User Interface

« Back to index

User Interface

Table of Contents:

Button

Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (`Enabled`). Its properties can be changed in the Designer or in the Blocks Editor.

Properties

BackgroundColor color

Specifies the `Button`'s background color as an alpha-red-green-blue integer. If an `Image` has been set, the color change will not be visible until the `Image` is removed.

Enabled boolean

Specifies whether the `Button` should be active and clickable.

FontBold boolean

Specifies whether the text of the `Button` should be bold. Some fonts do not support bold.

FontItalic boolean

Specifies whether the text of the `Button` should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the `Button`, measured in sp(scale-independent pixels).

FontTypeface number designer-only

Specifies the text font face of the `Button` as default, serif, sans serif, or monospace.

Height number blocks-only

Specifies the `Button`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `Button`'s vertical height as a percentage of the `Screen'S Height`.

Image text

Specifies the path of the `Button`'s image. If there is both an `Image` and a `BackgroundColor` specified, only the `Image` will be visible.

Shape number designer-only

Specifies the shape of the `Button`. The valid values for this property are `0` (default), `1` (rounded), `2` (rectangle), and `3` (oval). The `Shape` will not be visible if an `Image` is used.

ShowFeedback boolean

Specifies if a visual feedback should be shown when a `Button` with an assigned `Image` is pressed.

Text text

Specifies the text displayed by the `Button`.

TextAlignment number designer-only

Specifies the alignment of the `Button`'s text. Valid values are: `0` (normal; e.g., left-justified if text is written left to right), `1` (center), or `2` (opposite; e.g., right-justified if text is written left to right).

TextColor color

Specifies the text color of the `Button` as an alpha-red-green-blue integer.

Visible boolean

Specifies whether the `Button` should be visible on the screen. Value is `true` if the `Button` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `Button`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `Button` as a percentage of the `Screen's Width`.

Events

`Click()`

Indicates that the user tapped and released the `Button`.

`GotFocus()`

Indicates the cursor moved over the `Button` so it is now possible to click it.

`LongClick()`

Indicates that the user held the `Button` down.

`LostFocus()`

Indicates the cursor moved away from the `Button` so it is now no longer possible to click it.

`TouchDown()`

Indicates that the `Button` was pressed down.

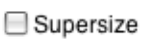
`TouchUp()`

Indicates that the `Button` has been released.

Methods

None

CheckBox



`CheckBox` components can detect user taps and can change their boolean state in response.

A `CheckBox` component raises an event when the user taps it. There are many properties affecting its appearance that can be set in the Designer or Blocks Editor.

Properties

BackgroundColor color

Specifies the background color of the `CheckBox` as an alpha-red-green-blue integer.

Checked boolean

Set to `true` if the box is checked, `false` otherwise.

Enabled boolean

Specifies whether the `CheckBox` should be active and clickable.

FontBold boolean designer-only

Specifies whether the text of the `CheckBox` should be bold. Some fonts do not support bold.

FontItalic boolean designer-only

Specifies whether the text of the `CheckBox` should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the `CheckBox`, measured in sp(scale-independent pixels).

FontTypeface number designer-only

Specifies the text font face of the `CheckBox` as default, serif, sans serif, or monospace.

Height number blocks-only

Specifies the `CheckBox`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `CheckBox`'s vertical height as a percentage of the `Screen's Height`.

Text text

Specifies the text displayed by the `CheckBox`.

TextColor color

Specifies the text color of the `CheckBox` as an alpha-red-green-blue integer.

Visible boolean

Specifies whether the `CheckBox` should be visible on the screen. Value is `true` if the `CheckBox` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `CheckBox`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `CheckBox` as a percentage of the `Screen's Width`.

Events

`Changed()`

User tapped and released the `CheckBox`.

`GotFocus()`

`CheckBox` became the focused component.

`LostFocus()`

`CheckBox` stopped being the focused component.

Methods

None

DatePicker

A button that, when clicked on, launches a popup dialog to allow the user to select a date on the Gregorian Calendar.

Note: Date and time are manipulated using methods in the Clock component.

Properties

BackgroundColor color

Specifies the `DatePicker`'s background color as an alpha-red-green-blue integer. If an `Image` has been set, the color change will not be visible until the `Image` is removed.

Day number read-only, blocks-only

Returns the Day of the month that was last picked using the DatePicker.

Enabled boolean

Specifies whether the `DatePicker` should be active and clickable.

FontBold boolean

Specifies whether the text of the `DatePicker` should be bold. Some fonts do not support bold.

FontItalic boolean

Specifies whether the text of the `DatePicker` should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the `DatePicker`, measured in sp(scale-independent pixels).

FontTypeface number designer-only

Specifies the text font face of the `DatePicker` as default, serif, sans serif, or monospace.

Height number blocks-only

Specifies the `DatePicker`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `DatePicker`'s vertical height as a percentage of the `Screen's Height`.

Image text

Specifies the path of the `DatePicker`'s image. If there is both an `Image` and a `BackgroundColor` specified, only the `Image` will be visible.

Instant read-only, blocks-only

Returns instant of the date that was last picked using the DatePicker.

Month number read-only, blocks-only

Returns the number of the Month that was last picked using the DatePicker.

MonthInText text read-only, blocks-only

Returns the name of the Month that was last picked using the DatePicker.

Shape number designer-only

Specifies the shape of the `DatePicker`. The valid values for this property are `0` (default), `1` (rounded), `2` (rectangle), and `3` (oval). The `Shape` will not be visible if an `Image` is used.

ShowFeedback boolean

Specifies if a visual feedback should be shown when a `DatePicker` with an assigned `Image` is pressed.

Text text

Specifies the text displayed by the `DatePicker`.

TextAlignment number designer-only

Specifies the alignment of the `DatePicker`'s text. Valid values are: `0` (normal; e.g., left-justified if text is written left to right), `1` (center), or `2` (opposite; e.g., right-justified if text is written left to right).

TextColor color

Specifies the text color of the `DatePicker` as an alpha-red-green-blue integer.

Visible boolean

Specifies whether the `DatePicker` should be visible on the screen. Value is `true` if the `DatePicker` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `DatePicker`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `DatePicker` as a percentage of the `Screen's Width`.

Year number read-only, blocks-only

Returns the Year that was last picked using the DatePicker.

Events

`AfterDateSet()`

Event that runs after the user chooses a Date in the dialog.

`GotFocus()`

Indicates the cursor moved over the `DatePicker` so it is now possible to click it.

`LostFocus()`

Indicates the cursor moved away from the `DatePicker` so it is now no longer possible to click it.

`TouchDown()`

Indicates that the `DatePicker` was pressed down.

`TouchUp()`

Indicates that the `DatePicker` has been released.

Methods

`LaunchPicker()`

Launches the DatePicker dialog. The `AfterDateSet` event will be run after the user confirms their selection.

`SetDateToDisplay(` number ***year*****`,`** number ***month*****`,`** number ***day*****`)`**

Allows the user to set the date to be displayed when the date picker opens. Valid values for the month field are 1-12 and 1-31 for the day field.

`SetDateToDisplayFromInstant(`*`instant`*`)`

Allows the user to set the date from the instant to be displayed when the date picker opens.

Image

Component for displaying images and basic animations.

The picture to display, and other aspects of the Image's appearance, can be specified in the Designer or in the Blocks Editor.

Properties

Animation text write-only, blocks-only

This is a limited form of animation that can attach a small number of motion types to images. The allowable motions are `ScrollRightSlow`, `ScrollRight`, `ScrollRightFast`, `ScrollLeftSlow`, `ScrollLeft`, `ScrollLeftFast`, and `Stop`.

Clickable boolean

Specifies whether the image should be clickable or not.

Height number blocks-only

Specifies the `Image`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `Image`'s vertical height as a percentage of the `Screen's Height`.

Picture text

Specifies the path of the `Image`'s `Picture`.

RotationAngle number

The angle at which the image picture appears rotated. This rotation does not appear on the designer screen, only on the device.

ScalePictureToFit boolean write-only

Specifies whether the image should be resized to match the size of the ImageView.

Scaling number blocks-only

This property determines how the picture scales according to the Height or Width of the Image. Scale proportionally (0) preserves the picture aspect ratio. Scale to fit (1) matches the Image area, even if the aspect ratio changes.

Visible boolean

Specifies whether the `Image` should be visible on the screen. Value is `true` if the `Image` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `Image`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `Image` as a percentage of the `Screen's Width`.

Events

`Click()`
An event that occurs when an image is clicked.

Methods

None

Label

Labels are components used to show text.

A label displays text which is specified by the `Text` property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.

Properties

`BackgroundColor` color
Specifies the label's background color as an alpha-red-green-blue integer.

`FontBold` boolean designer-only
Specifies whether the label's text should be bold. Some fonts do not support bold.

`FontItalic` boolean designer-only
Specifies whether the label's text should be italic. Some fonts do not support italic.

`FontSize` number
Specifies the label's text's font size, measured in sp(scale-independent pixels).

`FontTypeface` number designer-only
Specifies the label's text's font face as default, serif, sans serif, or monospace.

`HTMLContent` text read-only, blocks-only
Returns the content of the Label as HTML. This is only useful if the HTMLFormat property is true.

`HTMLFormat` boolean designer-only
Specifies the label's text's format

`HasMargins` boolean
Specifies whether the label should have margins. This margin value is not well coordinated with the designer, where the margins are defined for the arrangement, not just for individual labels.

`Height` number blocks-only
Specifies the `Label`'s vertical height, measured in pixels.

`HeightPercent` number write-only, blocks-only
Specifies the `Label`'s vertical height as a percentage of the `Screen's Height`.

`Text` text
Specifies the text displayed by the label.

`TextAlignment` number designer-only
Specifies the alignment of the label's text: center, normal (e.g., left-justified if text is written left to right), or opposite (e.g., right-justified if text is written left to right).

`TextColor` color
Specifies the label's text color as an alpha-red-green-blue integer.

`Visible` boolean
Specifies whether the `Label` should be visible on the screen. Value is `true` if the `Label` is showing and `false` if hidden.

`Width` number blocks-only
Specifies the horizontal width of the `Label`, measured in pixels.

`WidthPercent` number write-only, blocks-only
Specifies the horizontal width of the `Label` as a percentage of the `Screen's Width`.

Events

None

Methods

None

ListPicker

A button that, when clicked on, displays a list of texts for the user to choose among. The texts can be specified through the Designer or Blocks Editor by setting the `ElementsFromString` property to their string-separated concatenation (for example, `choice 1, choice 2, choice 3`) or by setting the `Elements` property to a List in the Blocks editor.

Setting property ShowFilterBar to true, will make the list searchable. Other properties affect the appearance of the button (TextAlignment, BackgroundColor, etc.) and whether it can be clicked on (Enabled).

Properties

BackgroundColor color

Specifies the ListPicker's background color as an alpha-red-green-blue integer. If an Image has been set, the color change will not be visible until the Image is removed.

Elements list blocks-only

Specifies the list of choices to display.

ElementsFromString text write-only

Set the list of choices from a string of comma-separated values.

Enabled boolean

Specifies whether the ListPicker should be active and clickable.

FontBold boolean

Specifies whether the text of the ListPicker should be bold. Some fonts do not support bold.

FontItalic boolean

Specifies whether the text of the ListPicker should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the ListPicker, measured in sp(scale-independent pixels).

FontTypeface number designer-only

Specifies the text font face of the ListPicker as default, serif, sans serif, or monospace.

Height number blocks-only

Specifies the ListPicker's vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the ListPicker's vertical height as a percentage of the Screen's Height.

Image text

Specifies the path of the ListPicker's image. If there is both an Image and a BackgroundColor specified, only the Image will be visible.

ItemBackgroundColor color

The background color of the ListPicker items.

ItemTextColor color

The text color of the ListPicker items.

Selection text

The selected item. When directly changed by the programmer, the SelectionIndex property is also changed to the first item in the ListPicker with the given value. If the value is not in Elements, SelectionIndex will be set to 0.

SelectionIndex number blocks-only

Selection index property setter method.

Shape number designer-only

Specifies the shape of the ListPicker. The valid values for this property are 0 (default), 1 (rounded), 2 (rectangle), and 3 (oval). The Shape will not be visible if an Image is used.

ShowFeedback boolean

Specifies if a visual feedback should be shown when a ListPicker with an assigned Image is pressed.

ShowFilterBar boolean

If true, the ListPicker will show a search filter bar.

Text text

Specifies the text displayed by the ListPicker.

TextAlignment number designer-only

Specifies the alignment of the ListPicker's text. Valid values are: 0 (normal; e.g., left-justified if text is written left to right), 1 (center), or 2 (opposite; e.g., right-justified if text is written left to right).

TextColor color

Specifies the text color of the ListPicker as an alpha-red-green-blue integer.

Title text

Optional title displayed at the top of the list of choices.

Visible boolean

Specifies whether the ListPicker should be visible on the screen. Value is true if the ListPicker is showing and false if hidden.

Width number blocks-only

Specifies the horizontal width of the ListPicker, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the ListPicker as a percentage of the Screen's Width.

Events

`AfterPicking()`
Event to be raised after the `ListPicker` activity returns its result and the properties have been filled in.

`BeforePicking()`
Event to raise when the `ListPicker` is clicked or the picker is shown using the `Open` method. This event occurs before the picker is displayed, and can be used to prepare the picker before it is shown.

`GotFocus()`
Indicates the cursor moved over the `ListPicker` so it is now possible to click it.

`LostFocus()`
Indicates the cursor moved away from the `ListPicker` so it is now no longer possible to click it.

`TouchDown()`
Indicates that the `ListPicker` was pressed down.

`TouchUp()`
Indicates that the `ListPicker` has been released.

Methods

`Open()`
Opens the `ListPicker`, as though the user clicked on it.

ListView

This is a visible component that allows to place a list of text elements in your `Screen` to display. The list can be set using the `ElementsFromString` property or using the `Elements` block in the blocks editor.

Warning: This component will not work correctly on Screens that are scrollable if its `Height` is set to Fill Parent.

Properties

`BackgroundColor` color
The color of the `ListView` background.

`Elements` list blocks-only
Specifies the list of choices to display.

`ElementsFromString` text write-only
Set the list of choices from a string of comma-separated values.

`Height` number blocks-only
Specifies the `ListView`'s vertical height, measured in pixels.

`HeightPercent` number write-only, blocks-only
Specifies the `ListView`'s vertical height as a percentage of the `Screen`'s `Height`.

`Selection` text
Returns the text in the `ListView` at the position of `SelectionIndex`.

`SelectionColor` color
The color of the item when it is selected.

`SelectionIndex` number blocks-only
The index of the currently selected item, starting at `1`. If no item is selected, the value will be `0`. If an attempt is made to set this to a number less than `1` or greater than the number of items in the `ListView`, `SelectionIndex` will be set to `0`, and `Selection` will be set to the empty text.

`ShowFilterBar` boolean
Sets visibility of the filter bar. `true` will show the bar, `false` will hide it.

`TextColor` color
The text color of the `ListView` items.

`TextSize` number
Specifies the `ListView` item's text font size

`Visible` boolean
Specifies whether the `ListView` should be visible on the screen. Value is `true` if the `ListView` is showing and `false` if hidden.

`Width` number blocks-only
Specifies the horizontal width of the `ListView`, measured in pixels.

`WidthPercent` number write-only, blocks-only
Specifies the horizontal width of the `ListView` as a percentage of the `Screen`'s `Width`.

Events

`AfterPicking()`
Simple event to be raised after the an element has been chosen in the list. The selected element is available in the `Selection` property.

Methods

None

Notifier

The Notifier component displays alert messages and creates Android log entries through an assortment of methods.

Properties

BackgroundColor color write-only

Specifies the background color for alerts (not dialogs).

NotifierLength number designer-only

Specifies the length of time that the alert is shown – either “short” or “long”.

TextColor color

Specifies the text color for alerts (not dialogs).

Events

AfterChoosing(text ***choice*****)**

Event after the user has made a selection for ShowChooseDialog.

AfterTextInput(text ***response*****)**

Event raised after the user has responded to ShowTextDialog.

ChoosingCanceled()

Event raised when the user cancels choosing an option. ShowChooseDialog.

TextInputCanceled()

Event raised when the user cancels ShowChooseDialog, ShowPasswordDialog, or ShowTextDialog.

Methods

DismissProgressDialog()

Dismisses the alert created by the ShowProgressDialog block

LogError(text ***message*****)**

Writes an error message to the Android system log. See the Google Android documentation for how to access the log.

LogInfo(text ***message*****)**

Writes an information message to the Android log.

LogWarning(text ***message*****)**

Writes a warning message to the Android log. See the Google Android documentation for how to access the log.

ShowAlert(text ***notice*****)**

Display a temporary notification.

ShowChooseDialog(text ***message*****,** text ***title*****,** text ***button1Text*****,** text ***button2Text*****,** boolean ***cancelable*****)**

Shows a dialog box with two buttons, from which the user can choose. If `cancelable` is true there will be an additional CANCEL button. Pressing a button will raise the AfterChoosing event. The “choice” parameter to AfterChoosing will be the text on the button that was pressed, or “Cancel” if the CANCEL button was pressed. If canceled, the TextInputCanceled event will also run.

ShowMessageDialog(text ***message*****,** text ***title*****,** text ***buttonText*****)**

Display an alert dialog with a single button that dismisses the alert.

ShowPasswordDialog(text ***message*****,** text ***title*****,** boolean ***cancelable*****)**

Shows a dialog box where the user can enter password (input is masked), after which the AfterTextInput event will be raised. If `cancelable` is true there will be an additional CANCEL button. The AfterTextInput and TextInputCanceled events behave the same way as described in ShowTextDialog.

ShowProgressDialog(text ***message*****,** text ***title*****)**

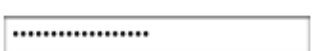
Shows a dialog box with an optional title and message (use empty strings if they are not wanted). This dialog box contains a spinning artifact to indicate that the program is working. It cannot be canceled by the user but must be dismissed by the App Inventor Program by using the DismissProgressDialog method.

ShowTextDialog(text ***message*****,** text ***title*****,** boolean ***cancelable*****)**

Shows a dialog box where the user can enter text, after which the AfterTextInput event will be raised. If `cancelable` is true there will be an additional CANCEL button. Entering text will raise the AfterTextInput event. The “response” parameter to AfterTextInput will be the text that was entered, or “Cancel” if the CANCEL button was pressed. If canceled, the TextInputCanceled event will also run.

PasswordTextBox

Users enter passwords in a password text box component, which hides the text that has been typed in it.



A password text box is the same as the ordinary `TextBox` component, except that it does not display the characters typed by the user.

You can get or set the value of the text in the box with the `Text` property. If `Text` is blank, you can use the `Hint` property to provide the user with a suggestion of what to type. The `Hint` appears as faint text in the box.

Password text box components are usually used with a `Button` component. The user taps the `Button` after entering text.

Properties

BackgroundColor color

The background color of the `PasswordTextBox`. You can choose a color by name in the Designer or in the Blocks Editor. The default background color is 'default' (shaded 3-D look).

Enabled boolean

If set, user can enter text into the `PasswordTextBox`.

FontBold boolean designer-only

Specifies whether the text of the `PasswordTextBox` should be bold. Some fonts do not support bold.

FontItalic boolean designer-only

Specifies whether the text of the `PasswordTextBox` should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the `PasswordTextBox`, measured in sp(scale-independent pixels).

FontTypeface number designer-only

The text font face of the `PasswordTextBox`. Valid values are `0` (default), `1` (serif), `2` (sans serif), or `3` (monospace).

Height number blocks-only

Specifies the `PasswordTextBox`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `PasswordTextBox`'s vertical height as a percentage of the `Screen's Height`.

Hint text

`PasswordTextBox` hint for the user.

PasswordVisible boolean blocks-only

Specifies whether the password is hidden (default) or shown.

Text text

The text in the `PasswordTextBox`, which can be set by the programmer in the Designer or Blocks Editor, or it can be entered by the user (unless the `Enabled` property is false).

TextAlignment number designer-only

Specifies the alignment of the `PasswordTextBox`'s text. Valid values are: `0` (normal; e.g., left-justified if text is written left to right), `1` (center), or `2` (opposite; e.g., right-justified if text is written left to right).

TextColor color

Specifies the text color of the `PasswordTextBox` as an alpha-red-green-blue integer.

Visible boolean

Specifies whether the `PasswordTextBox` should be visible on the screen. Value is `true` if the `PasswordTextBox` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `PasswordTextBox`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `PasswordTextBox` as a percentage of the `Screen's Width`.

Events

`GotFocus()`

Event raised when the `PasswordTextBox` is selected for input, such as by the user touching it.

`LostFocus()`

Event raised when the `PasswordTextBox` is no longer selected for input, such as if the user touches a different text box.

Methods

`RequestFocus()`

Request focus to current `PasswordTextBox`.

Screen

Top-level component containing all other components in the program.

Properties

AboutScreen text

Information about the screen. It appears when "About this Application" is selected from the system menu. Use it to tell users about your app. In multiple screen apps, each screen has its own `AboutScreen` info.

AccentColor color designer-only

This is the accent color used for highlights and other user interface accents in newer versions of Android. Components affected by this property include dialogs created by the `Notifier`, the `DatePicker`, and others.

AlignHorizontal number

A number that encodes how contents of the screen are aligned horizontally. The choices are: `1` (left aligned), `2` (horizontally centered), `3` (right aligned).

AlignVertical number

A number that encodes how the contents of the arrangement are aligned vertically. The choices are: `1` (aligned at the top), `2` (vertically centered), `3` (aligned at the bottom). Vertical alignment has no effect if the screen is scrollable.

AppName text write-only, designer-only

This is the display name of the installed application in the phone. If the `AppName` is blank, it will be set to the name of the project when the project is built.

BackgroundColor color

Specifies the `Screen`'s background color as an alpha-red-green-blue integer. If an `BackgroundImage` has been set, the color change will not be visible until the `BackgroundImage` is removed.

BackgroundImage text

Specifies the path of the `Screen`'s background image. If there is both an `BackgroundImage` and a `BackgroundColor` specified, only the `BackgroundImage` will be visible.

BlocksToolkit text write-only, designer-only

A JSON string representing the subset for the screen. Authors of template apps can use this to control what components, designer properties, and blocks are available in the project.

CloseScreenAnimation text

Sets the animation type for the transition of this form closing and returning to a form behind it in the activity stack.

Height number read-only, blocks-only

Returns the Screen height in pixels (y-size).

Icon text write-only, designer-only

The image used for your App's display icon should be a square png or jpeg image with dimensions up to 1024x1024 pixels. Larger images may cause compiling or installing the app to fail. The build server will generate images of standard dimensions for Android devices.

OpenScreenAnimation text

The animation for switching to another screen. Valid options are `default`, `fade`, `zoom`, `slidehorizontal`, `slidevertical`, and `none`.

Platform text read-only, blocks-only

Gets the name of the underlying platform running the app. Currently, this is the text “Android”. Other platforms may be supported in the future.

PlatformVersion text read-only, blocks-only

Gets the version number of the platform running the app. This is typically a dotted version number, such as 10.0. Any value can be returned, however, so you should take care to handle unexpected data. If the platform version is unavailable, the empty text will be returned.

PrimaryColor color designer-only

This is the primary color used as part of the Android theme, including coloring the `Screen`'s title bar.

PrimaryColorDark color designer-only

This is the primary color used when the Theme property is specified to be Dark. It applies to a number of elements, including the `Screen`'s title bar.

ScreenOrientation text

Declares the requested screen orientation, specified as a text value. Commonly used values are `landscape`, `portrait`, `sensor`, `user` and `unspecified`. See the Android developer documentation for the complete list of possible options.

Scrollable boolean

When checked, there will be a vertical scrollbar on the screen, and the height of the application can exceed the physical height of the device. When unchecked, the application height is constrained to the height of the device.

ShowListsAsJson boolean designer-only

If true (the default), lists will be shown as strings in JSON/Python notation for example [1, “a”, true]. If false, lists will be shown in the LISP notation, for example (1 a true).

Note: This property appears only in Screen1 and the value for Screen1 determines the behavior for all screens in the app.

ShowStatusBar boolean

The status bar is the topmost bar on the screen. This property reports whether the status bar is visible.

Sizing text write-only, designer-only

If set to responsive (the default), screen layouts will use the actual resolution of the device. See the documentation on responsive design in App Inventor for more information. If set to fixed, screen layouts will be created for a single fixed-size screen and autoscaled.

Note: This property appears on Screen1 only and controls the sizing for all screens in the app.

Theme text write-only, designer-only

Selects the theme for the application. Theme can only be set at compile time and the Companion will approximate changes during live development. Possible options are:

- `Classic`, which is the same as older versions of App Inventor;
- `Device Default`, which gives the same theme as the version of Android running on the device and uses PrimaryColor for the Action Bar and has light buttons;
- `Black Title Text`, which is the `Device Default` theme but with black title text; and
- `Dark`, which is a dark version of the `Device Default` theme using `PrimaryColorDark` and having dark grey components.

`Title` text

Title property setter method: sets a new caption for the form in the form's title bar.

`TitleVisible` boolean

The title bar is the top gray bar on the screen. This property reports whether the title bar is visible.

`TutorialURL` text write-only, designer-only

A URL which will be opened on the left side panel (which can be toggled once it is open). This is intended for projects that have an in-line tutorial as part of the project. For security reasons, only tutorials hosted on http://appinventor.mit.edu or linked to from our URL shortener (http://appinv.us) may be used here. Other URLs will be silently ignored.

`VersionCode` number write-only, designer-only

An integer value which must be incremented each time a new Android Application Package File (APK) is created for the Google Play Store.

`VersionName` text write-only, designer-only

A string which can be changed to allow Google Play Store users to distinguish between different versions of the App.

`Width` number read-only, blocks-only

Returns the Screen width in pixels (x-size).

Events

`BackPressed()`

Device back button pressed.

`ErrorOccurred(` component ***`component`***`,` text ***`functionName`***`,` number ***`errorNumber`***`,` text ***`message`***`)`

Event raised when an error occurs. Only some errors will raise this condition. For those errors, the system will show a notification by default. You can use this event handler to prescribe an error behavior different than the default.

`Initialize()`

The Initialize event is run when the Screen starts and is only run once per screen.

`OtherScreenClosed(` text ***`otherScreenName`***`,` any ***`result`***`)`

Event raised when another screen has closed and control has returned to this screen.

`PermissionDenied(` component ***`component`***`,` text ***`functionName`***`,` text ***`permissionName`***`)`

Event to handle when the app user has denied a needed permission.

`PermissionGranted(` text ***`permissionName`***`)`

Event to handle when the app user has granted a needed permission. This event is only run when permission is granted in response to the `AskForPermission` method.

`ScreenOrientationChanged()`

Screen orientation changed

Methods

`AskForPermission(` text ***`permissionName`***`)`

Ask the user to grant access to a sensitive permission, such as `ACCESS_FINE_LOCATION`. This block is typically used as part of a `PermissionDenied` event to ask for permission. If the user grants permission, the `PermissionGranted` event will be run. If the user denies permission, the `PermissionDenied` event will be run.

Note: It is a best practice to only ask for permissions at the time they are needed, which App Inventor components will do when necessary. You should not use `AskForPermission` in your `Initialize` event unless access to that permission is critical to the behavior of your app and is needed up front, such as location services for a navigation app.

`HideKeyboard()`

Hide the soft keyboard

Slider

This class is used to display a `Slider`.

A `Slider` is a progress bar that adds a draggable thumb. You can touch the thumb and drag left or right to set the slider thumb position. As the Slider thumb is dragged, it will trigger the `PositionChanged` event, reporting the position of the `Slider` thumb. The reported position of the thumb can be used to dynamically update another component attribute, such as the `TextBox`'s `FontSize` of a `TextBox` or the Radius of a `Ball`.

The `Slider` uses the following default values. However these values can be changed through the Designer or Blocks editor:

- `MinValue` = 10
- `MaxValue` = 50
- `ThumbPosition` = 30

Properties

ColorLeft color

Specifies the color of the slider bar to the left of the thumb as an alpha-red-green-blue integer, i.e., `0xAARRGGBB`. An alpha of `00` indicates fully transparent and `FF` means opaque.

ColorRight color

Specifies the color of the slider bar to the right of the thumb as an alpha-red-green-blue integer, i.e., `0xAARRGGBB`. An alpha of `00` indicates fully transparent and `FF` means opaque.

HeightPercent number write-only, blocks-only

Specifies the `Slider`'s vertical height as a percentage of the `Screen's Height`.

MaxValue number

Sets the maximum value of slider. If the new maximum is less than the current minimum, then minimum and maximum will both be set to this value. Setting `MaxValue` resets the thumb position to halfway between `MinValue` and `MaxValue` and signals the `PositionChanged`` event.

MinValue number

Sets the minimum value of slider. If the new minimum is greater than the current maximum, then minimum and maximum will both be set to this value. Setting `MinValue` resets the thumb position to halfway between `MinValue` and `MaxValue` and signals the `PositionChanged`` event.

ThumbEnabled boolean

Whether or not the slider thumb is being be shown.

ThumbPosition number

Sets the position of the slider thumb. If this value is greater than `MaxValue`, then it will be set to same value as `MaxValue`. If this value is less than `MinValue`, then it will be set to same value as `MinValue`.

Visible boolean

Specifies whether the `Slider` should be visible on the screen. Value is true if the `Slider` is showing and false if hidden.

Width number blocks-only

Specifies the horizontal width of the `Slider`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `Slider` as a percentage of the `Screen's Width`.

Events

PositionChanged(number ***thumbPosition*****)**

Indicates that position of the slider thumb has changed.

Methods

None

Spinner

A `Spinner` component that displays a dialog with a list of elements. These elements can be set in the Designer or Blocks Editor by setting the `ElementsFromString` property to a comma-separated list of values (for example, `choice 1, choice 2, choice 3`) or by setting the `Elements` property to a List in the Blocks editor. Spinners are created with the first item already selected, so selecting it does not generate an `AfterSelecting` event. Consequently it's useful to make the first `Spinner` item be a non-choice like “Select from below…".

Properties

Elements list blocks-only

Specifies the list of choices to display.

ElementsFromString text write-only

Set the list of choices from a string of comma-separated values.

Height number blocks-only

Specifies the `Spinner`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `Spinner`'s vertical height as a percentage of the `Screen's Height`.

Prompt text

Specifies the text used for the title of the Spinner window.

Selection text

Specifies the current selected item in the `Spinner`.

SelectionIndex number blocks-only

Set the `Spinner` selection to the element at the given index. If an attempt is made to set this to a number less than `1` or greater than the number of items in the `Spinner`, `SelectionIndex` will be set to `0`, and `Selection` will be set to the empty text.

Visible boolean

Specifies whether the `Spinner` should be visible on the screen. Value is `true` if the `Spinner` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `Spinner`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `Spinner` as a percentage of the `Screen's Width`.

Events

AfterSelecting(text ***selection*****)**

Event called after the user selects an item from the dropdown list.

Methods

DisplayDropdown()

Displays the dropdown list for selection, same action as when the user clicks on the spinner.

Switch

`Switch` components can detect user taps and can change their boolean state in response. They are identical to `CheckBox`es except in appearance.

Switches have an on (true) state and an off (false) state. A `Switch` component raises an event when the user taps it to toggle between states.

Properties

BackgroundColor color

Specifies the background color of the `Switch` as an alpha-red-green-blue integer.

Enabled boolean

Specifies whether the `Switch` should be active and clickable.

FontBold boolean designer-only

Specifies whether the text of the `Switch` should be bold. Some fonts do not support bold.

FontItalic boolean designer-only

Specifies whether the text of the `Switch` should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the `Switch`, measured in sp(scale-independent pixels).

FontTypeface number designer-only

Specifies the text font face of the `Switch` as default, serif, sans serif, or monospace.

Height number blocks-only

Specifies the `Switch`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `Switch`'s vertical height as a percentage of the `Screen's Height`.

On boolean

True if the switch is in the On state, false otherwise.

Text text

Specifies the text displayed by the `Switch`.

TextColor color

Specifies the text color of the `Switch` as an alpha-red-green-blue integer.

ThumbColorActive color

Specifies the `Switch`'s thumb color when switch is in the On state.

ThumbColorInactive color

Specifies the `Switch`'s thumb color when switch is in the Off state.

TrackColorActive color

Specifies the `Switch`'s track color when in the On state.

TrackColorInactive color

Specifies the `Switch`'s track color when in the Off state.

Visible boolean

Specifies whether the `Switch` should be visible on the screen. Value is `true` if the `Switch` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `Switch`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `Switch` as a percentage of the `Screen's Width`.

Events

`Changed()`

User change the state of the `Switch` from On to Off or back.

`GotFocus()`

`Switch` became the focused component.

`LostFocus()`

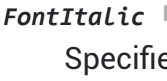
`Switch` stopped being the focused component.

Methods

None

TextBox

Users enter text in a text box component.



The initial or user-entered text value in a text box component is in the `Text` property. If `Text` is blank, you can use the `Hint` property to provide the user with a suggestion of what to type. The `Hint` appears as faint text in the box.

The `MultiLine` property determines if the text can have more than one line. For a single line text box, the keyboard will close automatically when the user presses the Done key. To close the keyboard for multiline text boxes, the app should use the `HideKeyboard` method or rely on the user to press the Back key.

The `NumbersOnly` property restricts the keyboard to accept numeric input only.

Other properties affect the appearance of the text box (`TextAlignment`, `BackgroundColor`, etc.) and whether it can be used (`Enabled`).

Text boxes are usually used with the `Button` component, with the user clicking on the `Button` when text entry is complete.

If the text entered by the user should not be displayed, use `PasswordTextBox` instead.

Properties

BackgroundColor color

The background color of the `TextBox`. You can choose a color by name in the Designer or in the Blocks Editor. The default background color is 'default' (shaded 3-D look).

Enabled boolean

If set, user can enter text into the `TextBox`.

FontBold boolean designer-only

Specifies whether the text of the `TextBox` should be bold. Some fonts do not support bold.

FontItalic boolean designer-only

Specifies whether the text of the `TextBox` should be italic. Some fonts do not support italic.

FontSize number

Specifies the text font size of the `TextBox`, measured in sp(scale-independent pixels).

FontTypeface number designer-only

The text font face of the `TextBox`. Valid values are `0` (default), `1` (serif), `2` (sans serif), or `3` (monospace).

Height number blocks-only

Specifies the `TextBox`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `TextBox`'s vertical height as a percentage of the `Screen's Height`.

Hint text

`TextBox` hint for the user.

MultiLine boolean

If true, then this `TextBox` accepts multiple lines of input, which are entered using the return key. For single line text boxes there is a Done key instead of a return key, and pressing Done hides the keyboard. The app should call the HideKeyboard method to hide the keyboard for a mutiline text box.

NumbersOnly boolean

If true, then this `TextBox accepts only numbers as keyboard input. Numbers can include a decimal point and an optional leading minus sign. This applies to keyboard input only. Even if` NumbersOnly `is true, you can set the text to anything at all using the [`Text`](#TextBox.Text) property.

ReadOnly boolean

Whether the TextBox is read-only. By default, this is true.

Text text

The text in the `TextBox`, which can be set by the programmer in the Designer or Blocks Editor, or it can be entered by the user (unless the `Enabled` property is false).

`TextAlignment` number designer-only

Specifies the alignment of the `TextBox`'s text. Valid values are: `0` (normal; e.g., left-justified if text is written left to right), `1` (center), or `2` (opposite; e.g., right-justified if text is written left to right).

`TextColor` color

Specifies the text color of the `TextBox` as an alpha-red-green-blue integer.

`Visible` boolean

Specifies whether the `TextBox` should be visible on the screen. Value is true if the `TextBox` is showing and false if hidden.

`Width` number blocks-only

Specifies the horizontal width of the `TextBox`, measured in pixels.

`WidthPercent` number write-only, blocks-only

Specifies the horizontal width of the `TextBox` as a percentage of the `Screen's Width`.

Events

`GotFocus()`

Event raised when the `TextBox` is selected for input, such as by the user touching it.

`LostFocus()`

Event raised when the `TextBox` is no longer selected for input, such as if the user touches a different text box.

Methods

`HideKeyboard()`

Hide the keyboard. Only multiline text boxes need this. Single line text boxes close the keyboard when the users presses the Done key.

`RequestFocus()`

Request focus to current `TextBox`.

TimePicker

A button that, when clicked on, opens a dialog to allow the user to select a time.

Note: Date and time are manipulated using methods in the `Clock` component.

Properties

`BackgroundColor` color

Specifies the `TimePicker`'s background color as an alpha-red-green-blue integer. If an `Image` has been set, the color change will not be visible until the `Image` is removed.

`Enabled` boolean

Specifies whether the `TimePicker` should be active and clickable.

`FontBold` boolean

Specifies whether the text of the `TimePicker` should be bold. Some fonts do not support bold.

`FontItalic` boolean

Specifies whether the text of the `TimePicker` should be italic. Some fonts do not support italic.

`FontSize` number

Specifies the text font size of the `TimePicker`, measured in sp(scale-independent pixels).

`FontTypeface` number designer-only

Specifies the text font face of the `TimePicker` as default, serif, sans serif, or monospace.

`Height` number blocks-only

Specifies the `TimePicker`'s vertical height, measured in pixels.

`HeightPercent` number write-only, blocks-only

Specifies the `TimePicker`'s vertical height as a percentage of the `Screen's Height`.

`Hour` number read-only, blocks-only

Returns the hour of the time that was last picked using the `TimePicker`. The time returned is always in the 24hour format.

`Image` text

Specifies the path of the `TimePicker`'s image. If there is both an `Image` and a `BackgroundColor` specified, only the `Image` will be visible.

`Instant` read-only, blocks-only

Returns the instant in time that was last picked using the `TimePicker`.

`Minute` number read-only, blocks-only

Returns the hour of the time that was last picked using the `TimePicker`. The time returned is always in the 24hour format.

`Shape` number designer-only

Specifies the shape of the `TimePicker`. The valid values for this property are `0` (default), `1` (rounded), `2` (rectangle), and `3` (oval). The `Shape` will not be visible if an `Image` is used.

ShowFeedback boolean

Specifies if a visual feedback should be shown when a `TimePicker` with an assigned `Image` is pressed.

Text text

Specifies the text displayed by the `TimePicker`.

TextAlignment number designer-only

Specifies the alignment of the `TimePicker`'s text. Valid values are: `0` (normal; e.g., left-justified if text is written left to right), `1` (center), or `2` (opposite; e.g., right-justified if text is written left to right).

TextColor color

Specifies the text color of the `TimePicker` as an alpha-red-green-blue integer.

Visible boolean

Specifies whether the `TimePicker` should be visible on the screen. Value is `true` if the `TimePicker` is showing and `false` if hidden.

Width number blocks-only

Specifies the horizontal width of the `TimePicker`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `TimePicker` as a percentage of the `Screen`'s `Width`.

Events

`AfterTimeSet()`

This event is run when a user has set the time in the popup dialog.

`GotFocus()`

Indicates the cursor moved over the `TimePicker` so it is now possible to click it.

`LostFocus()`

Indicates the cursor moved away from the `TimePicker` so it is now no longer possible to click it.

`TouchDown()`

Indicates that the `TimePicker` was pressed down.

`TouchUp()`

Indicates that the `TimePicker` has been released.

Methods

`LaunchPicker()`

Launches the `TimePicker` dialog.

`SetTimeToDisplay(` number ***hour*****`,`** number ***minute*****`)`**

Allows the user to set the time to be displayed when the `TimePicker` opens. Valid values for the hour field are 0-23 and 0-59 for the second field.

`SetTimeToDisplayFromInstant(`*`instant`*`)`

Allows the instant to set the hour and minute to be displayed when the `TimePicker` opens. Instants are used in `Clock`, `DatePicker`, and `TimePicker` components.

WebViewer

Component for viewing Web pages.

The `HomeUrl` can be specified in the Designer or in the Blocks Editor. The view can be set to follow links when they are tapped, and users can fill in Web forms.

Warning: This is not a full browser. For example, pressing the phone's hardware Back key will exit the app, rather than move back in the browser history.

You can use the `WebViewString` property to communicate between your app and Javascript code running in the `WebViewer` page. In the app, you get and set `WebViewString`. In the `WebViewer`, you include Javascript that references the `window.AppInventor` object, using the methods `getWebViewString()` and `setWebViewString(text)`.

For example, if the `WebViewer` opens to a page that contains the Javascript command

```
document.write("The answer is" + window.AppInventor.getWebViewString());
```

and if you set `WebViewString` to “hello”, then the web page will show

```
The answer is hello.
```

And if the Web page contains Javascript that executes the command

```
windowAppInventor.setWebViewString("hello from Javascript"),
```

then the value of the `WebViewString` property will be

```
hello from Javascript.
```

Calling `setWebViewString` from JavaScript will also run the `WebViewStringChange` event so that the blocks can handle when the `WebViewString` property changes.

Beginning with release nb184a, you can specify a HomeUrl beginning with `http://localhost/` to reference assets both in the Companion and in compiled apps. Previously, apps needed to use `file:///android_asset/` in compiled apps and `/sdcard/AppInventor/assets/` in the Companion. Both of these options will continue to work but the `http://localhost/` approach will work in both scenarios. You may also use "file:///appinventor_asset/" which provides more security by preventing the use of asynchronous requests from JavaScript in your assets from going out to the web.

Properties

CurrentPageTitle text read-only, blocks-only

Returns the title of the page currently being viewed

CurrentUrl text read-only, blocks-only

Returns the URL currently being viewed. This could be different from the `HomeUrl` if new pages were visited by following links.

FollowLinks boolean

Determines whether to follow links when they are tapped in the `WebViewer`. If you follow links, you can use `GoBack` and `GoForward` to navigate the browser history.

Height number blocks-only

Specifies the `WebViewer`'s vertical height, measured in pixels.

HeightPercent number write-only, blocks-only

Specifies the `WebViewer`'s vertical height as a percentage of the `Screen`'s `Height`.

HomeUrl text

Specifies the URL of the page the `WebViewer` should initially open to. Setting this will load the page.

IgnoreSslErrors boolean

Determine whether or not to ignore SSL errors. Set to true to ignore errors. Use this to accept self signed certificates from websites.

PromptforPermission boolean

Determine if the user should be prompted for permission to use the geolocation API while in the `WebViewer`. If true, prompt the user of the `WebViewer` to give permission to access the geolocation API. If false, assume permission is granted.

UsesLocation boolean write-only, designer-only

Specifies whether or not this `WebViewer` can access the JavaScript geolocation API.

Visible boolean

Specifies whether the `WebViewer` should be visible on the screen. Value is true if the `WebViewer` is showing and false if hidden.

WebViewString text blocks-only

Gets the `WebView`'s String, which is viewable through Javascript in the `WebView` as the `window.AppInventor` object.

Width number blocks-only

Specifies the horizontal width of the `WebViewer`, measured in pixels.

WidthPercent number write-only, blocks-only

Specifies the horizontal width of the `WebViewer` as a percentage of the `Screen`'s `Width`.

Events

BeforePageLoad(text ***url*** **)**

When a page is about to load this event is run.

ErrorOccurred(number ***errorCode*** **,** text ***description*** **,** text ***failingUrl*** **)**

When an error occurs this event is run.

PageLoaded(text ***url*** **)**

When a page is finished loading this event is run.

WebViewStringChange(text ***value*** **)**

Event that runs when the `AppInventor.setWebViewString` method is called from JavaScript. The new `WebViewString` is given by the value parameter.

Methods

boolean **CanGoBack()**

Returns true if the WebViewer can go back in the history list.

boolean **CanGoForward()**

Returns true if the WebViewer can go forward in the history list.

ClearCaches()

Clear the internal webview cache, both ram and disk. This is useful when using the `WebViewer` to poll a page that may not be sending appropriate cache control headers.

`ClearCookies()`
Clear the webview's cookies. This is useful if you want to sign the user out of a website that uses them to store logins.

`ClearLocations()`
Clear Stored Location permissions. When the geolocation API is used in the `WebViewer`, the end user is prompted on a per URL basis for whether or not permission should be granted to access their location. This function clears this information for all locations.

As the permissions interface is not available on phones older then Eclair, this function is a no-op on older phones.

`GoBack()`
Go back to the previous page in the history list. Does nothing if there is no previous page.

`GoForward()`
Go forward to the next page in the history list. Does nothing if there is no next page.

`GoHome()`
Loads the page from the home URL. This happens automatically when home URL is changed.

`GoToUrl(` text ***`url`*****`)`**
Load the page at the given URL.

`Reload()`
Reload the current page.

`RunJavaScript(` text ***`js`*****`)`**
Run JavaScript in the current page.

`StopLoading()`
Stop loading a page.

MIT App Inventor

© 2012-2020 Massachusetts Institute of Technology

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0

Terms of Service and Privacy Policy

Support / Help
Other Inquiries
Twitter: @MITAppInventor
GitHub: mit-cml