Módulo 1 - Lectura: Archivos personalizados por plataforma

Directorio platforms

El directorio de plataformas (*platforms*) se crea cuando inicia una compilación o agrega una plataforma de destino a su proyecto. Las herramientas de Nativescript crean un nuevo subdirectorio con el nombre de la plataforma correspondiente. Estos subdirectorios tienen la estructura de proyecto específica de la plataforma requerida para el desarrollo nativo con los SDK nativos de la plataforma. Cuando el proyecto está preparado para la compilación, las herramientas de Nativescript copian el contenido relevante del directorio de la aplicación al subdirectorio específico de la plataforma para cada plataforma de destino.

> IMPORTANTE: evita editar los archivos ubicados en los subdirectorios de las plataformas, ya que la CLI de Nativescript los anula con el contenido del directorio de la aplicación durante el proceso de preparación de la plataforma. Es decir que es una carpeta auto-administrada por Nativescript.

Directorio App_Resources

La carpeta App_Resources contiene los recursos específicos de la plataforma de la aplicación (iconos, archivos de configuración, etc.).
Los archivos de configuración que respetan las herramientas de Nativescript son:

1. App_Resources/Android/src/main/AndroidManifest.xml para Android
2. App_Resources/iOS/Info.plist para iOS.

Además, puede modificar App_Resources/iOS/build.xcconfig o App_Resources/Android/app.gradle para agregar o eliminar propiedades de compilación adicionales para las plataformas iOS y Android, respectivamente. Ambos son archivos de configuración de las utilidades de compilación usadas por XCode para iOS y Android Studio para Android.

Archivos CSS separados

Puedes tener archivos CSS completamente separados, utilizando la misma sintaxis de nombres de archivos presentada anteriormente para Code Sharing. Sin embargo, en este caso, para diferenciar entre iOS y Android, por ejemplo:

> myapp.ios.css
> - y -
> myapp.android.css

Sin embargo, es más probable que tengas un solo archivo CSS (globalmente o para una vista específica) con estilos comunes, y dos archivos CSS separados, específicos de la plataforma. Entonces, ¿cómo combinamos tres hojas de estilo en una?

Simplemente, usas el comando @import en su archivo CSS principal. Por ejemplo, nuestro archivo myapp.css podría verse así:

```
@import { url('~/myapp.ios.css') }
@import { url('~/myapp.android.css') }

.button {
  border-color: #b2b2b2;
  background-color: #f8f8f8;
  ...
```

Los estilos específicos de iOS sólo se importarán en iOS y los estilos específicos de Android sólo se importarán en Android.

Un archivo CSS para múltiples plataformas

Si no deseas mantener hojas de estilo separadas, puedes crear diferentes opciones de estilo para cada plataforma dentro de un archivo CSS. Esto se logra más fácilmente usando el complemento nativescript-platform-css de Nathanael Anderson.

Ese complemento agrega clases de plataforma a su hoja de estilo, de modo que puede escribir lo siguiente:

```
<Page>
  <Label class="color " text="Hola Mundo!">
</Page>

.ios .color { color: blue; }
.android .color { color: green; }
.windows .color { color: red; }
```

En este ejemplo, en iOS, la clase .color proporcionará texto de color azul. Los demás serán ignorados. Puedes encontrar más información en: https://github.com/NathanaelA/nativescript-platform-css

Nosotros, en el presente curso, utilizamos principalmente el desarrollo en emulador con Android, por lo que no usaremos esta funcionalidad, pero es vital conocerla para sacar todo el provecho posible a la herramienta Nativescript.

Bloques de Código

Es posible que solo necesites separar bloques de código JavaScript/Typescript para asegurarte de que estás exponiendo la funcionalidad específica de la plataforma, de la manera correcta.
Por ejemplo, si deseass establecer el color del texto de la barra de encabezado en su dispositivo iOS, no desea ejecutar ese código en Android:

```
if (page.ios) {
    var navigationBar = frameModule.topmost().ios.controller.navigationBar;
    navigationBar.barStyle = UIBarStyle.UIBarStyleBlack;
```

}

Si no es obvio, la clave aquí es page.ios (y page.android), son variables que simplemente devuelven un booleano para decirle en qué plataforma se está ejecutando la aplicación.

También podemos usar, como dijimos anteriormente, el código Typescript, por ejemplo:

MiComponente.android.ts y MiComponente.ios.ts

Pero lo que suele hacerse es evitar esto y utilizar un solo archivo de código para el Typescript de nuestro componente.