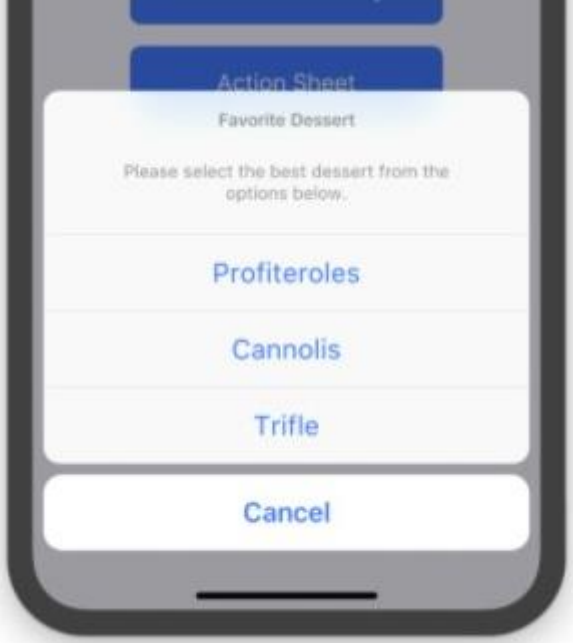
Cupertino widgets

iOS-style

CupertinoActionSheet



Constructors

CupertinoActionSheet({Key key, Widget title, Widget message, List<Widget> actions, ScrollController messageScrollController, ScrollController actionScrollController, Widget cancelButton})

Properties
actions → List<Widget> //as CupertionActionSheetAction
The set of actions that are displayed for the user to select. [...]

actionScrollController → ScrollController
A scroll controller that can be used to control the scrolling of the actions in the action sheet. [...]

cancelButton → Widget
The optional cancel button that is grouped separately from the other actions. [...]

CupertinoActivityIndicator

Constructors

CupertinoActivityIndicator({Key key, bool animating: true, double radius: _kDefaultIndicatorRadius, @Deprecated('Leave this field default to use latest style. ' 'This feature was deprecated after v1.21.0-1.0.pre.') CupertinoActivityIndicatorIOSVersionStyle iOSVersionStyle: CupertinoActivityIndicatorIOSVersionStyle.iOS14})

Creates an iOS-style activity indicator that spins clockwise.

CupertinoActivityIndicator.partiallyRevealed({Key key, double radius: _kDefaultIndicatorRadius, double progress: 1.0, @Deprecated('Leave this field default to use latest style. ' 'This feature was deprecated after v1.21.0-1.0.pre.') CupertinoActivityIndicatorIOSVersionStyle iOSVersionStyle: CupertinoActivityIndicatorIOSVersionStyle.iOS14})

Creates a non-animated iOS-style activity indicator that displays a partial count of ticks based on the value of progress. [...]

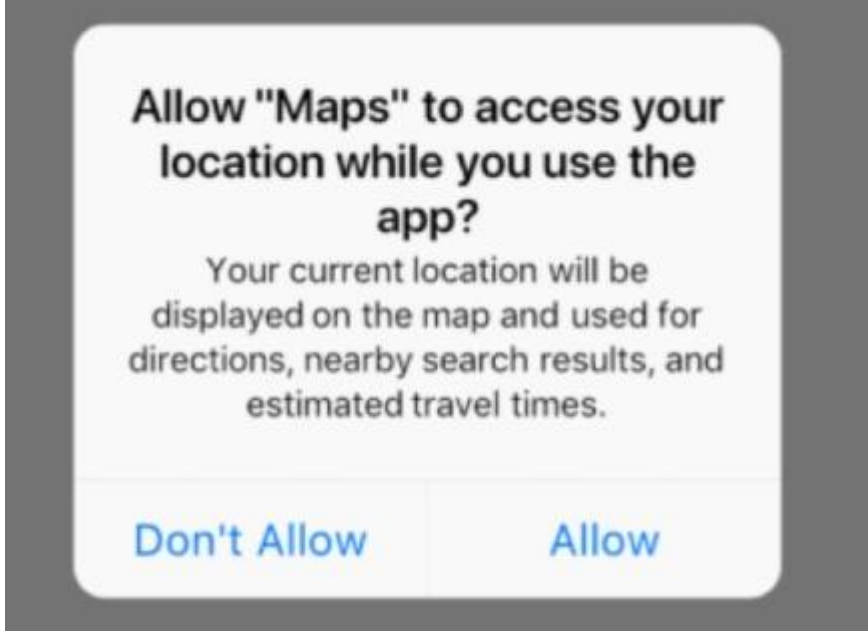
Properties
animating → bool
Whether the activity indicator is running its animation. [...]

hashCode → int
The hash code for this object. [...]
@nonVirtual, read-only, inherited
iOSVersionStyle → CupertinoActivityIndicatorIOSVersionStyle
The iOS version style of activity indicator. [...]

progress → double
Determines the percentage of spinner ticks that will be shown. Typical usage would display all ticks, however, this allows for more fine-grained control such as during pull-to-refresh when the drag-down action shows one tick at a time as the user continues to drag down. [...]

radius → double
Radius of the spinner widget. [...]

CupertinoAlertDialog



Constructors

CupertinoAlertDialog({Key key, Widget title, Widget content, List<Widget> actions: const <Widget>[], ScrollController scrollController, ScrollController actionScrollController, Duration insetAnimationDuration: const Duration(milliseconds: 100), Curve insetAnimationCurve: Curves.decelerate})

Creates an iOS-style alert dialog. [...]

Properties

actions → List<Widget>
The (optional) set of actions that are displayed at the bottom of the dialog. [...]

actionScrollController → ScrollController
A scroll controller that can be used to control the scrolling of the actions in the dialog. [...]

content → Widget
The (optional) content of the dialog is displayed in the center of the dialog in a lighter font. [...]

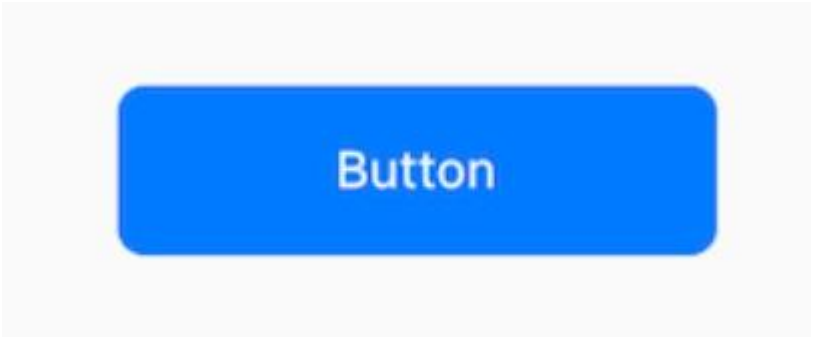
insetAnimationCurve → Curve
The curve to use for the animation shown when the system keyboard intrudes into the space that the dialog is placed in. [...]

insetAnimationDuration → Duration
The duration of the animation to show when the system keyboard intrudes into the space that the dialog is placed in. [...]

scrollController → ScrollController
A scroll controller that can be used to control the scrolling of the content in the dialog. [...]

title → Widget
The (optional) title of the dialog is displayed in a large font at the top of the dialog. [...]

CupertinoButton



Constructors

CupertinoButton({Key key, @required Widget child, EdgeInsetsGeometry padding, Color color, Color disabledColor: CupertinoColors.quaternarySystemFill, double minSize: kMinInteractiveDimensionCupertino, double pressedOpacity: 0.4, BorderRadius borderRadius: const BorderRadius.all(Radius.circular(8.0)), @required VoidCallback onPressed})

Creates an iOS-style button.

CupertinoButton.filled({Key key, @required Widget child, EdgeInsetsGeometry padding, Color disabledColor: CupertinoColors.quaternarySystemFill, double minSize: kMinInteractiveDimensionCupertino, double pressedOpacity: 0.4, BorderRadius borderRadius: const BorderRadius.all(Radius.circular(8.0)), @required VoidCallback onPressed})

Creates an iOS-style button with a filled background. [...]

Properties

borderRadius → BorderRadius
The radius of the button's corners when it has a background color. [...]

child → Widget
The widget below this widget in the tree. [...]

color → Color
The color of the button's background. [...]

disabledColor → Color
The color of the button's background when the button is disabled. [...]

enabled → bool
Whether the button is enabled or disabled. Buttons are disabled by default. To enable a button, set its onPressed property to a non-null value.

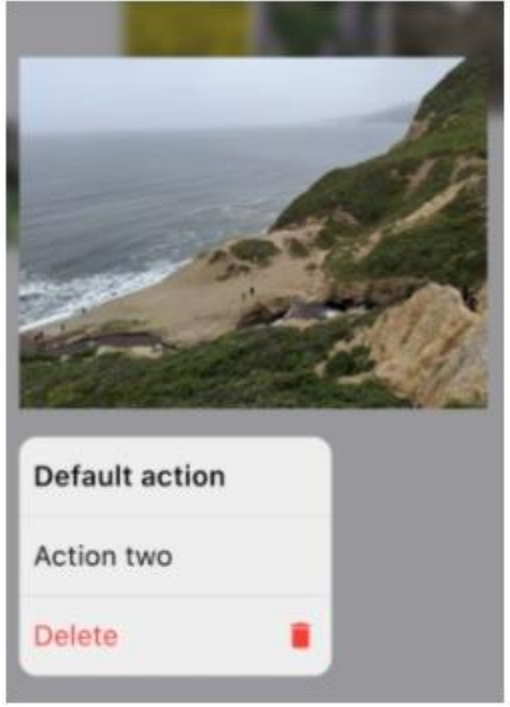
minSize → double
Minimum size of the button. [...]

onPressed → VoidCallback
The callback that is called when the button is tapped or otherwise activated. [...]

padding → EdgeInsetsGeometry
The amount of space to surround the child inside the bounds of the button. [...]

pressedOpacity → double
The opacity that the button will fade to when it is pressed. The button will have an opacity of 1.0 when it is not pressed. [...]

CupertinoContextMenu



Constructors

CupertinoContextMenu({Key key, @required List<Widget> actions, (child, ContextMenuPreviewBuilder previewBuilder})

Create a context menu. [...]

Properties

actions → List<Widget>

The actions that are shown in the menu. [...]

child → Widget

The widget that can be "opened" with the CupertinoContextMenu. [...]

```
Container(
        width: 100,
        height: 100,
        child: CupertinoContextMenu(
          child: Container(
            color: Colors.red,
          ),
          actions: <Widget>[
            CupertinoContextMenuAction(
              child: const Text('Action one'),
              onPressed: () {
                Navigator.pop(context);
              },
            ),
            CupertinoContextMenuAction(
              child: const Text('Action two'),
              onPressed: () {
                Navigator.pop(context);
              },
            ),
          ],
        ),
      ),
```

CupertinoDatePicker

Constructors

CupertinoDatePicker({Key key, CupertinoDatePickerMode
CupertinoDatePickerMode.dateAndTime, @required ValueChanged<DateTime>
onDateTimeChanged, DateTime initialDateTime, DateTime minimumDate, DateTime
maximumDate, int minimumYear: 1, int maximumYear, int minuteInterval: 1, bool
use24hFormat: false, Color backgroundColor})

Constructs an iOS style date picker. [...]

Properties

backgroundColor → Color
Background color of date picker. [...]

initialDateTime → DateTime
The initial date and/or time of the picker. Defaults to the present date and time and must not be null. The present must conform to the intervals set in minimumDate, maximumDate, minimumYear, and maximumYear. [...]

maximumDate → DateTime
The maximum selectable date that the picker can settle on. [...]

maximumYear → int
Maximum year that the picker can be scrolled to in CupertinoDatePickerMode.date mode. Null if there's no limit.

minimumDate → DateTime
The minimum selectable date that the picker can settle on. [...]

minimumYear → int
Minimum year that the picker can be scrolled to in CupertinoDatePickerMode.date mode. Defaults to 1 and must not be null.

minuteInterval → int
The granularity of the minutes spinner, if it is shown in the current mode. Must be an integer factor of 60.

mode → CupertinoDatePickerMode
The mode of the date picker as one of CupertinoDatePickerMode. Defaults to CupertinoDatePickerMode.dateAndTime. Cannot be null and value cannot change after initial build.

onDateTimeChanged → ValueChanged<DateTime>
Callback called when the selected date and/or time changes. If the new selected DateTime is not valid, or is not in the minimumDate through maximumDate range, this callback will not be called. [...]

use24hFormat → bool
Whether to use 24 hour format. Defaults to false.
final

CupertinoDialogAction

Constructors

CupertinoDialogAction({Key key, VoidCallback onPressed, bool isDefaultAction: false, bool isDestructiveAction: false, TextStyle textStyle, @required Widget child})

Creates an action for an iOS-style dialog.

Properties

child → Widget
The widget below this widget in the tree. [...]

enabled → bool
Whether the button is enabled or disabled. Buttons are disabled by default. To enable a button, set its onPressed property to a non-null value.

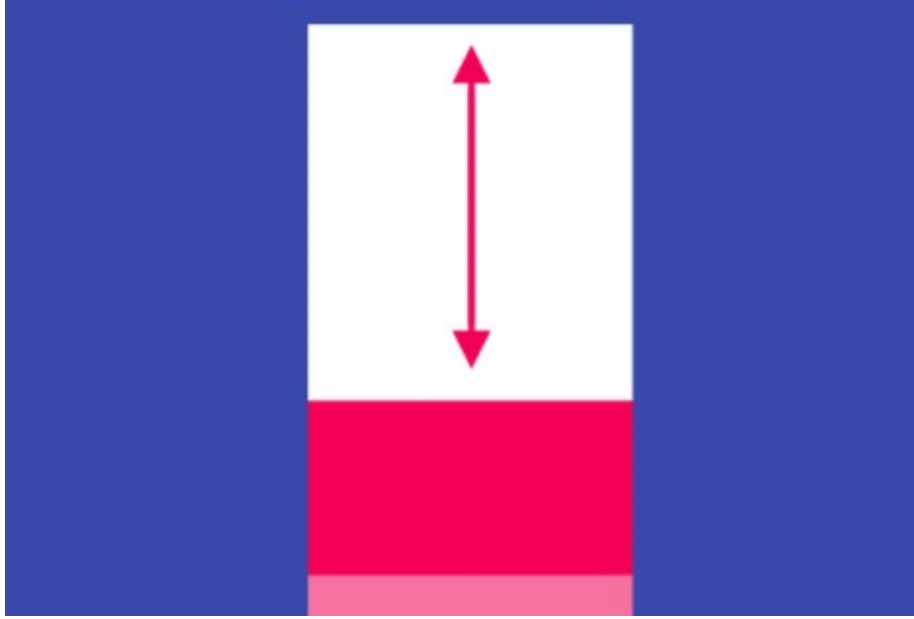
isDefaultAction → bool
Set to true if button is the default choice in the dialog. [...]

isDestructiveAction → bool
Whether this action destroys an object. [...]

onPressed → VoidCallback
The callback that is called when the button is tapped or otherwise activated. [...]

textStyle → TextStyle
TextStyle to apply to any text that appears in this button. [...]
final

CupertinoFullscreen DialogTransition



Constructors

CupertinoFullscreenDialogTransition({Key key, @required Animation<double> primaryRouteAnimation, @required Animation<double> secondaryRouteAnimation, @required Widget child, @required bool linearTransition})

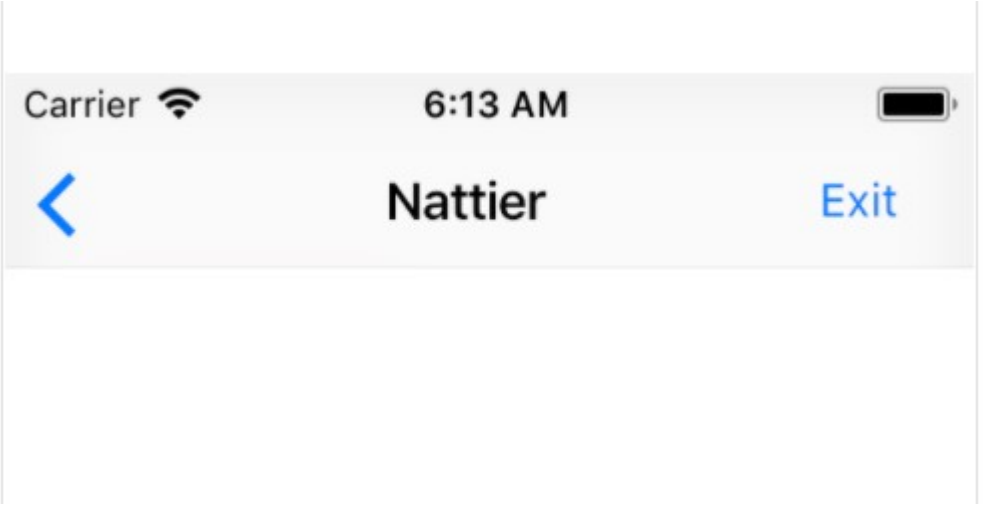
Creates an iOS-style transition used for summoning fullscreen dialogs. [...]

Properties

child → Widget

The widget below this widget in the tree.

CupertinoNavigationBar



Constructors

CupertinoNavigationBar({Key key, Widget leading, bool automaticallyImplyLeading: true, bool automaticallyImplyMiddle: true, String previousPageTitle, Widget middle, Widget trailing, Border border: _kDefaultNavBarBorder, Color backgroundColor, Brightness brightness, EdgeInsetsDirectional padding, Color actionsForegroundColor, bool transitionBetweenRoutes: true, Object heroTag: _defaultHeroTag})

Creates a navigation bar in the iOS style.

Properties

automaticallyImplyLeading → bool
Controls whether we should try to imply the leading widget if null. [...]

automaticallyImplyMiddle → bool
Controls whether we should try to imply the middle widget if null. [...]

backgroundColor → Color
The background color of the navigation bar. If it contains transparency, the tab bar will automatically produce a blurring effect to the content behind it. [...]

border → Border
The border of the navigation bar. By default renders a single pixel bottom border side. [...]

brightness → Brightness
The brightness of the specified backgroundColor. [...]

heroTag → Object
Tag for the navigation bar's Hero widget if transitionBetweenRoutes is true. [...]

function.

transitionBetweenRoutes → bool
Whether to transition between navigation bars. [...]

leading → Widget
Widget to place at the start of the navigation bar. Normally a back button for a normal page or a cancel button for full page dialogs. [...]

middle → Widget
Widget to place in the middle of the navigation bar. Normally a title or a segmented control. [...]

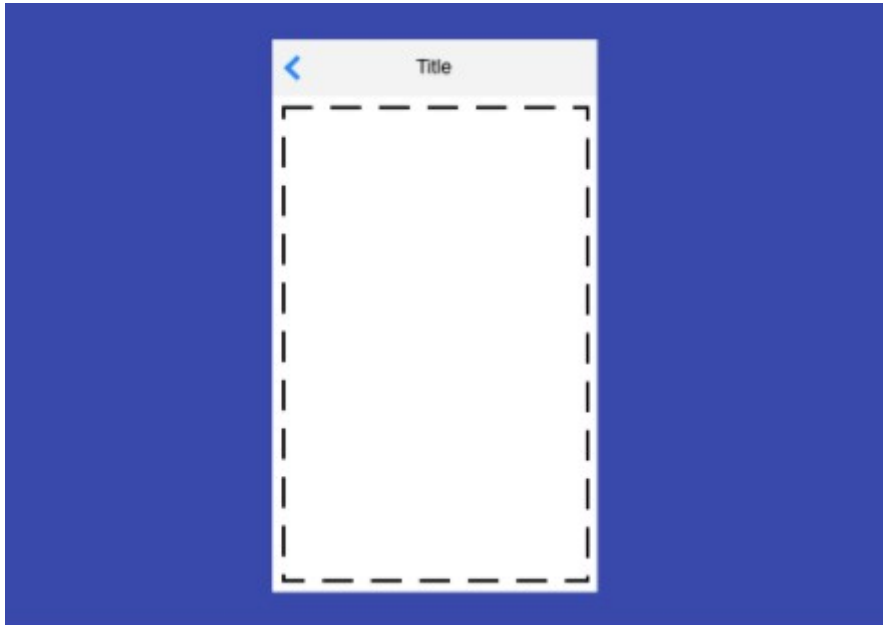
padding → EdgeInsetsDirectional
Padding for the contents of the navigation bar. [...]

preferredSize → Size
The size this widget would prefer if it were otherwise unconstrained. [...]

previousPageTitle → String
Manually specify the previous route's title when automatically implying the leading back button. [...]

trailing → Widget
Widget to place at the end of the navigation bar. Normally additional actions taken on the page such as a search or edit

CupertinoPageScaffold



Constructors

CupertinoPageScaffold({Key key, ObstructingPreferredSizeWidget navigationBar, Color backgroundColor, bool resizeToAvoidBottomInset: true, @required Widget child})

Creates a layout for pages with a navigation bar at the top.

const

Properties

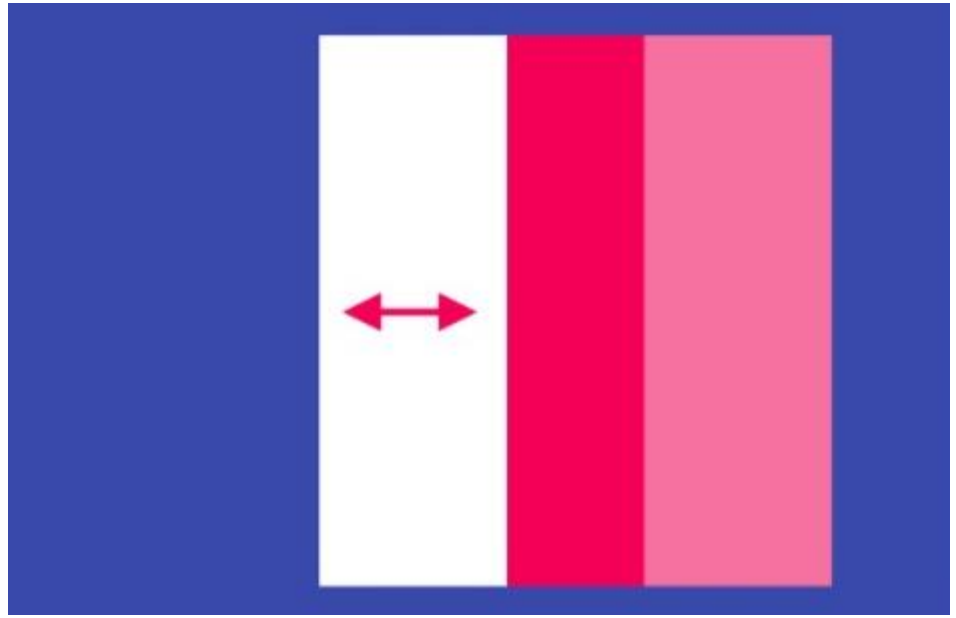
backgroundColor → Color
The color of the widget that underlies the entire scaffold. [...]

child → Widget
Widget to show in the main content area. [...]

navigationBar → ObstructingPreferredSizeWidget
The navigationBar, typically a CupertinoNavigationBar, is drawn at the top of the screen. [...]

resizeToAvoidBottomInset → bool
Whether the child should size itself to avoid the window's bottom inset. [...]

CupertinoPageTransition



Constructors

CupertinoPageTransition({Key key, @required

Animation<double> primaryRouteAnimation,

@required Animation<double> secondaryRouteAnimation, @required Widget child, @required bool linearTransition})

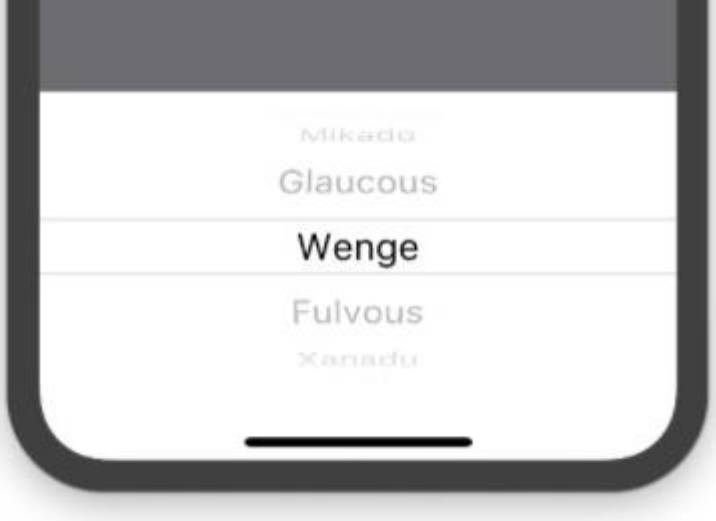
Creates an iOS-style page transition. [...]

Properties

child → Widget

The widget below this widget in the tree.

CupertinoPicker



Constructors

CupertinoPicker({Key key, double diameterRatio: _kDefaultDiameterRatio, Color backgroundColor, double offAxisFraction: 0.0, bool useMagnifier: false, double magnification: 1.0, FixedExtentScrollController scrollController, double squeeze: _kSqueeze, @required double itemExtent, @required ValueChanged<int> onSelectedItemChanged, @required List<Widget> children, bool looping: false})

Creates a picker from a concrete list of children. [...]

CupertinoPicker.builder({Key key, double diameterRatio: _kDefaultDiameterRatio, Color backgroundColor, double offAxisFraction: 0.0, bool useMagnifier: false, double magnification: 1.0, FixedExtentScrollController scrollController, double squeeze: _kSqueeze, @required double itemExtent, @required ValueChanged<int> onSelectedItemChanged, @required IndexedWidgetBuilder itemBuilder, int childCount})

Creates a picker from an IndexedWidgetBuilder callback where the builder is dynamically invoked during layout. [...]

Properties

backgroundColor → Color
Background color behind the children. [...]

childDelegate → ListWheelChildDelegate
A delegate that lazily instantiates children.

diameterRatio → double
Relative ratio between this picker's height and the simulated cylinder's diameter. [...]

itemExtent → double
The uniform height of all children. [...]

magnification → double
The zoomed-in rate of the magnifier, if it is used. [...]

offAxisFraction → double
How much the wheel is horizontally off-center, as a fraction of its width. This property creates the visual effect of looking at a vertical wheel from its side where its vanishing points at the edge curves to one side instead of looking at the wheel head-on. [...]

onSelectedItemChanged → ValueChanged<int>
An option callback when the currently centered item changes. [...]

scrollController → FixedExtentScrollController
A FixedExtentScrollController to read and control the current item, and to set the initial item.
[...]

squeeze → double
The angular compactness of the children on the wheel. [...]

useMagnifier → bool
Whether to use the magnifier for the center item of the wheel.

CupertinoScrollbar

Constructors

CupertinoScrollbar({Key key, ScrollController controller, bool isAlwaysShown: false, double thickness: defaultThickness, double thicknessWhileDragging: defaultThicknessWhileDragging, Radius radius: defaultRadius, Radius radiusWhileDragging: defaultRadiusWhileDragging, @required Widget child})

Creates an iOS style scrollbar that wraps the given child. [...]

Properties

child → Widget
The subtree to place inside the CupertinoScrollbar. [...]

controller → ScrollController
The ScrollController used to implement Scrollbar dragging. [...]

isAlwaysShown → bool
Indicates whether the Scrollbar should always be visible. [...]

radius → Radius
The radius of the scrollbar edges when the scrollbar is not being dragged by the user. [...]

radiusWhileDragging → Radius
The radius of the scrollbar edges when the scrollbar is being dragged by the user. [...]

thickness → double
The thickness of the scrollbar when it's not being dragged by the user. [...]

thicknessWhileDragging → double
The thickness of the scrollbar when it's being dragged by the user. [...]

CupertinoSegmentedControl

Constructors

CupertinoSegmentedControl({Key key, @required Map<T, Widget> children, @required ValueChanged<T> onValueChanged, T groupValue, Color unselectedColor, Color selectedColor, Color borderColor, Color pressedColor, EdgeInsetsGeometry padding})

Creates an iOS-style segmented control bar. [...]

Properties

borderColor → Color
The color used as the border around each widget. [...]

children → Map<T, Widget>
The identifying keys and corresponding widget values in the segmented control. [...]

groupValue → T
The identifier of the widget that is currently selected. [...]

onValueChanged → ValueChanged<T>
The callback that is called when a new option is tapped. [...]

padding → EdgeInsetsGeometry
The CupertinoSegmentedControl will be placed inside this padding. [...]

pressedColor → Color
The color used to fill the background of the widget the user is temporarily interacting with through a long press or drag. [...]

selectedColor → Color
The color used to fill the background of the selected widget and as the text color of unselected widgets. [...]

unselectedColor → Color
The color used to fill the backgrounds of unselected widgets and as the text color of the selected widget. [...]

CupertinoSlider

Constructors

CupertinoSlider({Key key, @required double value, @required ValueChanged<double> onChanged, ValueChanged<double> onChangeStart, ValueChanged<double> onChangeEnd, double min: 0.0, double max: 1.0, int divisions, Color activeColor, Color thumbColor: CupertinoColors.white})

Creates an iOS-style slider. [...]

Properties

activeColor → Color
The color to use for the portion of the slider that has been selected. [...]

divisions → int
The number of discrete divisions. [...]

max → double
The maximum value the user can select. [...]

min → double
The minimum value the user can select. [...]

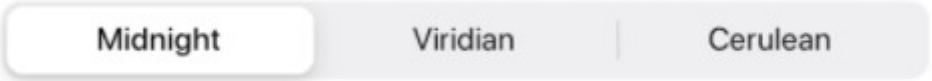
onChanged → ValueChanged<double>
Called when the user selects a new value for the slider. [...]

onChangeEnd → ValueChanged<double>
Called when the user is done selecting a new value for the slider. [...]

onChangeStart → ValueChanged<double>
Called when the user starts selecting a new value for the slider. [...]

thumbColor → Color
The color to use for the thumb of the slider. [...]

value → double
The currently selected value for this slider. [...]



CupertinoSlidingSegment

Constructors

CupertinoSlidingSegmentedControl({Key key, @required Map<T, Widget> children, @required ValueChanged<T> onValueChanged, T groupValue, Color thumbColor: _kThumbColor, EdgeInsetsGeometry padding: _kHorizontalItemPadding, Color backgroundColor: CupertinoColors.tertiarySystemFill})

Creates an iOS-style segmented control bar. [...]

Properties

backgroundColor → Color
The color used to paint the rounded rect behind the children and the separators. [...]

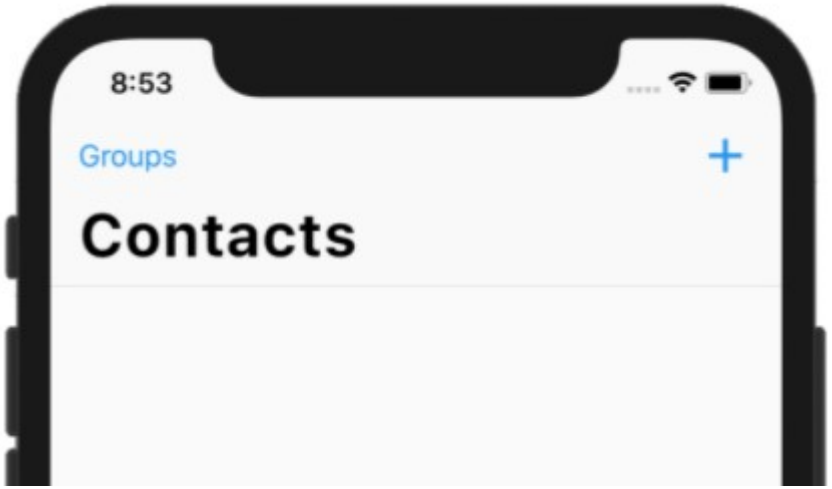
children → Map<T, Widget>
The identifying keys and corresponding widget values in the segmented control. [...]

groupValue → T
The identifier of the widget that is currently selected. [...]

onValueChanged → ValueChanged<T>
The callback that is called when a new option is tapped. [...]
final
padding → EdgeInsetsGeometry
The amount of space by which to inset the children. [...]

thumbColor → Color
The color used to paint the interior of the thumb that appears behind the currently selected item. [...]

CupertinoSliverNavigation



Constructors

CupertinoSliverNavigationBar({Key key, Widget largeTitle, Widget leading, bool automaticallyImplyLeading: true, bool automaticallyImplyTitle: true, String previousPageTitle, Widget middle, Widget trailing, Border border: _kDefaultNavBarBorder, Color backgroundColor, Brightness brightness, EdgeInsetsDirectional padding, Color actionsForegroundColor, bool transitionBetweenRoutes: true, Object heroTag: _defaultHeroTag})

Creates a navigation bar for scrolling lists. [...]

Properties

automaticallyImplyLeading → bool
Controls whether we should try to imply the leading widget if null. [...]

automaticallyImplyTitle → bool
Controls whether we should try to imply the largeTitle widget if null. [...]

backgroundColor → Color
The background color of the navigation bar. If it contains transparency, the tab bar will automatically produce a blurring effect to th

border → Border
The border of the navigation bar. By default renders a single pixel bottom border side. [...]

brightness → Brightness
The brightness of the specified backgroundColor. [...]

heroTag → Object
Tag for the navigation bar's Hero widget if transitionBetweenRoutes is true. [...]

largeTitle → Widget
The navigation bar's title. [...]

leading → Widget
Widget to place at the start of the navigation bar. Normally a back button for a normal page or a cancel button for full page dialogs

middle → Widget
A widget to place in the middle of the static navigation bar instead of the largeTitle. [...]

opaque → bool
True if the navigation bar's background color has no transparency.

padding → EdgeInsetsDirectional
Padding for the contents of the navigation bar. [...]

previousPageTitle → String
Manually specify the previous route's title when automatically implying the leading back button. [...]

trailing → Widget
Widget to place at the end of the navigation bar. Normally additional actions taken on the page such as a search or edit function. [...]

transitionBetweenRoutes → bool
Whether to transition between navigation bars. [...]

CupertinoSwitch

Constructors

CupertinoSwitch({Key key, @required bool value, @required ValueChanged<bool> onChanged, Color activeColor, Color trackColor, DragStartBehavior dragStartBehavior: DragStartBehavior.start})

Creates an iOS-style switch. [...]

Properties

activeColor → Color
The color to use when this switch is on. [...]

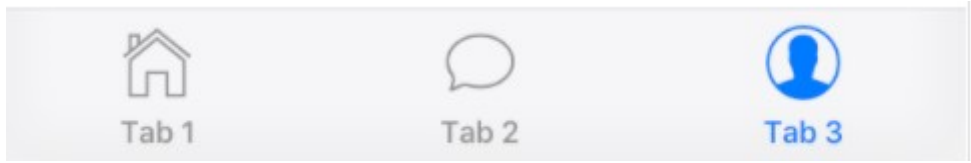
dragStartBehavior → DragStartBehavior
Determines the way that drag start behavior is handled. [...]

onChanged → ValueChanged<bool>
Called when the user toggles with switch on or off. [...]

trackColor → Color
The color to use for the background when the switch is off. [...]

value → bool
Whether this switch is on or off. [...]

CupertinoTabBar



Constructors

CupertinoTabBar({Key key, @required List<BottomNavigationBarItem> items, ValueChanged<int> onTap, int currentIndex: 0, Color backgroundColor, Color activeColor, Color inactiveColor: _kDefaultTabBarInactiveColor, double iconSize: 30.0, Border border: const Border(top: BorderSide(color: _kDefaultTabBarBorderColor, width: 0.0, style: BorderStyle.solid))})

Creates a tab bar in the iOS style.

Properties

activeColor → Color
The foreground color of the icon and title for the BottomNavigationBarItem of the selected tab. [...]

backgroundColor → Color
The background color of the tab bar. If it contains transparency, the tab bar will automatically produce a blurring effect to the content behind it. [...]

border → Border
The border of the CupertinoTabBar. [...]

currentIndex → int
The index into items of the current active item. [...]

iconSize → double
The size of all of the BottomNavigationBarItem icons. [...]

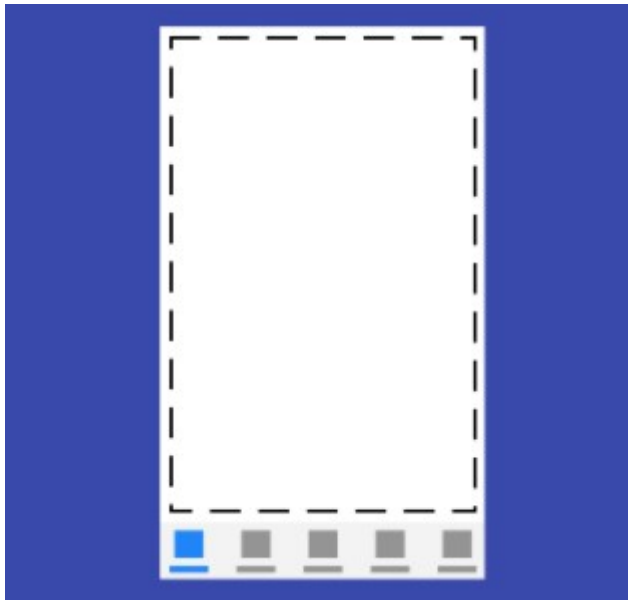
inactiveColor → Color
The foreground color of the icon and title for the BottomNavigationBarItems in the unselected state. [...]

items → List<BottomNavigationBarItem>
The interactive items laid out within the bottom navigation bar. [...]

onTap → ValueChanged<int>
The callback that is called when a item is tapped. [...]

preferredSize → Size
The size this widget would prefer if it were otherwise unconstrained. [...]
read-only, override

CupertinoTabScaffold



Constructors

CupertinoTabScaffold({Key key, @required CupertinoTabBar tabBar, @required IndexedWidgetBuilder tabBuilder, CupertinoTabController controller, Color backgroundColor, bool resizeToAvoidBottomInset: true})

Creates a layout for applications with a tab bar at the bottom. [...]

Properties

backgroundColor → Color
The color of the widget that underlies the entire scaffold. [...]

controller → CupertinoTabController
Controls the currently selected tab index of the tabBar, as well as the active tab index of the tabBuilder. Providing a different controller will also update the scaffold's current active index to the new controller's index value. [...]

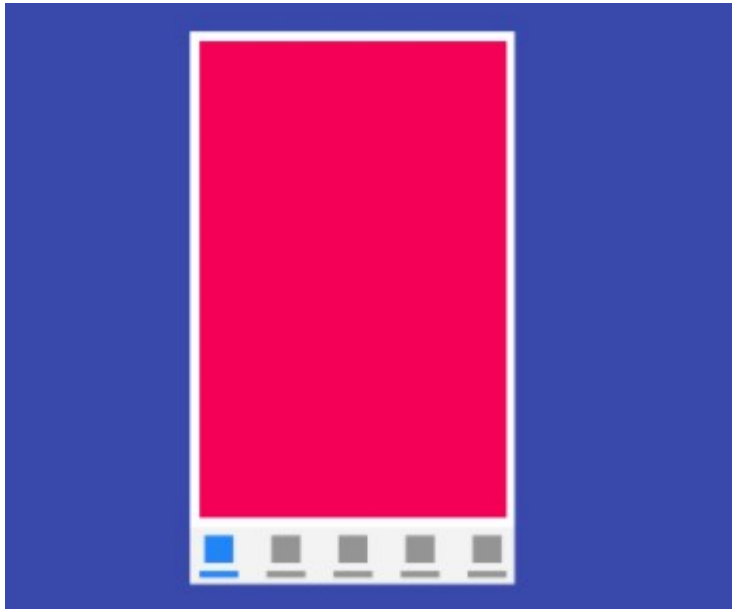
resizeToAvoidBottomInset → bool
Whether the body should size itself to avoid the window's bottom inset. [...]

tabBar → CupertinoTabBar
The tabBar is a CupertinoTabBar drawn at the bottom of the screen that lets the user switch between different tabs in the main content area when present. [...]

tabBuilder → IndexedWidgetBuilder
An IndexedWidgetBuilder that's called when tabs become active. [...]
final

```
CupertinoTabScaffold(
  tabBar: CupertinoTabBar(
    items: <BottomNavigationBarItem> [
      // ... ],),
  tabBuilder: (BuildContext context, int index) {
    return CupertinoTabView(
      builder: (BuildContext context) {
        return CupertinoPageScaffold(
          navigationBar: CupertinoNavigationBar(
            middle: Text('Page 1 of tab $index'), ),
          child: Center(
            child: CupertinoButton(
              child: const Text('Next page'),
              onPressed: () {
                Navigator.of(context).push(
                  CupertinoPageRoute<void>(
                    builder: (BuildContext context) {
                      return CupertinoPageScaffold(
                        navigationBar: CupertinoNavigationBar(
                          middle: Text('Page 2 of tab $index'),       ),
                        child: Center(
                          child: CupertinoButton(
                            child: const Text('Back'),
                            onPressed: () { Navigator.of(context).pop(); },
```

CupertinoTabView



Constructors

CupertinoTabView({Key key, WidgetBuilder builder, GlobalKey<NavigatorState> navigatorKey, String defaultTitle, Map<String, WidgetBuilder> routes, RouteFactory onGenerateRoute, RouteFactory onUnknownRoute, List<NavigatorObserver> navigatorObservers: const <NavigatorObserver>[]})

Creates the content area for a tab in a CupertinoTabScaffold.

const

Properties

builder → WidgetBuilder
The widget builder for the default route of the tab view (Navigator.defaultRouteName, which is /). [...]

defaultTitle → String
The title of the default route.

navigatorKey → GlobalKey<NavigatorState>
A key to use when building this widget's Navigator. [...]

navigatorObservers → List<NavigatorObserver>
The list of observers for the Navigator created in this tab view. [...]

onGenerateRoute → RouteFactory
The route generator callback used when the tab view is navigated to a named route. [...]

onUnknownRoute → RouteFactory
Called when onGenerateRoute also fails to generate a route. [...]

routes → Map<String, WidgetBuilder>
This tab view's routing table. [...]

CupertinoTextField

Constructors

CupertinoTextField({Key key, TextEditingController controller, FocusNode focusNode, BoxDecoration decoration: _kDefaultRoundedBorderDecoration, EdgeInsetsGeometry padding: const EdgeInsets.all(6.0), String placeholder, TextStyle placeholderStyle: const TextStyle(fontWeight: FontWeight.w400, color: CupertinoColors.placeholderText), Widget prefix, OverlayVisibilityMode prefixMode: OverlayVisibilityMode.always, Widget suffix, OverlayVisibilityMode suffixMode: OverlayVisibilityMode.always, OverlayVisibilityMode clearButtonMode: OverlayVisibilityMode.never, TextInputType keyboardType, TextInputAction textInputAction, TextCapitalization textCapitalization: TextCapitalization.none, TextStyle style, StrutStyle strutStyle, TextAlign textAlign: TextAlign.start, TextAlignVertical textAlignVertical, bool readOnly: false, ToolbarOptions toolbarOptions, bool showCursor, bool autofocus: false, String obscuringCharacter: '•', bool obscureText: false, bool autocorrect: true, SmartDashesType smartDashesType, SmartQuotesType smartQuotesType, bool enableSuggestions: true, int maxLines: 1, int minLines, bool expands: false, int maxLength, bool maxLengthEnforced: true, ValueChanged<String> onChanged, VoidCallback onEditingComplete, ValueChanged<String> onSubmitted, List<TextInputFormatter> inputFormatters, bool enabled, double cursorWidth: 2.0, double cursorHeight, Radius cursorRadius: const Radius.circular(2.0), Color cursorColor, BoxHeightStyle selectionHeightStyle: ui.BoxHeightStyle.tight, BoxWidthStyle selectionWidthStyle: ui.BoxWidthStyle.tight, Brightness keyboardAppearance, EdgeInsets scrollPadding: const EdgeInsets.all(20.0), DragStartBehavior dragStartBehavior: DragStartBehavior.start, bool enableInteractiveSelection: true, GestureTapCallback onTap, ScrollController scrollController, ScrollPhysics scrollPhysics, Iterable<String> autofillHints, String restorationId})

Creates an iOS-style text field. [...]

Properties
autocorrect → bool
Whether to enable autocorrection. [...]

autofillHints → Iterable<String>
A list of strings that helps the autofill service identify the type of this text input. [...]

autofocus → bool
Whether this text field should focus itself if nothing else is already focused. [...]

clearButtonMode → OverlayVisibilityMode
Show an iOS-style clear button to clear the current text entry. [...]

controller → TextEditingController
Controls the text being edited. [...]

cursorColor → Color
The color to use when painting the cursor. [...]

cursorHeight → double
How tall the cursor will be. [...]

cursorRadius → Radius
How rounded the corners of the cursor should be. [...]

cursorWidth → double
How thick the cursor will be. [...]

decoration → BoxDecoration
Controls the BoxDecoration of the box behind the text input. [...]

dragStartBehavior → DragStartBehavior
Determines the way that drag start behavior is handled. [...]

enabled → bool
Disables the text field when false. [...]

enableInteractiveSelection → bool
Whether to enable user interface affordances for changing the text selection. [...]

enableSuggestions → bool
Whether to show input suggestions as the user types. [...]

expands → bool
Whether this widget's height will be sized to fill its parent. [...]

focusNode → FocusNode
An optional focus node to use as the focus node for this widget. [...]

inputFormatters → List<TextInputFormatter>
Optional input validation and formatting overrides. [...]

keyboardAppearance → Brightness
The appearance of the keyboard. [...]

keyboardType → TextInputType
The type of keyboard to use for editing the text. [...]
final
maxLength → int
The maximum number of characters (Unicode scalar values) to allow in the text field. [...]

maxLengthEnforced → bool
If true, prevents the field from allowing more than maxLength characters. [...]

maxLines → int
The maximum number of lines for the text to span, wrapping if necessary. [...]

minLines → int
The minimum number of lines to occupy when the content spans fewer lines. [...]

obscureText → bool
Whether to hide the text being edited (e.g., for passwords). [...]

obscuringCharacter → String
Character used for obscuring text if obscureText is true. [...]

onChanged → ValueChanged<String>
Called when the user initiates a change to the TextField's value: when they have inserted or deleted text. [...]

onEditingComplete → VoidCallback
Called when the user submits editable content (e.g., user presses the "done" button on the keyboard). [...]

onSubmitted → ValueChanged<String>
Called when the user indicates that they are done editing the text in the field. [...]

onTap → GestureTapCallback
Called for each distinct tap except for every second tap of a double tap. [...]

padding → EdgeInsetsGeometry
Padding around the text entry area between the prefix and suffix or the clear button when clearButtonMode is not never. [...]

placeholder → String
A lighter colored placeholder hint that appears on the first line of the text field when the text entry is empty. [...]

placeholderStyle → TextStyle
The style to use for the placeholder text. [...]

prefix → Widget
An optional Widget to display before the text.

prefixMode → OverlayVisibilityMode
Controls the visibility of the prefix widget based on the state of text entry when the prefix argument is not null. [...]

readOnly → bool
Whether the text can be changed. [...]

restorationId → String
Restoration ID to save and restore the state of the text field. [...]

scrollController → ScrollController
The ScrollController to use when vertically scrolling the input. [...]

scrollPadding → EdgeInsets
Configures padding to edges surrounding a Scrollable when the Textfield scrolls into view. [...]

scrollPhysics → ScrollPhysics
The ScrollPhysics to use when vertically scrolling the input. [...]

selectionEnabled → bool
Same as enableInteractiveSelection. [...]

selectionHeightStyle → BoxHeightStyle
Controls how tall the selection highlight boxes are computed to be. [...]

selectionWidthStyle → BoxWidthStyle
Controls how wide the selection highlight boxes are computed to be. [...]

showCursor → bool
Whether to show cursor. [...]

smartDashesType → SmartDashesType
Whether to allow the platform to automatically format dashes. [...]

smartQuotesType → SmartQuotesType
Whether to allow the platform to automatically format quotes. [...]

strutStyle → StrutStyle
The strut style used for the vertical layout. [...]

style → TextStyle
The style to use for the text being edited. [...]

suffix → Widget
An optional Widget to display after the text.

suffixMode → OverlayVisibilityMode
Controls the visibility of the suffix widget based on the state of text entry when the suffix argument is not null. [...]

textAlign → TextAlign
How the text should be aligned horizontally. [...]

textAlignVertical → TextAlignVertical
How the text should be aligned vertically. [...]

textCapitalization → TextCapitalization
Configures how the platform keyboard will select an uppercase or lowercase keyboard. [...]

textInputAction → TextInputAction
The type of action button to use for the keyboard. [...]

toolbarOptions → ToolbarOptions
Configuration of toolbar options. [...]

CupertinoTimerPicker

Constructors

CupertinoTimerPicker({Key key, CupertinoTimerPickerMode mode: CupertinoTimerPickerMode.hms, Duration initialTimerDuration: Duration.zero, int minuteInterval: 1, int secondInterval: 1, AlignmentGeometry alignment: Alignment.center, Color backgroundColor, @required ValueChanged<Duration> onTimerDurationChanged})

Constructs an iOS style countdown timer picker. [...]

Properties

alignment → AlignmentGeometry
Defines how the timer picker should be positioned within its parent. [...]
final
backgroundColor → Color
Background color of timer picker. [...]

initialTimerDuration → Duration
The initial duration of the countdown timer.

minuteInterval → int
The granularity of the minute spinner. Must be a positive integer factor of 60.
final
mode → CupertinoTimerPickerMode
The mode of the timer picker.
final
onTimerDurationChanged → ValueChanged<Duration>
Callback called when the timer duration changes.

secondInterval → int
The granularity of the second spinner. Must be a positive integer factor of 60.
final