Search 100s of mobile development tutorials...

ALL BEGINNER INTERMEDIATE ADVANCED CAPACITOR UI UX PERFORMANCE TESTING

IONIC IONIC/ANGULAR IONIC/VUE STENCIL PWA NESTJS PHASER VIDEOS

ANGULAR TUTORIALS | IONIC TUTORIALS

Creating a Theme Switcher Service in Ionic Using CSS4 Variables

BY JOSH MORONY | LAST UPDATED: JULY 26, 2018

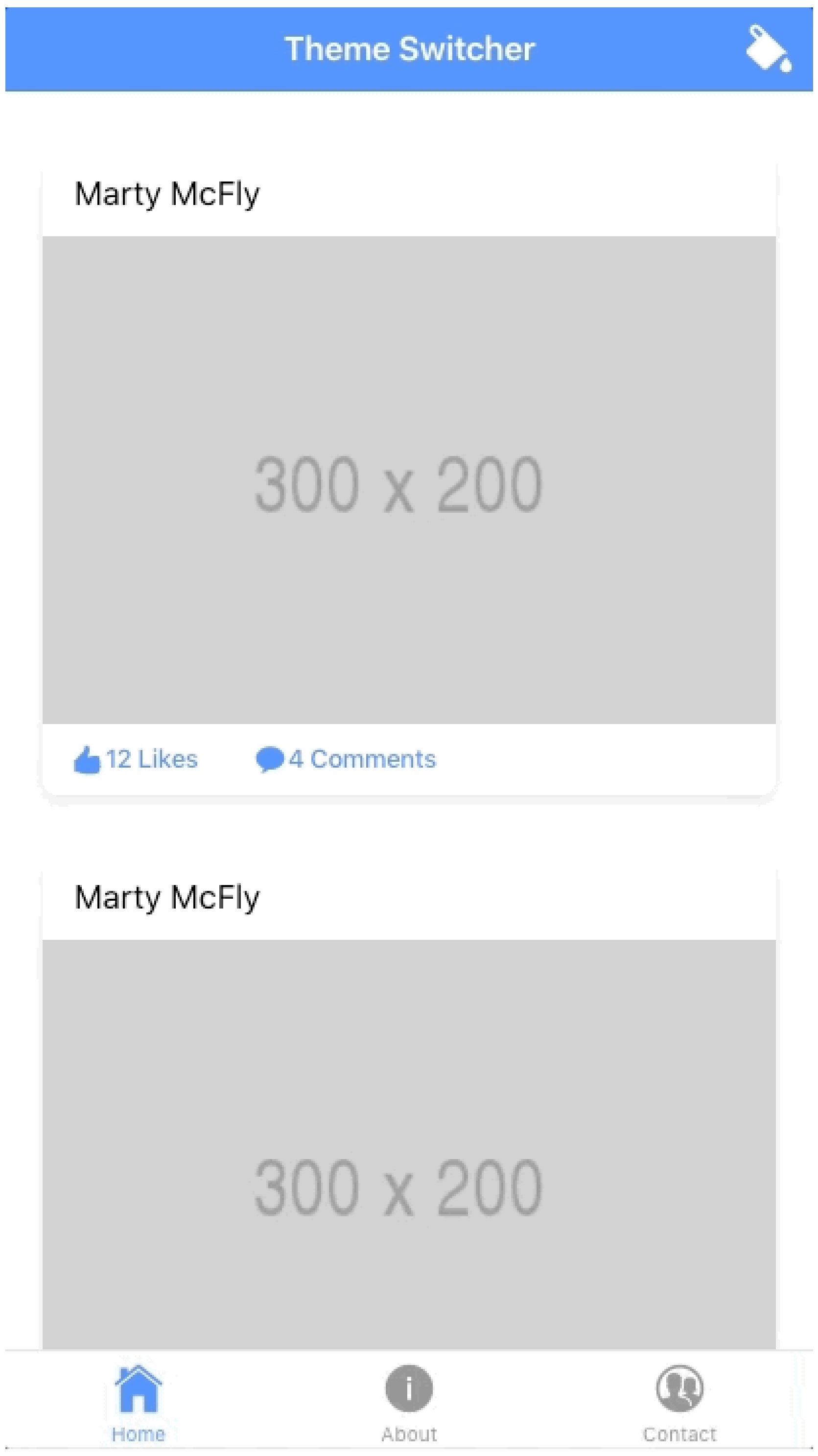
ANGULAR INTERMEDIATE IONIC UI UX

Follow **Josh Morony** on

A couple of weeks ago I released an article about how CSS4 variables would be used in Ionic 4. In that article, I discussed how using CSS4 variables for theming allows you to modify these variables at runtime because unlike SASS variables which need to be compiled, CSS4 variables are supported natively by the browser. This means that you can pull up the debugging window and mess around with the variables to your heart's content, and it also means you can change the values of the variables on the fly with Javascript.

This makes it easy to make sweeping changes to the style of your application whilst it is running, and at the click of a button you could completely change the entire theme of your application instantly. In this tutorial, we are going to walk through implementing a theme switcher service in an Ionic 4 application that will allow you to define multiple themes that the user could switch between.

At the end of this tutorial, you will have a service that can apply theme changes to your application whilst it is running, like this:



1. Create the Theme Switcher Service

If you have already read this post, and I would recommend that you do, then you would know that we can overwrite Ionic's default CSS4 variables with our own variables, e.g.:

```
:root {
    --ion-color-primary: red;
    --ion-background-color: #f4f4f4;
}
```

This is easy to do with CSS because all we need to do is drop it into the **variables.scss** file. If we want to change these variables dynamically at runtime, then we will need to modify them with Javascript. We can do this by modifying the `document` object and setting the appropriate variable, e.g:

```
document.documentElement.style.setProperty('--ion-color-primary', 'red');
```

In order to do this in a more manageable way, we are going to create a `ThemeSwitcherService` that handles modifying these variables. We would then be able to inject that service into any page in our application to access its functionality.

> *Run the following command to create a* `ThemeSwitcherService`*:*

```
ionic g service ThemeSwitcher
```

Now we are going to set up the basic code for the theme switcher service. We will just focus on the basic implementation first, and then we will add some themes to it.

> *Modify* ***app/services/theme-switcher.service.ts*** *to reflect the following:*

```
import { Injectable, Inject } from '@angular/core';
import { DOCUMENT } from '@angular/common';
import { DomController } from '@ionic/angular';

interface Theme {
  name: string;
  styles: ThemeStyle[];
}

interface ThemeStyle {
  themeVariable: string;
  value: string;
}

@Injectable({
  providedIn: 'root'
})
export class ThemeSwitcherService {

  private themes: Theme[] = [];
  private currentTheme: number = 0;

  constructor(private domCtrl: DomController, @Inject(DOCUMENT) private docume

  }

  cycleTheme(): void {

    if(this.themes.length > this.currentTheme + 1){
      this.currentTheme++;
    } else {
      this.currentTheme = 0;
    }

    this.setTheme(this.themes[this.currentTheme].name);

  }

  setTheme(name): void {

    let theme = this.themes.find(theme => theme.name === name);

    this.domCtrl.write(() => {

      theme.styles.forEach(style => {
        document.documentElement.style.setProperty(style.themeVariable, style.
      });

    });

  }

}
```

One thing you might notice right away is the strange use of `@Injec` `DOCUMENT` – this is just an "Angular way" for us to get a reference to `document` object. The way we define themes is going to have a spe

Join my mailing list and I'll start sending you some curated Ionic/Angular tutorials.
Click to learn more

to it, so we have also set up some interfaces to define types for them. Themes will be an object with a name that contains an array of additional objects to specify theme variables. The theme objects will look something like this:

```
{
        name: 'day',
        styles: [
          { themeVariable: '--ion-color-primary', value: '#f8383a'},
          { themeVariable: '--ion-background-color', value: '#f94c4e'}
        ]
      }
```

We have defined two methods in this service. The `setTheme` method will allow us to supply it with a theme name, and then it will find that theme and loop through all of its theme variables, setting each one on the `document` object. We are also using the `DomController` from Ionic here – it is a good idea to use this service whenever you are writing to the DOM, as it will ensure that the writes occur efficiently which helps avoid performance problems.

Although the `setTheme` method can be called directly, the `cycleTheme` method will allow us to cycle through the various themes sequentially.

2. Define the Themes

Now that we have the basic structure of the service set up, we can add some themes to it.

> *Modify **app/services/theme-switcher.service.ts** to reflect the following:*

```
import { Injectable, Inject } from '@angular/core';
import { DOCUMENT } from '@angular/common';
import { DomController } from '@ionic/angular';

interface Theme {
  name: string;
  styles: ThemeStyle[];
}

interface ThemeStyle {
  themeVariable: string;
  value: string;
}

@Injectable({
  providedIn: 'root'
})
export class ThemeSwitcherService {

  private themes: Theme[] = [];
  private currentTheme: number = 0;

  constructor(private domCtrl: DomController, @Inject(DOCUMENT) private docume

    this.themes = [
      {
        name: 'day',
        styles: [
          { themeVariable: '--ion-color-primary', value: '#f8383a'},
          { themeVariable: '--ion-color-primary-rgb', value: '248,56,58'},
          { themeVariable: '--ion-color-primary-contrast', value: '#ffffff'},
          { themeVariable: '--ion-color-primary-contrast-rgb', value: '255,255
          { themeVariable: '--ion-color-primary-shade', value: '#da3133'},
          { themeVariable: '--ion-color-primary-tint', value: '#f94c4e'},
          { themeVariable: '--ion-item-ios-background-color', value: '#ffffff'
          { themeVariable: '--ion-item-md-background-color', value: '#ffffff'}
          { themeVariable: '--ion-tabbar-background-color', value: '#fff'},
          { themeVariable: '--ion-tabbar-ios-text-color-active', value: '#0000
          { themeVariable: '--ion-tabbar-md-text-color-active', value: '#00000
          { themeVariable: '--ion-background-color', value: '#f94c4e'}
        ]
      },
      {
        name: 'night',
        styles: [
          { themeVariable: '--ion-color-primary', value: '#222428'},
          { themeVariable: '--ion-color-primary-rgb', value: '34,34,34'},
          { themeVariable: '--ion-color-primary-contrast', value: '#ffffff'},
          { themeVariable: '--ion-color-primary-contrast-rgb', value: '255,255
          { themeVariable: '--ion-color-primary-shade', value: '#1e2023'},
          { themeVariable: '--ion-color-primary-tint', value: '#383a3e'},
          { themeVariable: '--ion-item-ios-background-color', value: '#717171'
          { themeVariable: '--ion-item-md-background-color', value: '#717171'}
          { themeVariable: '--ion-tabbar-background-color', value: '#222428'},
          { themeVariable: '--ion-tabbar-ios-text-color-active', value: '#ffff
          { themeVariable: '--ion-tabbar-md-text-color-active', value: '#fffff
          { themeVariable: '--ion-background-color', value: '#383838'}
        ]
      }
    ]

  }
  cycleTheme(): void {

    if(this.themes.length > this.currentTheme + 1){
      this.currentTheme++;
    } else {
      this.currentTheme = 0;
    }

    this.setTheme(this.themes[this.currentTheme].name);

  }

  setTheme(name): void {

    let theme = this.themes.find(theme => theme.name === name);

    this.domCtrl.write(() => {

      theme.styles.forEach(style => {
        document.documentElement.style.setProperty(style.themeVariable, style.
      });

    });

  }

}
```

In the code above, you can see that we have defined two themes: a `day` theme, and a `night` theme. We are changing the variables for the `primary` colours, the item background colour, the tab colour, text colour, and background colour. However, you can changes as few or as many variables as you like (keep in mind that if you change a variable for one theme, you will need to change it back when switching to another theme).

It can be a little tedious setting up *all* of these variables, but the Ionic team will be releasing a tool to help generate themes automatically (e.g. it will calculate the `contrast`, `shade`, and `tint` values automatically for you). If you change the primary colour without changing the contrast and other values, you could end up with some less than desirable combinations.

3. Trigger the Theme Change

All that is left to do now is trigger the theme change. This is as simple as injecting the service in a page:

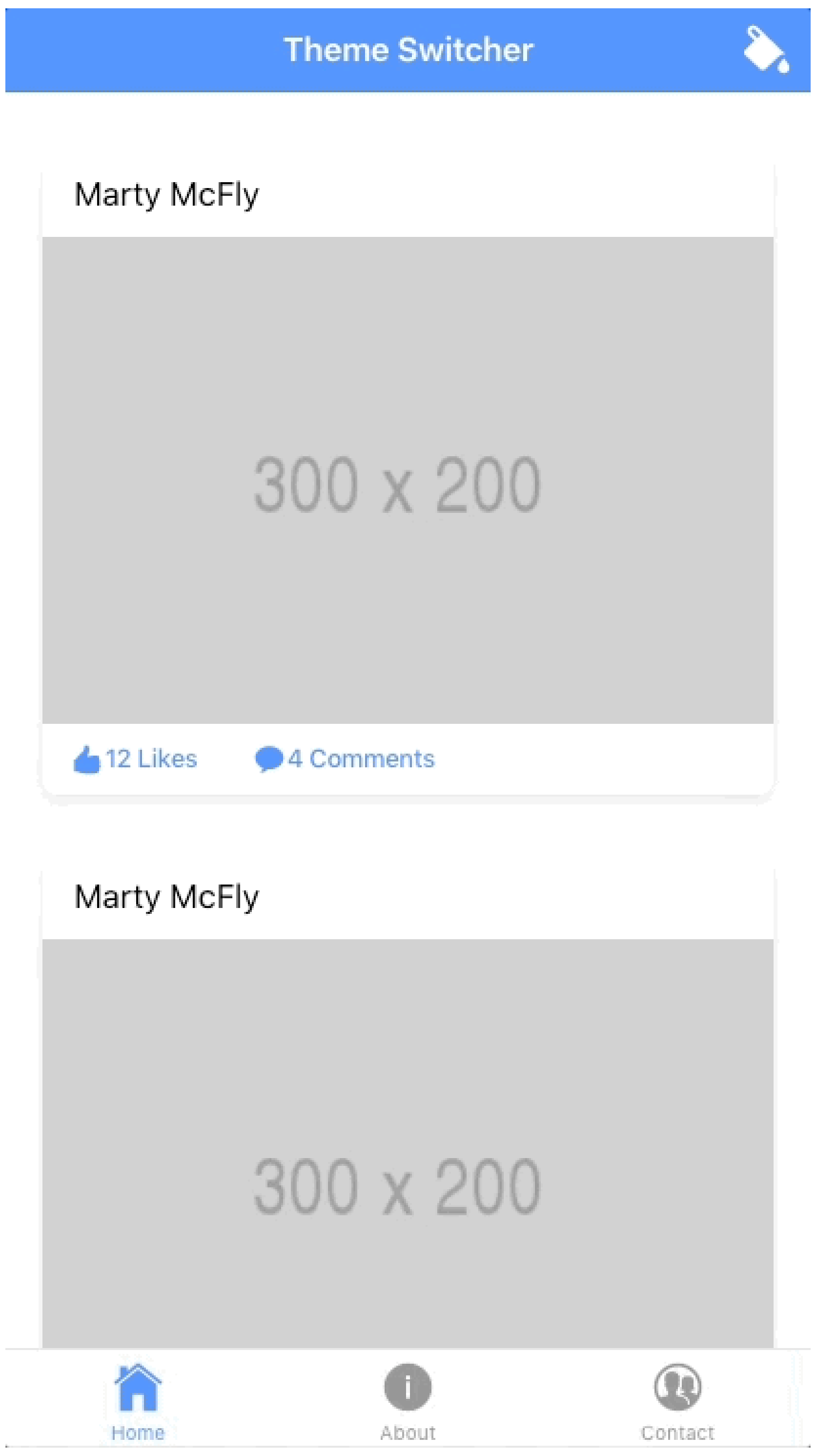
```
constructor(public themeSwitcher: ThemeSwitcherService){

  }
```

and then calling one of the two methods:

```
this.themeSwitcher.setTheme('day');
```

```
this.themeSwitcher.cycleTheme();
```

As soon as you trigger either of these methods, the theme will change instantly:



Summary

Without CSS4 variables creating functionality like this would have been a much bigger task, and now with the help of this service, it is quite simple to completely change the style of your application. It is not all that common to change the entire theme of an application (although, day/night theme switchers are reasonably common) but you could use this same concept to make changes to your application at runtime.

In this example, we are just manually defining themes, but if you wanted to you could even improve this service by allowing it to pull in either a local or external JSON file that defines the themes.

← Create a PWA with Angular Service Workers in Ionic 4

Creating a WebVR Experience in an Ionic/Angular Application →

Check out my latest videos:

2 Comments joshmorony.com 1 Login

Recommend Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Sergio Eric • 5 months ago
Awesome, now, how can we use the new dark mode in ionic with angular?
The documentation show an example with just JavaScript listening with addEventListener dom event
Reply • Share ›

Javier Moreno Valle • 10 months ago
After searching and searching, testing and testing ... this form is the only one that works.
Tested on Ionic 4

Thank you for your work
Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

PRIVACY POLICY TERMS OF SERVICE CONTACT

Join my mailing list and I'll start sending you some curated Ionic/Angular tutorials.
Click to learn more