Flutter IconTheme and IconButton defaults

SUMMARY

IconButton's new Material 3 defaults must coexist with ThemeData.iconTheme and IconTheme.

Author: Hans Muller (HansMuller)
Created: 7/2022 / **Last updated:** 7/2022

WHAT PROBLEM IS THIS SOLVING?

The discussion that follows was prompted by:
Added IconButtonTheme and apply it to IconButton in M3 #108332

IconButton's defaults need to be updated for Material 3. Existing apps may use ThemeData.icon or IconTheme to override the default appearance of IconButtons and we do not want to break that. ThemeData.iconTheme has a non-null default values for dark and light themes which makes it difficult to introduce the new Material 3 IconButton defaults.

BACKGROUND

Flutter's IconTheme class is part of the widgets library and is used to "[define] the default color, opacity, and size of icons in a widget subtree". IconTheme is one of the oldest Widgets in the framework, it was created in October 2015.

OVERVIEW

Flutter's IconTheme class is part of the widgets library and is used to "[define] the default color, opacity, and size of icons in a widget subtree". IconTheme is one of the oldest Widgets in the framework, it was created in October 2015.

The Theme's iconTheme property is used to define a default IconTheme for the entire application:

```
iconTheme ??= isDark
  ? const IconThemeData(color: Colors.white)
  : const IconThemeData(color: Colors.black87);
```

This value is used to create an IconTheme that typically contains the entire app.

Currently the API documentation for IconTheme consists of two sentences. The first is noted above and was just changed to be less specific - #106896. The second sentence is misleading.

"The icon theme is honored by Icon and ImageIcon widgets."

This is true for the widgets library. The material library also has IconTheme dependencies.

AppBar

The implementation of AppBar's iconTheme and actionsIconTheme uses an IconTheme to change the appearance of the corresponding Icon and IconButton AppBar descendants.

The AppBar changes in #108332 shouldn't be needed if the IconButton changes outlined below are implemented.

BackButton

BackButton's implementation creates an IconButton and its color property explicitly depends on IconTheme.

```
return IconButton(
  icon: const BackButtonIcon(),
  color: color,
  ...
```

Even if we make the IconButton changes outlined below, BackButton's implementation should probably stay as it is.

CloseButton

CloseButton is nearly identical to BackButton.

ChipAttributes - all of the Chip widgets

As of Flutter 3.3 (#107166) of the chip widgets extend ChipAttributes which includes an IconThemeData valued iconTheme property.

If we make the IconButton changes, the Chip implementation can probably remain as it is.

IconButton

Depends on the IconTheme for the icon button's size. More on IconButton below.

PopupMenuButton

Creates an IconButton and passes along the IconTheme's size. Presumably the _kDefaultIconSize clause below has no effect, since IconTheme.of() will always return an IconThemeData with a non-null size.

```
final IconThemeData iconTheme = IconTheme.of(context);
…
return IconButton(
  icon: widget.icon ?? Icon(Icons.adaptive.more),
  iconSize: widget.iconSize ?? iconTheme.size ?? _kDefaultIconSize,
  …
);
```

If we make the IconButton changes described below, we should be able to change the code above to:

```
return IconButton(
  icon: widget.icon ?? Icon(Icons.adaptive.more),
  iconSize: widget.iconSize,
  …
);
```

FlutterLogo

The IconTheme defines the size of the logo.

Even if we make the IconButton changes outlined below, FlutterLogo's implementation should probably stay as it is.

What to do about IconButton

IconButton's default size and the color of its Icon descendant are currently defined by the IconTheme. We'd like them to be defined by the new IconButtonTheme instead, if an IconButtonTheme is defined with non-null properties. Otherwise we'd like them to be defined by the defaults for Material 3 rather than the current IconTheme. At least not by the current IconTheme, *if* the current IconTheme is the default one automatically defined by Theme.

We could check for that case:

```
IconThemeData overallIconTheme = IconTheme.of(context);
IconThemeData? iconTheme = isThemeIconTheme(overallIconTheme)
  ? null
  : overallIconTheme;

bool isThemeIconTheme(IconThemeData theme) {
   // Return true if theme is the same as the default
   // value of Theme.of(context).iconTheme
}
```

So in this case `iconTheme` would only be used to define the icon button's default size and color if it was non-null, i.e. if it wasn't the one defined by the overall Theme. That means that if the app has inserted its own IconTheme into the widget tree, we'd use that instead of the M3 defaults. Just to clarify, here's how we'd resolve each IconButton property, in order of precedence:

1. Widget properties like IconButton.color
2. IconButtonTheme properties
3. IconTheme properties unless they were defined by Theme.of(context).iconTheme
4. Material 3 default IconButtonTheme properties

As always, the first non-null value is the one that's used. As noted in the code sample, the exception for IconTheme properties is for ones defined by the *default* value of ThemeData.iconTheme.