos SCSS finalmente se compilan en CSS, porque los navegadores sólo entienden el CSS. Básicamente SCSS es una extensión de CSS, por lo que cualquier código CSS válido es válido en SCSS.

**Integración de diferentes temas en una aplicación**

* **Creación de un proyecto**

Comenzamos creando un nuevo proyecto utilizando el siguiente comando:

$ionic new ionic-theme blank –v2

Podemos definir las variables globalmente o realizar cambios en los archivos SCSS específicos de la página. Podemos modificar las variables en "***theme / variable.scss***" que se aplicarán a la configuración global, son valores predeterminados que se pueden modificar. El "***app / app.scss***" se utiliza para importar los diferentes temas, y los cambios en este archivo se aplica a todos los componentes y no sólo al componente de la aplicación, estas configuraciones sobrescriben el "***variable.scss***". Por ejemplo, si definimos "$ instagram-button-size: 100px;" en "variable.scss", pero definimos "$ instagram-button-size: 350px;" en "app.scss", entonces la segunda configuración será la que se aplicara.

Vamos a crear una variable en SCSS, de la siguiente manera:

$facebook-width: 40px;

Básicamente, utilizamos el signo $ para declarar variables. Se puede definir tantas variables como se desee y siempre que se lo necesite. Dicho esto vamos a crear una variable como la siguiente:

page-home {

.facebook-button {

width: $facebook-width;

}

}

La ventaja de esto es que nos permite aplicar las variables a tantos elementos que uno quiera, y si cambias el valor de una variable, ¡entonces se aplica a cada elemento que usa la variable!

Si se observa el código anterior, "***.facebook-button***" está anidado en "***page-home***". Esto no era posible en CSS simple. Se llama anidamiento y se puede anidar tantos elementos como se desee.

* **Comencemos a realizar un poco de estilo!**

Abramos "***theme / variables.scss***" y procederemos a cambiar el mapa ***"$ colors"*** a lo siguiente:

$colors: (

primary: #387ef5,

secondary: #32db64,

danger: #f53d3d,

light: #f4f4f4,

dark: #222,

twitter: #55acee,

facebook: #3b5998,

tumblr: #35465c,

instagram: #3f729b

);

El array "***$*** ***colors***" se utiliza para cambiar globalmente los valores de color. Por defecto los componentes de ionic usan **primary, secondary, danger,** etc, así que si se cambia los valores a otra cosa todos esos elementos / componentes serán cambiados en su color. Se puede agregar tantas variables personalizadas como se desee para los colores.

Ahora, vamos a agregar la línea siguiente:

$instagram-button-size: 100px;

Vamos a añadir estos dos muy pronto. Ahora, abramos "***app / app.scss***" y agreguemos lo siguiente en cualquier parte del archivo:

$instagram-button-size: 350px;

*He cambiado su valor para demostrar que "****app.scss****" anula las variables en "****variables.scss****", y se aplica a todos los componentes.*

A continuación abrimos "***home / home.scss***" y colocamos el siguiente contenido:

**home.scss**

page-home {

.instagram-button {

width: $instagram-button-size;

}

}

Como se puede observar hemos aplicado el valor al elemento. Ahora abrimos el "***home.html***" y copiamos el siguiente contenido (estoy pegando todo el contenido):

**home.html**

<ion-header>

<ion-navbar>

<button ion-button menuToggle>

<ion-icon name="menu"></ion-icon>

</button>

<ion-title>Home</ion-title>

</ion-navbar>

</ion-header>

<ion-content padding>

<br>

<button ion-button color="twitter">Twitter Button</button>

<button ion-button color="facebook">Facebook Button</button>

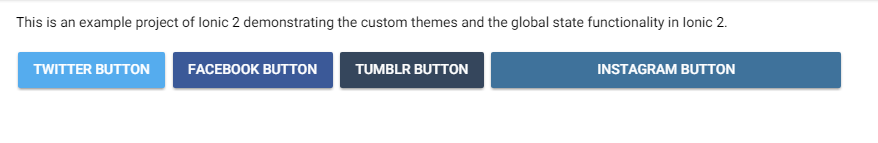
<button ion-button color="tumblr">Tumblr Button</button>

<button class="instagram-button" ion-button color="instagram">Instagram Button</button>

</ion-content>

Solo nos resta ver los cambios y para ello debemos escribir el siguiente comando:

$ionic serve

El resultado debería verse como el siguiente: 

* **Es hora de hacer nuestros propios temas!**

Vamos a crear tres temas: **light**, **dark** e **intense**. Pero, primero crearemos una condición global que almacenará las variables globales porque necesitamos cambiar los temas dinámicamente. En "***src / app***" crearemos un nuevo archivo llamándolo "***app.global.ts***" y vamos a copiar el siguiente contenido:

**app.global.ts**

import { Injectable } from '@angular/core';

@Injectable()

export class AppState {

\_state = {}

constructor() {

}

get state() {

return this.\_state = this.\_clone(this.\_state);

}

set state(value) {

throw new Error('do not mutate the `.state` directly');

}

get(prop?: any) {

const state = this.state;

return state.hasOwnProperty(prop) ? state[prop] : state;

}

set(prop: string, value: any) {

return this.\_state[prop] = value;

}

\_clone(object) {

return JSON.parse(JSON.stringify(object));

}

}

Esta clase es para almacenar las variables globales. No nos olvidemos de añadir este archivo al array "***providers***" de "***app.module.ts***".

Ahora, vamos a crear una nueva página para cambiar temas y para ello utilizaremos el siguiente comando:

$ionic g page theming

Abrimos el "***theming.ts***" y copiamos el siguiente contenido:

**theming.ts**

import { Component } from '@angular/core';

import { NavController, NavParams } from 'ionic-angular';

import { AppState } from '../../app/app.global';

@Component({

selector: 'page-theming',

templateUrl: 'theming.html'

})

export class ThemingPage {

ionViewDidLoad() {

console.log('ionViewDidLoad ThemingPage');

}

constructor(public navCtrl: NavController, public navParams: NavParams, public global: AppState) {

}

changeTheme(theme) {

this.global.set('theme', theme);

}

}

Estamos utilizando el método **changeTheme ()** para cambiar nuestro estado global. Este cambio de estilo se verá reflejado en el archivo de forma global. Posteriormente aplicaremos el tema a nuestras páginas enteras.

Antes de editar el "***theming.html***", vamos a añadir primero los temas. En la carpeta "***src / theme /***", creamos un nuevo archivo "***light.theme.scss***" y copiamos el siguiente contenido:

**light.theme.scss**

.theme-light {

transition: all 1.1s ease;

background-color: #fff;

// background: #0c1a2c;

// background: linear-gradient(to bottom,rgba(9,16,25,1) 0,rgba(30,28,58,1) 25%,rgba(30,28,58,1) 75%,rgba(11,12,22,1) 100%);

h1,

h2,

h3,

h4,

h5,

h6,

p,

.toolbar-title,

.bar-button-default,

body,

div {

transition: all 1.1s ease;

color: white;

}

.onboard {

border-color: #212121 **!important**;

}

.toolbar-background,

.toggle-checked {

transition: all 1.1s ease;

background-color: #212121 **!important**;

}

.item-note,

.themeButton,

.swiper-pagination-bullet {

transition: all 1.1s ease;

//background-color: #fff;

//border-style: solid;

//border-width: 1px;

//border-color: white;

}

ion-content,

.input-wrapper,

.item-inner,

.item,

ion-footer,

ion-item,

ion-label {

transition: all 1.1s ease;

background-color: #303030 **!important**;

background: #303030 **!important**;

color: black **!important**;

}

ion-list {

background-color: #100815;

}

.time {

background-color: white;

}

.item {

transition: all 1s ease;

background-color: white;

color: black;

}

.range-bar {

background-color: #463E3F **!important**;

}

.s2,

.s3,

.toggle-inner,

.range-pin,

.range-pin::before,

.range-knob,

.range-bar-active {

background-color: white **!important**;

color: black **!important**;

}

}

En el mismo directorio, creamos un nuevo archivo "***dark.theme.scss***" y copiamos el siguiente contenido:

**dark.theme.scss**

.theme-dark {

transition: all 1.1s ease;

background-color: #303030;

// background: #0c1a2c;

// background: linear-gradient(to bottom,rgba(9,16,25,1) 0,rgba(30,28,58,1) 25%,rgba(30,28,58,1) 75%,rgba(11,12,22,1) 100%);

h1,

h2,

h3,

h4,

h5,

h6,

p,

.toolbar-title,

.bar-button-default,

body,

div {

transition: all 1.1s ease;

color: white;

}

.onboard {

border-color: #212121 **!important**;

}

.toolbar-background,

.toggle-checked {

transition: all 1.1s ease;

background-color: #212121 **!important**;

}

.item-note,

.themeButton,

.swiper-pagination-bullet {

transition: all 1.1s ease;

//background-color: #fff;

//border-style: solid;

//border-width: 1px;

//border-color: white;

}

ion-content,

.input-wrapper,

.item-inner,

.item,

ion-footer,

ion-item,

ion-label {

transition: all 1.1s ease;

background-color: #303030 **!important**;

background: #303030 **!important**;

color: white **!important**;

}

ion-list {

background-color: #100815;

}

.time {

background-color: white;

}

.item {

transition: all 1s ease;

background-color: #100815;

color: white;

}

.range-bar {

background-color: #463E3F **!important**;

}

.s2,

.s3,

.toggle-inner,

.range-pin,

.range-pin::before,

.range-knob,

.range-bar-active {

background-color: white **!important**;

color: black **!important**;

}

}

*La diferencia en este tema y el tema anterior son algunos cambios en el código hexadecimal de color y la sustitución de "blanco" por "negro" y no mucho más en realidad.*

En "***app.scss***", vamos a importar los temas creados:

@import '../theme/dark.theme';

@import '../theme/light.theme';

Ahora, abra el "***theming.html***" y copie el siguiente contenido:

**theming.html**

<ion-header>

<ion-navbar>

<button ion-button menuToggle>

<ion-icon name="menu"></ion-icon>

</button>

<ion-title>Theming</ion-title>

</ion-navbar>

</ion-header>

<ion-content padding>

<h3>Change the theme</h3>

<button ion-button (click)="changeTheme('')">Default</button>

<button ion-button color="dark" (click)="changeTheme('theme-dark')">Dark</button>

</ion-content>

Como se puede observar hemos creado dos botones aquí. El segundo botón tiene el color "***dark***". Asegúrese de que esta página esté incluida en "***app.component.ts***" y "***app.module.ts***".

Si realizamos un "*ionic serve*" ahora, notaremos que los botones no hacen nada porque no hemos definido un código para cambiar el tema. En ***"app.html"***, necesitamos agregar un elemento **<div>** a todos los elementos / cosas, y la clase se establecerá como ***"global.state ['theme']".*** Vayamos al archivo y copiemos el código siguiente por encima del código que exista en el archivo:

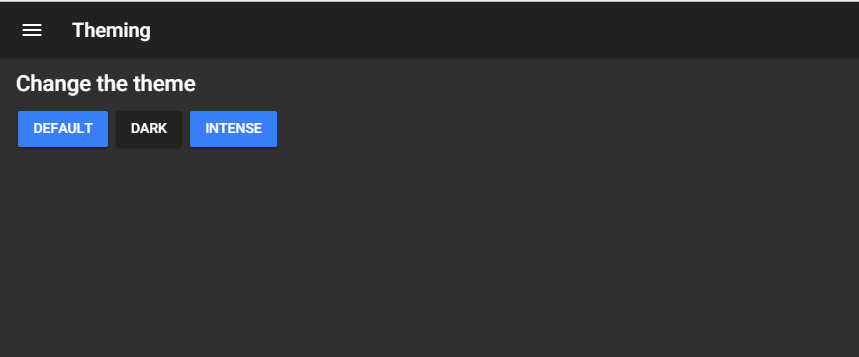
**app.html**

<div class="{{global.state['theme']}}">

*No olvides de cerrar el* ***</div>*** *al finalizar el código del archivo.*

Esto aplicará el tema a todos los elementos y componentes siempre que haya un cambio de estado el cual estamos activando a través de botones de alternancia en nuestra página "***theming***".

Ahora hagamos "***ionic serve***" para ver los cambios!

Deberíamos obtener un resultado como el siguiente:

Con este método, podemos crear tantos temas como queramos. Todo lo que necesitamos hacer es crear un nuevo archivo de tema en la carpeta "***src / theme***" e importar el tema en "***app.scss***".

Vamos a crear un tema más "***intense.theme.scss***". Copie el siguiente contenido:

**intense.theme.scss**

.theme-intense {

transition: all 1.1s ease;

background-color: #c00000;

// background: #0c1a2c;

// background: linear-gradient(to bottom,rgba(9,16,25,1) 0,rgba(30,28,58,1) 25%,rgba(30,28,58,1) 75%,rgba(11,12,22,1) 100%);

h1,

h2,

h3,

h4,

h5,

h6,

p,

.toolbar-title,

.bar-button-default,

body,

div {

transition: all 1.1s ease;

color: black;

}

.onboard {

border-color: #ffff00 **!important**;

}

.toolbar-background,

.toggle-checked {

transition: all 1.1s ease;

background-color: #ffff00 **!important**;

}

.item-note,

.themeButton,

.swiper-pagination-bullet {

transition: all 1.1s ease;

//background-color: #fff;

//border-style: solid;

//border-width: 1px;

//border-color: white;

}

ion-content,

.input-wrapper,

.item-inner,

.item,

ion-footer,

ion-item,

ion-label {

transition: all 1.1s ease;

background-color: #c00000 **!important**;

background: #c00000 **!important**;

color: black **!important**;

}

ion-list {

background-color: #100815;

}

.time {

background-color: black;

}

.item {

transition: all 1s ease;

background-color: 100815;

color: black;

}

.range-bar {

background-color: #463E3F **!important**;

}

.s2,

.s3,

.toggle-inner,

.range-pin,

.range-pin::before,

.range-knob,

.range-bar-active {

background-color: black **!important**;

color: black **!important**;

}

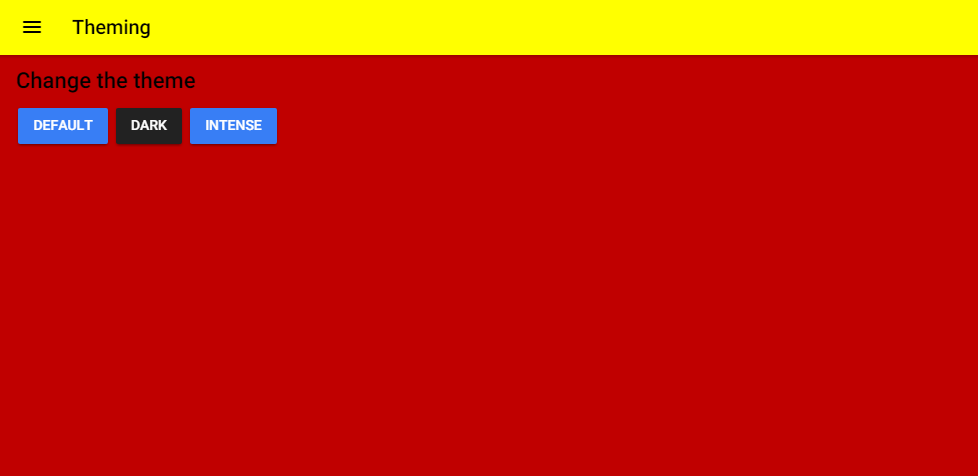
}

Importamos el tema en "***app.scss***":

@import '../theme/intense.theme';

Incluimos un botón correspondiente en "***theming.html***":

<button ion-button (click)="changeTheme('theme-intense')">Intense</button>

Y ahora hacemos "ionic serve" para ver el resultado, el cual debería ser el siguiente:

Como pueden observar no es realmente difícil de crear temas personalizados en Ionic 2. Es como ajustar y cambiar un poco de texto y copiar / pegar el código de aquí y allá. Así que si realmente disfrutas realizando temas y estilos podrás destacar en la fase de diseño de desarrollo de aplicaciones en Ionic 2.