

Dictionary	Guides	Lessons	Courses
(https://livecode.com/resources/api/)	(https://livecode.com/resources/guides/)	(http://lessons.livecode.com/)	(https://livecode.com/products/learn/)

Beginners Guide

Resources (https://livecode.com/resources/) / Guides (https://livecode.com/resources/guides/) / Beginners Guide

If you are a beginner, whether to programming or just to LiveCode this guide will introduce you to the LiveCode development environment and guide you step-by-step to building your first app.

Introduction

LiveCode makes it easy to turn your concepts, ideas or designs into powerful applications for Desktop, Mobile and Server.

LiveCode has an easy to use drag and drop interface for creating your user interface, once you have created your UI you use LiveCode's high-level, English like language to code your application. LiveCode is a compile free language so you can run and edit your application live, allowing you to add code gradually, and develop iteratively, testing as you go, with no compile time or delay.

A LiveCode application is completely cross-platform. That means you can build any application you create to run on Windows, Mac OS, Linux, iOS, Android or a server.

This guide will introduce you to the basics of LiveCode, the architecture behind LiveCode, the integrated development environment(IDE), resources and support and how to create your first apps.

Installation

General

To use LiveCode you will need:

- 1024×768 or larger monitor
- True color display (16-bit or 32-bit depth)
- At least 256Mb of memory
- At least 150Mb of disk space

Windows

LiveCode supports the following versions of Windows:

- Windows 2000 SP4
- Windows XP SP2 and above
- Windows Server 2003
- Windows Vista SP1 and above (both 32-bit and 64-bit)
- Windows 7 (both 32-bit and 64-bit)
- Windows Server 2008
- Windows 8 and 8.1 Desktop

Additionally, QuickTime 7 or later is required for most multimedia features.

Mac OS X

LiveCode supports the following versions of Mac OS X:

- 10.3.9 (Panther) on PowerPC
- 10.4.11 (Tiger) on Intel and PowerPC
- 10.5.8 and later (Leopard) on Intel and PowerPC
- 10.6.x (Snow Leopard) on Intel
- 10.7.x (Lion)
- 10.8.x (Mountain Lion)
- 10.9.x (Mavericks)
- 10.10 (Yosemite)

Linux

Offline (Leave a message)

The minimal requirements for LiveCode to run on Linux are:

- 32-bit installation, or a 64-bit linux distribution that has a 32-bit compatibility layer
- 2.4.x or later kernel
- glibc 2.3.2 or later X11R5 capable Xserver running locally on a 24-bit display
- compositing window manager (optional – required for alpha-blended window shapes)
- gtk/gdk/glib (optional – required for native theme support)
- pango/xft (optional – required for pdf printing, anti-aliased text and unicode font support)
- lcms (optional – required for color profile support in JPEGs and PNGs)
- gksu (optional – required for elevate process support)
- mplayer (optional – required for video playback)
- esd (optional – required for audio playback)

Although impossible to test every existing Linux distribution, we are aiming to ensure that LiveCode runs on as wide a variety of systems as possible. To achieve this, the engine has been implemented to have minimal direct dependencies on system software, and will gracefully degrade in feature set if it cannot find the libraries it needs. Generally any recent linux distribution including Gnome/GTK support will have the required libraries for full feature support – for example, Ubuntu 7 supports all these features (although alpha blended window shape support requires you to be running with 'Advance Desktop Effects' turned on).

iOS

Installing the SDK

Before you can use iOS deployment, you need to install the appropriate iOS SDKs available from Apple.

In order to get the iPhone SDK, you need to be a registered iPhone developer. You can register for this and download the SDK by visiting the apple developer site (<http://developer.apple.com/ios>)

LiveCode recommends the following set up:

Platform Xcode SDK Simulators Included

Snow Leopard

- LiveCode 4.5.3
- Xcode 4.2
- SDK 5.0
- Simulators Included 5.0, 4.3

Lion & Mountain Lion

- LiveCode 5.5.3 or above
- XCode 4.6
- SDK 6.1
- Simulators Included 6.1, 6.0, 5.1, 5.0
- LiveCode 5.5.2 or above
- XCode 4.4
- SDK 5.1
- Simulators Included 5.1, 5.0

Mavericks

- LiveCode 6.6.1 or above
- XCode 6.0.1/6.1.1
- SDK 8.0/8.1

Yosemite

- LiveCode 6.7.1/7.0.1 or above
- XCode 6.0.1/6.1.1
- SDK 8.0/8.1

Make sure you have at least one SDK installed, otherwise you will not be able to use the iOS deployment feature.

Note: As a registered iOS developer you will be able to develop and run applications in the iPhone Simulator only. To build applications that can be run on an actual device you will need to enroll in the iOS Developer Programme.

Prerequisites

Snow Leopard

When running on Snow Leopard, LiveCode uses the iOS 5.0 SDK to produce device builds. This is available as part of Xcode 4.2.

The 4.3 and 5.0 simulators are supported on Snow Leopard.

Lion & Mountain Lion

When building for Arm v7 devices, LiveCode uses the iOS 6.1 SDK, available as part of Xcode 4.6.

When building for Arm v6 devices, LiveCode uses the iOS 5.1 SDK, available as part of Xcode 4.4.

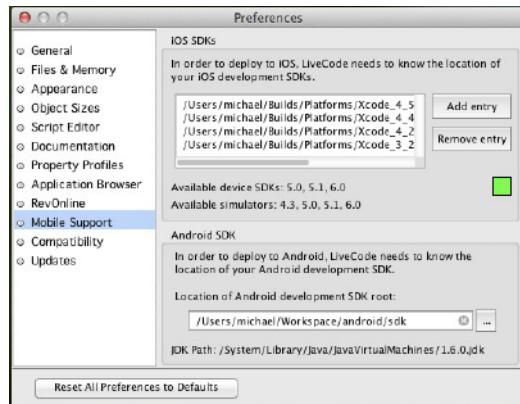
If you wish to produce universal device builds (including both Arm v6 and Arm v7 instructions) you must have the iOS 5.1 and iOS 6.1 SDKs installed.

The 5.0, 5.1, 6.0 and 6.1 simulators are supported on Lion & Mountain Lion.

Configuring LiveCode

After you have installed an iOS SDK, it is necessary to tell LiveCode where to find it (or them, if you have installed more than one).

To configure the paths to your installed SDKs, use the Mobile Support panel in Preferences.



Use this pane to choose the correct SDK paths by using the 'Add entry' button. You should choose the folder you selected when installing the SDK (for Xcode versions 4.2 and earlier) or the Xcode app bundle (for Xcode version 4.3 and later).

When you have successively chosen your SDK(s), the list of simulators and SDKs that you will have available will be updated.

Note: On startup if SDKs have not been previously configured, LiveCode will check to see if there is a recognised SDK at /Developer and /Applications.

You can find more detailed lessons on setting LiveCode up for iOS development here. (<http://lessons.livecode.com/s/lessons/m/4069/l/23275-how-do-i-become-an-ios-developer>)

Android Installing the SDK

Before you can use the Android plugin, you need to ensure you have set up your system appropriately.

Prerequisites

Windows

If you are intending to use the Android deployment pack on Windows, you will need:

- Windows XP/Vista or Windows 7
- LiveCode 4.5.3 or later
- The Java SDK (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
- The Android SDK (<http://developer.android.com/sdk/index.html>)

Mac

If you are intending to use the Android deployment pack on Mac, you will need:

- Mac OS 10.5.x or later
- LiveCode 4.5.3 or later
- The Android SDK (<http://developer.android.com/sdk/index.html>)

Linux

If you are intending to use the Android deployment pack on Linux, you will need:

- LiveCode 5.5.2 or later
- The Java SDK (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
- The Android SDK (<http://developer.android.com/sdk/index.html>)

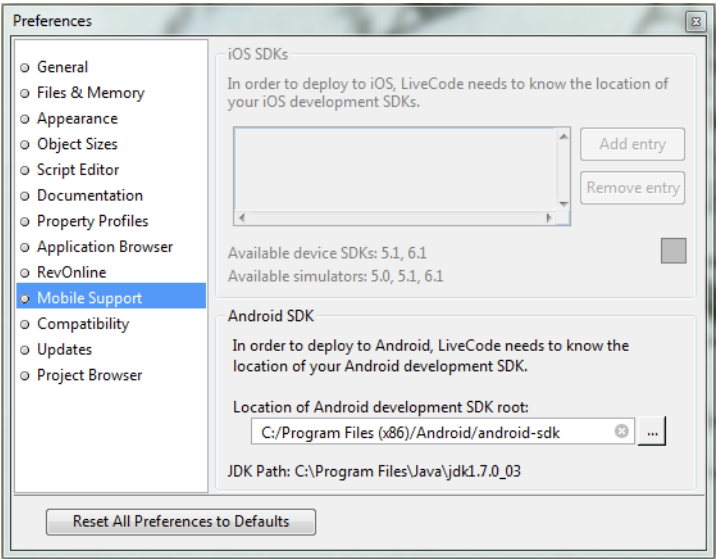
After you have installed the pre-requisites make sure you run the Android SDK Manager and have installed the 'SDK Platform Android 2.2, API 8, revision 2' package.

Configuring LiveCode

After you have set up your system with the Java Development Kit and Android Development Kit, it is necessary to inform LiveCode

where to find them.

To configure the paths to your installed SDKs, use the Android section of the Mobile Support pane in Preferences.



Use this pane to choose the correct SDK paths by clicking the ‘...’ button after the Android development SDK location field.

After doing so, the JDK path should be automatically located. If a path fails to appear for the JDK it means you have not correctly installed or configured your JDK.

If you are developing on Linux you may have to manually set the path of the JDK.

You can find more detailed lessons on setting LiveCode up for Android development here. (<http://lessons.livecode.com/s/lessons/m/4069/c/16395>)

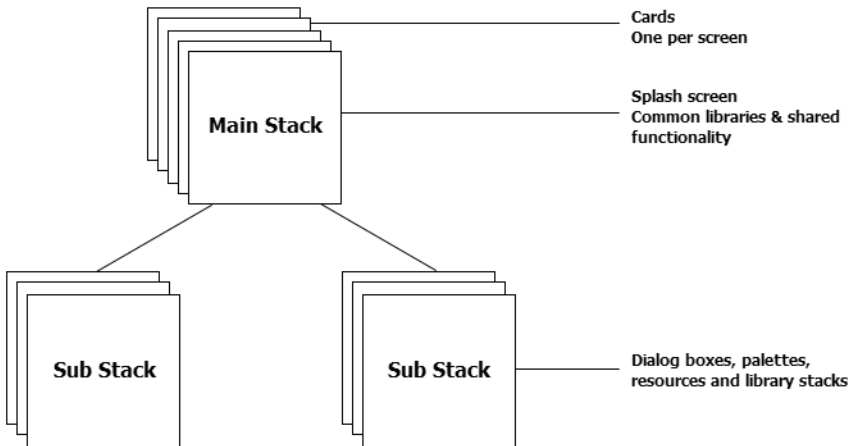
How LiveCode Works

Creating a simple graphical application in LiveCode can take just minutes. First you create a user interface, including any windows, palettes, dialogs you require. Then you populate the user interface with controls, like push buttons, check boxes, text fields or menus. Finally, you use LiveCode’s English-like programming language to tell your application how to behave.

Stacks, cards and objects

The first step in creating a LiveCode application is creating a window, which in LiveCode is called a stack. Each window you see in LiveCode is a stack. Palettes, dialog boxes, and standard windows are all stacks.

Each stack contains one or more sets of information called cards. Each card can have a different appearance or all the cards in a stack can look the same. By going from card to card in a stack, you change what’s being displayed in that stack’s window. You can think of a LiveCode stack as a stack of playing cards (hence the name), where you can flip through the cards, but only one card at a time is visible. A stack can have a single card or many cards.



All user interface objects (controls) are created by dragging them on to a card area.

Object and Event Driven Programming

Any graphical application you build using LiveCode will be based on objects. With LiveCode you typically create the objects of your

application before writing any code. You can start by drawing the buttons, text fields, and other controls that make up your application. LiveCode operates like other layout, drawing or application development environment. You can select controls by clicking them, move them by dragging them around, resize them, and change their layer to move them closer or further from the top of the interface.

Once you have the objects in place, you can proceed to attach code to each object to respond to the events you want. LiveCode includes objects for all the basic operating system elements, including buttons, checkboxes, text fields, menus, graphics, and many more. In addition there are native mobile controls and you can create and customize your own objects that look and behave however you want.

A LiveCode application is driven by user actions. LiveCode constantly watches the computer for common actions, such as clicking on a button, typing into a field, sending data across a network, or quitting an application.

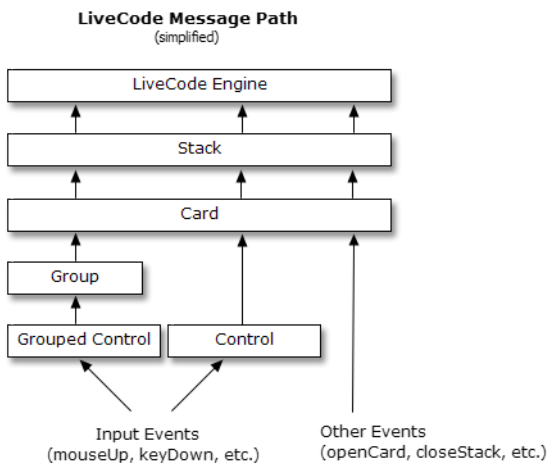
When an event occurs, LiveCode sends a message to the most appropriate object. When writing your program, you decide what messages you want your program to respond to. For example, if a user clicks on a button, LiveCode sends a message to the button. You add code to the button that tells it how to respond to being clicked on.

There are a wide range of possible events. When a user clicks on a button, a series of events are sent to the button. For example, when the mouse first moves within the border of the button a `mouseenter` message is sent. Then a series of `mousemove` messages are sent as the mouse moves over the button. When the mouse button is depressed a `mousedown` message is sent and when the mouse is released a `mouseup` message is sent. You don't have to respond to all of these events. You simply place code within an object to handle the events you want your application to respond to.

Messages and the Message Path

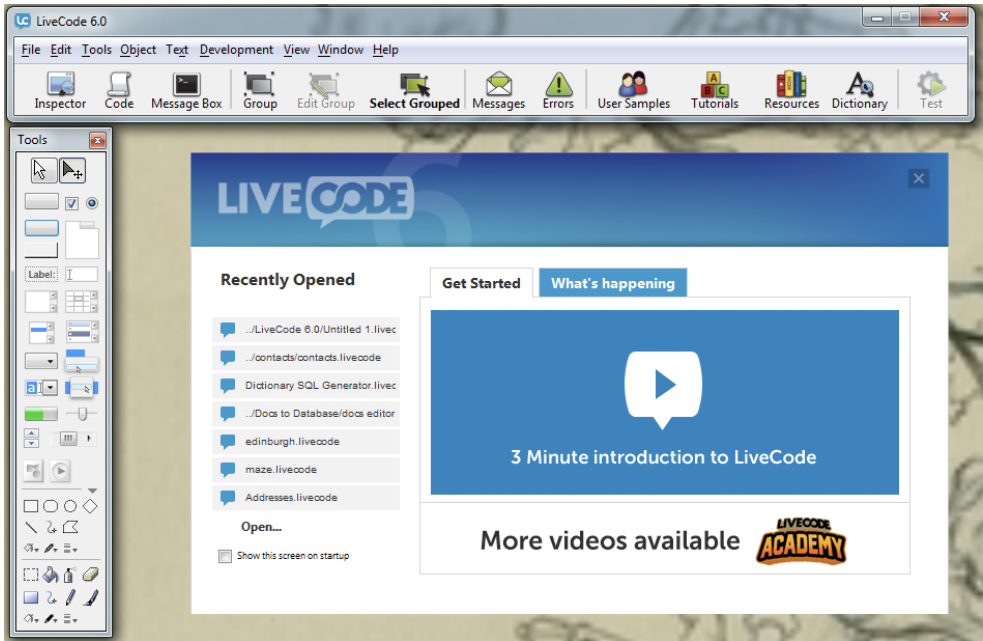
Each LiveCode object is part of another object, of a different object type. For example, each object is part of a card, each card is part of a stack etc. This object hierarchy defines the ownership and inheritance relationship between objects.

If a message is not handled by the object it is initially sent to, meaning you have chosen not to respond to that message, it is passed on to the owner of the initial object. In LiveCode the order in which objects have the opportunity to respond to a message is called the Message Path and is based on the object hierarchy. This makes it possible to group similar functionality together at different levels within your application.



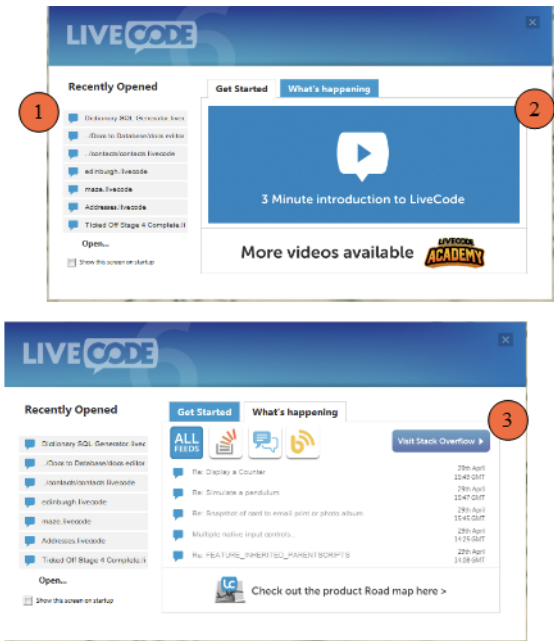
LiveCode Environment

When you start LiveCode you will see a collection of components, these components make up the Integrated Development Environment, or IDE. This is where you will create your application.



The Start Center

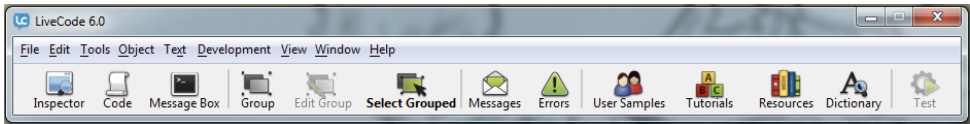
The Start Center introduces you to LiveCode, lets you know what is happening on the RunRev blog, what other users are discussing on the forums and Stack Overflow and provides easy access to your recent stacks.



- 1. List of Recent Stacks
- 2. Introductory Video
- 3. Forums, blog and Stack Overflow feeds

The Menubar

The Menubar provides access to most common functions and features.



As you would expect the File menu allow you to create stacks, save stacks, open files etc. The Edit menu allows you to select, copy and paste objects. The Tools menu provides access to the components being discussed in this section and the other menus provide additional features and help.

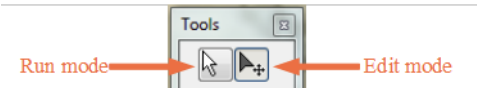
As well as the menus the Menubar allows you to open the Property Inspector or Code Editor for the current object, work with grouped controls in various modes, control messages and error reporting and provides access to documentation and support

resources.

The Tools Palette

The Tools Palette allows you to switch between Edit mode, for adding objects, making changes, coding, and Run mode, for interacting with your application.

Run and Edit Mode



Run mode: When in run mode, objects receive all the normal messages that drive a LiveCode application. For example, clicking on a button in run mode will cause a mouseUp message to be sent to it and the script will run.

Edit mode: When in edit mode, objects do not receive messages when you click on them, and you can move, resize or edit the properties for objects.

Controls

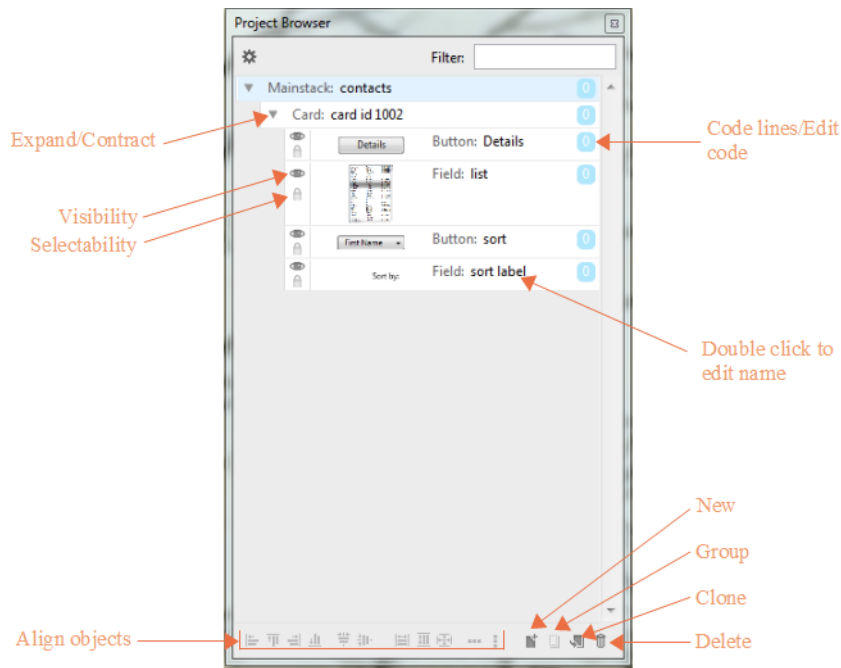


Drag objects such as buttons, fields, scrollbars and graphics from the Tools Palette onto your stack.

Once you have added objects to your stack, choose Edit mode to move, resize and change the properties of objects.

The Project Browser

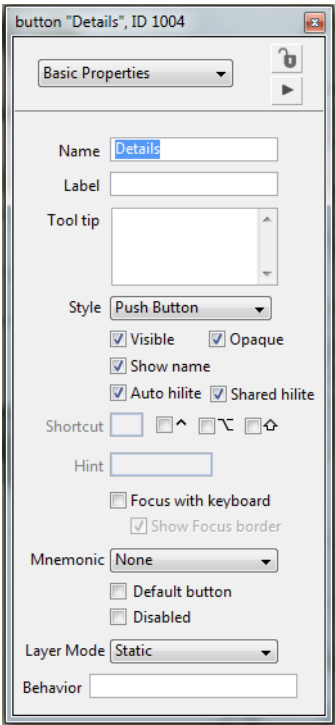
The Project Browser provides an overview of your stacks in tree form, showing you all the open stacks, cards and controls that make up your application.



The Project Browser also allows you to access some of the basic properties of objects as well as editing code, adding and deleting objects and aligning controls.

The Property Inspector

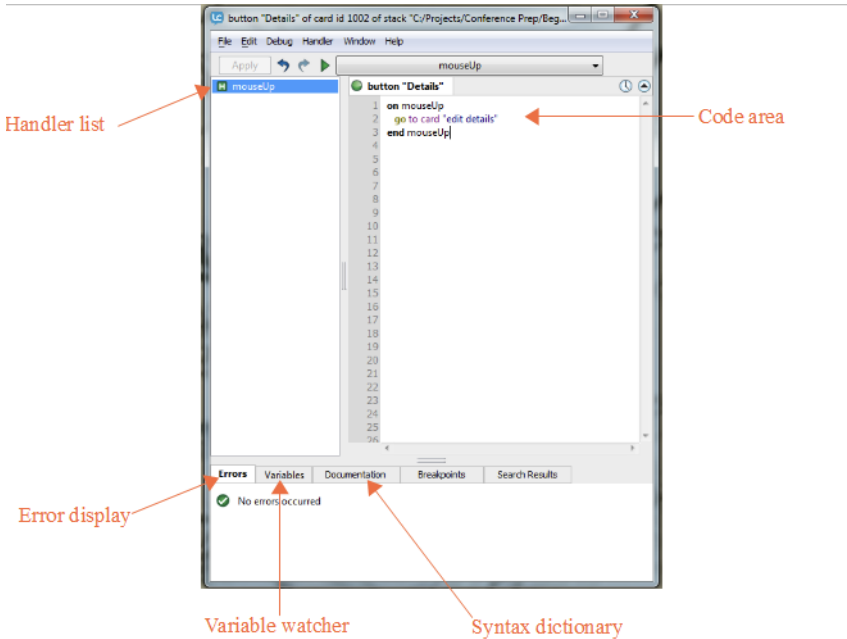
The Property Inspector allows you to view and edit the properties for any selected object. Properties control how an object looks and some aspects of an object's behavior.



The Inspector can be accessed by double clicking on a selected object, from the toolbar, from the Object menu and from context sensitive menus.

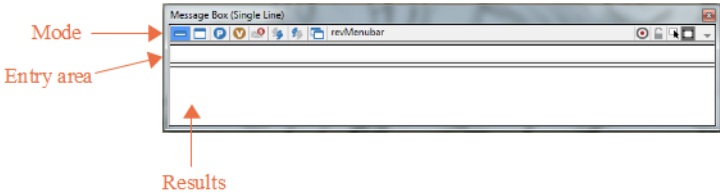
The Code Editor

The Code Editor includes features to help make code more understandable. These include code indentation and color coded syntax highlighting, as well as other integrated tools such as a Debugger and Dictionary. You can access the Code Editor for an object by selecting the object then choosing Script from the Tool bar. The Code Editor is also available from the Object menu, and from the context menu of objects.



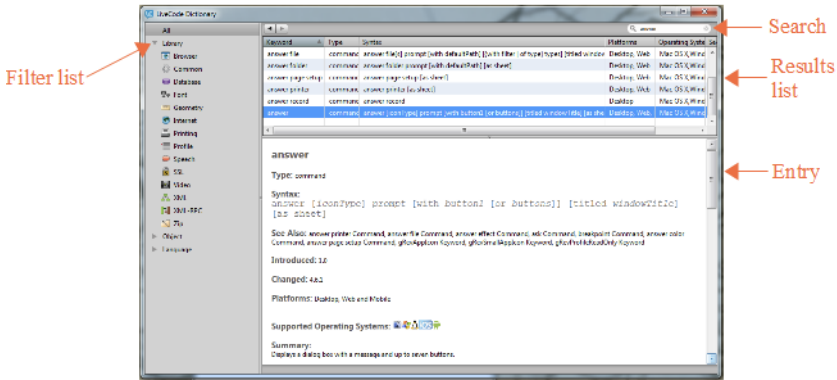
The Message Box

The Message Box is a command line tool that allows you to run scripts or perform automated editing operations. It allows you to try out short scripts, test parts of your program, provides a convenient output window for debugging, show the values of Global Properties and Global Variables and can be used for editing and setting properties.



The Dictionary

The Dictionary contains the complete LiveCode syntax. This can be searched using the quick search box in the top right of the dictionary or filtered using the list of topics on the left.



A dictionary entry for LiveCode keyword gives you the syntax, some examples, a description and the supported platforms and operating systems.

Resources and Support

There is a wealth of support available when you are getting started with LiveCode.

Guides

There are a variety of Guides to various aspects of LiveCode available at
Beginners (<https://livecode.com/resources/guides/beginners-guide>)
Developers (<https://livecode.com/resources/guides/developers-guide>)
iOS Externals (<https://livecode.com/resources/guides/externals>)
LiveCode Server (<https://livecode.com/resources/guides/server>)

Tutorials

A wide range of step-by-step tutorials are available here. (<http://livecode.com/developers/tutorials/>)

Forums

You can join our Forums (<http://livecode.com/community/forums/>) where you can discuss your LiveCode projects with other developers and get assistance from more experienced LiveCode users.

Getting Started

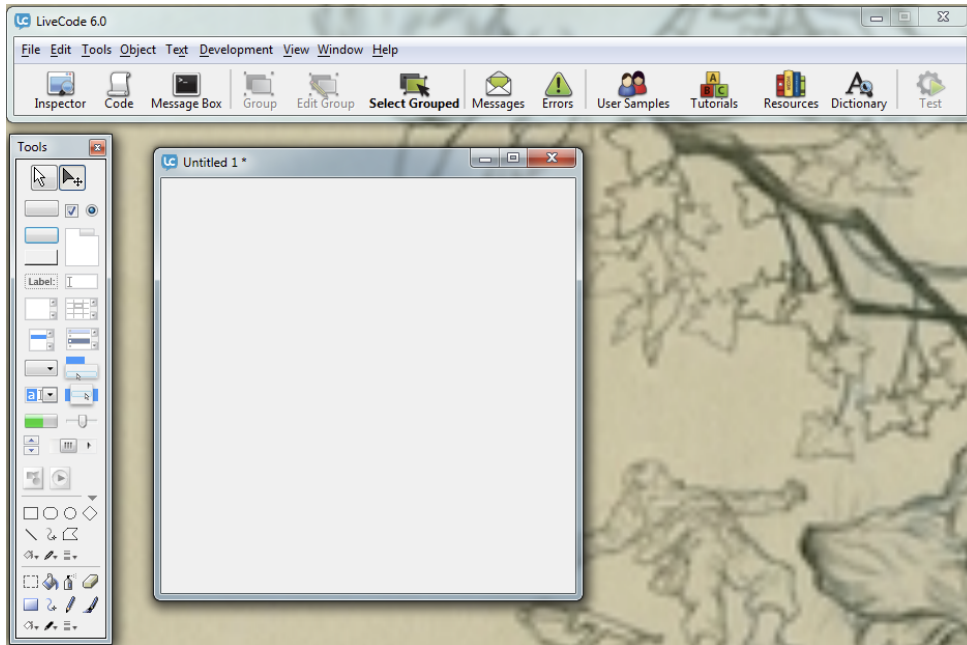
It is very simple to create an application in LiveCode, by dragging objects onto your stack from the Tools Palette and adding code for the messages you want to respond to you can create a simple application in minutes.

This section will cover the basic controls and the most common messages associated with them.

Basic Controls

Stacks

When creating a LiveCode application the first step is to create a new mainstack, you do this by selecting New Mainstack from the File menu.



A new stack is created with 1 card.

Common messages associated with stacks are

preOpenStack: Sent to the destination card when you open a stack. Any preOpenStack handlers are executed before the stack window appears. Because of this, the preOpenStack handler is a good place to put code that adjusts the size, position, and appearance of objects; the changes are made before the stack appears.

openStack: Sent to the destination card right after you open a stack. Handle the openStack message to change a stack's objects, or perform other updates, when the stack is opened. The openStack message is sent after the stack is visible.

closeStack: Sent to the current card when the stack closes. A stack is closed when the user or a handler closes the stack window. Handle the closeStack message if you want to perform cleanup or do other tasks when the user closes a window.

Cards

A stack consists of one or more cards, each card can have a different appearance and moving between cards changes how the application looks.

Common messages and command associated with cards are

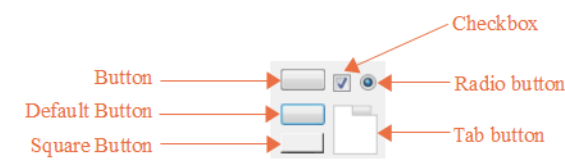
preOpenCard: Sent to a card when you go to the card. Handle the preOpenCard message to update a card's appearance before the card appears on screen.

OpenCard: Sent to a card right after you go to the card. Handle the openCard message to change a card's objects, or perform other updates, when the card is visited.

CloseCard: Sent to the current card when the user goes to another card. Handle the closeCard message if you want to perform cleanup or do other tasks when the user leaves a card.

Buttons

There are 6 types of button in the Tools Palette. All these buttons are fundamentally the same but have different default property sets.



Common messages associated with buttons

mouseUp: Sent when the user releases the mouse button. Handle the mouseUp message to perform an action when the user releases the mouse button after clicking.

Fields

There are 5 types of field in the Tools Palette. All these fields are fundamentally the same but have different default property sets.



Common messages associated with fields

openField: Sent to an unlocked field when you click or select text in that field. Handle the openField message if you want to do something when the user enters a field.

closeField: Sent to a field when the focus is being removed from that field and the field's content has changed. Handle the closeField message if you want to make sure a field's content is correct after it has been changed.

textChanged: Sent when the content of a field has changed. Handle the textChanged message if you want to perform an action when the content of a field changes.

The DataGrid also appears in this section of the Tools Palette. The DataGrid object allows you to use complex tables and forms into your application.

Menus

There are 4 types of menu in the Tools Palette, menus are buttons with certain properties set that make them behave as menus.



Common messages associated with menus

menuPick: Sent to a button when a menu item is chosen from the menu associated with that button. Handle the menuPick message to do something when the user chooses a menu item from a button menu, or chooses a tab in a tabbed button.

Scrollbars

There are 4 types of scrollbar in the Tools Palette, all these scrollabars are fundamentally the same but have different default property sets.



Common messages associated with scrollbars

scrollbarDrag Sent to a field, scrollbar, or group when the user drags the scrollbar thumb or when a text selection causes a field to scroll. Handle the scrollbarDrag message if you want to respond to the user dragging the scrollbar thumb.

Images

An image object is a control that contains a bitmapped picture. Use the image object type to hold photographs, icons, and decorative elements, and to allow the user to paint.



You can set the filename property of an image to display an image stored in a separate file.

Players

A player object is a control that displays a movie or sound file. Set the filename property of a player to display a movie or sound

from a separate file.



Common messages associated with players

playStarted: Sent to a player when it starts playing. Handle the playStarted message if you want to perform a task when a movie or sound starts playing.

playPaused: Sent to a player when the user pauses it. Handle the playPaused message if you want to perform an update when a player is paused.

playStopped: Sent to a player when it stops playing. Handle the playStopped message if you want to perform a task when a movie or sound finishes playing.

Graphics

The Tools Palette allows you to choose from a selection of shapes when creating a new graphic.



Use the fill bucket to choose the fill color, the fill pencil to choose the line color, the line thickness menu to choose the line thickness, and the optional shape menu to choose preferences specific to the type of graphic selected.

Groups

You can create a group by selecting multiple controls and clicking the Group button in the Menubar.

Once you've created the group, it becomes an object in its own right. You can select, copy, move, and resize the group, and all the objects in the group come with it. The objects in the group maintain their own identities, and you can add objects to the group or delete them, but the objects are owned by the group instead of the card.

A group has its own properties and its own script. Groups can be any size, can be shown or hidden, and can be moved to any location in the stack window, just like any other control. Like other controls, groups can be layered in any order with the other controls on the card. Groups can also display a border around a set of objects.

Importing controls

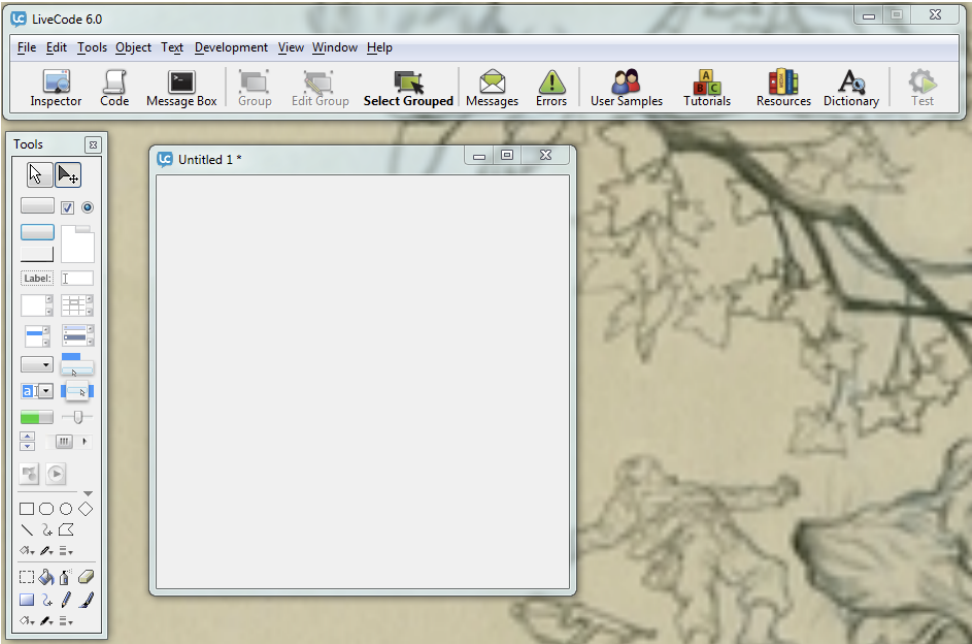
You can also import images into your stack. These become image objects and are part of the stack file. This means that you do not need to include external image files when distributing your application.

To import an image choose Import as Control from the File menu. You can also import Audio and Video files.

Hello World Example

In this section we will create a simple stack consisting of a single button that will display a message to the user. We will use the answer command to display the message. The answer command displays a dialog box with a message and up to seven buttons.

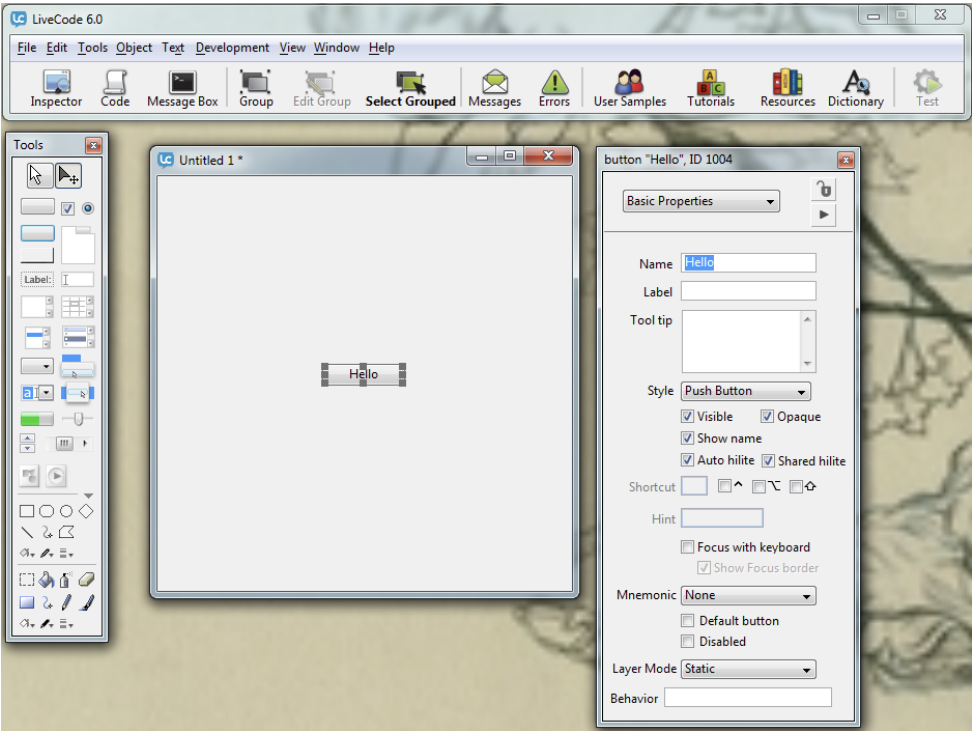
Choose New Mainstack from the File menu. This will create a new stack for you to work with.



Ensure you have Edit mode selected in the Tools Palette.

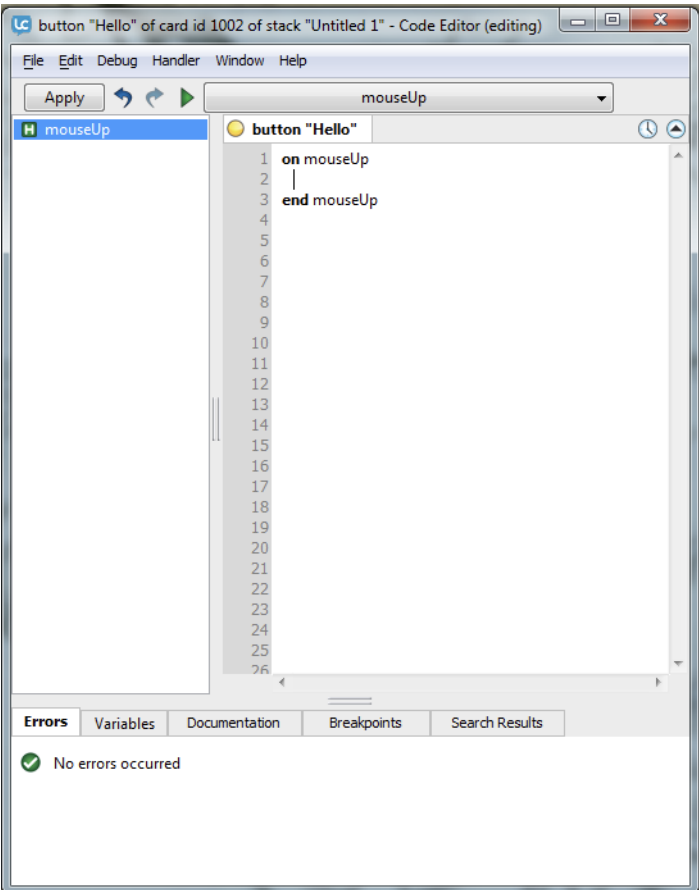


Drag a button onto your stack.



Open the Property Inspector for the button by selecting the button and clicking the Inspector button in the Menubar. Change the name of the button to “Hello”.

Now open the Code Editor by clicking the Code button in the Menubar. As this is a button the Code Editor is pre-populated with an empty on mouseUp handler.

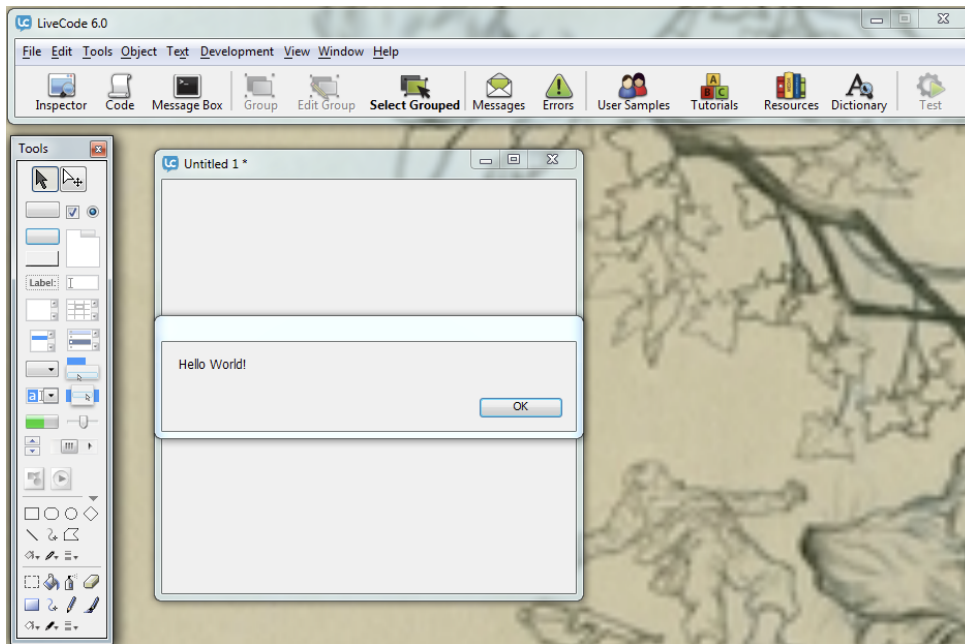


Add the following code to the button

```
on mouseUp
    answer "Hello World!"
end mouseUp
```

Click the Apply button, the indicator will change from yellow to green showing there are no errors in the code.

Switch to Run mode, allowing you to interact with the application, and click the button. You will see a dialog box appear displaying the Hello World message.

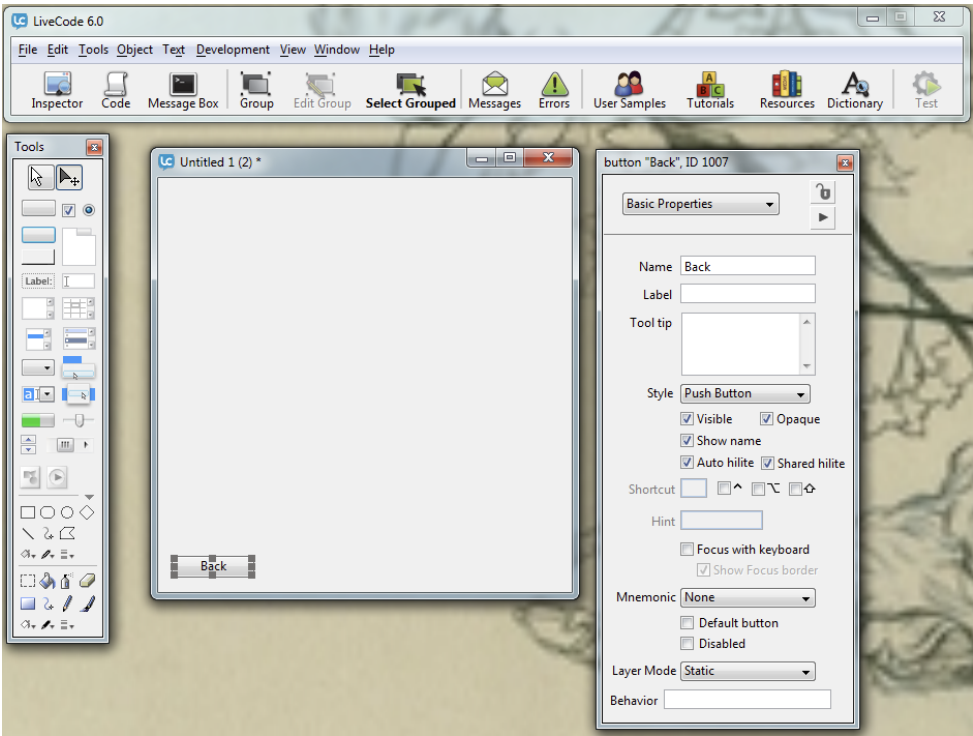


Stacks with Multiple Cards

Your stacks will often consist of more than one card, multiple cards allow you to display different information on screen and give your user access to different options.

Creating new cards

To add additional cards to your stack select New Card from the Object menu. This will create a new card and go to it.



Add a button to this card and name it “Back”.

Moving between cards

To move between cards use the Go command. The Go command navigates between cards. The target card can be identified by name, number or position

```
go to card "edit"
```

```
go to card 1
```

```
go to the first card
```

Open the script editor for the button and add the following code

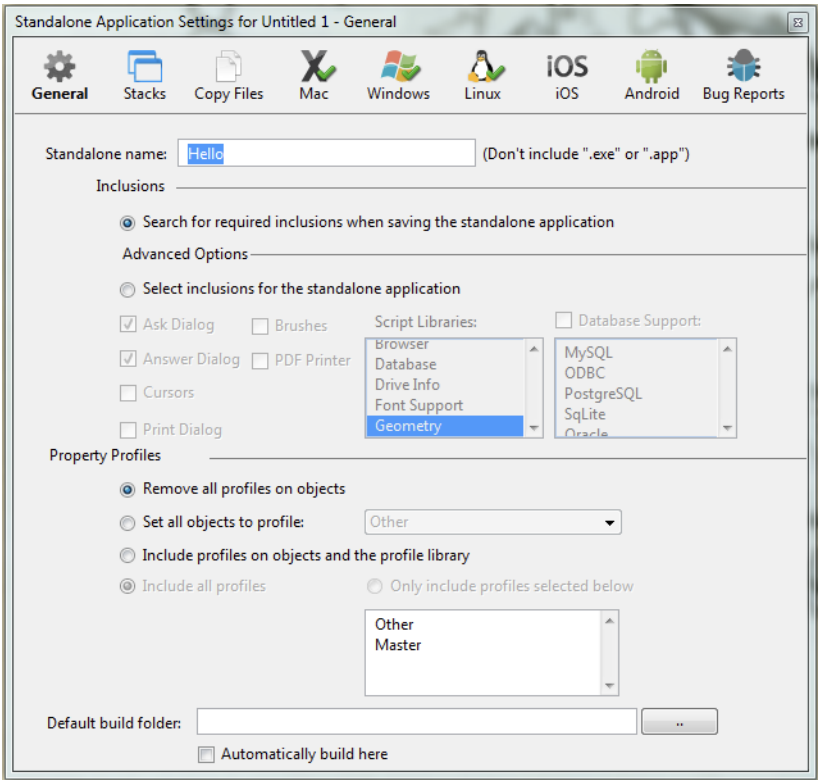
```
on mouseUp
  go to card 1
end mouseUp
```

Apply the code and switch to Run mode.

Click the Back button, this will navigate to the first card.

Building a Standalone

When you have finished your LiveCode application and what to distribute it you can build it into a standalone application. LiveCode stacks can be build for Windows, Mac OS, Linux, iOS and Android.



To build a standalone open the Standalone Application Settings from the File Menu. This is where you select the platforms you want to build for and set up the properties of the standalones.

General Pane

This is where you set the basic properties of the standalone, including the name and any additional libraries that your application requires.

Stacks Pane

You can include additional stacks as resources to your main stack. You add the stacks you want to include in this pane.

Copy Files

Your application may require additional resources such as sound and video files and databases. Use this pane to add any external resources that are to be included in the standalone.

Mac, Windows, Linux, iOS, Android Panes

These panes allow you to set up platform specific properties for the platforms you choose to build for.

Select the platforms you want to build for and tick the checkboxes on the relevant panes.

Once the platforms are selected choose Save As Standalone Application from the File menu and select the folder where the Standalone Application Settings will be saved.

On iOS and Android you can also test your app directly. Once you have installed all the requirements and set up the simulators and devices you can choose the target device or simulator from the Test Target list in the Development menu and then click Test in the Menubar.

On Android you can test this way on any running emulator or connected device, on iOS you can use this method to test on the simulator but to test on a device you need to build a standalone as described above and use XCode to install it.

LiveCode

Why LiveCode?
(<https://livecode.com/core-benefits-of-livecode/>)

Pricing (<https://livecode.com/products/livecode-platform/pricing/>)

Customer Stories
(<https://livecode.com/studies/>)

Resources

Docs (<https://livecode.com/docs/9-0-0/introduction/welcome/>)

API (Language Dictionary)
(<https://livecode.com/resources/api/>)

Lessons
(<http://lessons.livecode.com>)


About


Meet The Team
(<https://livecode.com/about/team/>)

About (<https://livecode.com/about/>)

Press & Media
(<https://livecode.com/about/press/>)

Recent Posts

 LiveCode 9.5 Released
(<https://livecode.com/livecode-9-5-released/>)

 LiveCode 9.5 goes into testing
(<https://livecode.com/livecode-9-5-goes-into-testing/>)


Extensions (/products/ extensions)

LiveCode in Education (https://livecode.com/develop- education/)

LiveCode in Business (https://livecode.com/develop- your-startup/)

LiveCode Conference 2019 (https://livecode.com /california19)

Looking for the LiveCode open source project?

 Open Source (http://livecode.org)

Looking for LiveCode FileMaker?

LiveCode for FM (https://filemaker.livecode.com)

Sample Stacks (http://livecodeshare.runrev.com /search/direction/descending /searchtype/top%20rated/)

Forums (http://forums.livecode.com)

Stackoverflow (http://stackoverflow.com /questions/tagged/livecode)

Roadmap (https://livecode.com /resources/roadmap/)

Contribute to LiveCode (https://github.com/livecode /livecode/blob/develop /CONTRIBUTING.md)

Contribute to Docs (https://github.com/livecode /livecode/blob/develop /docs/contributing_to_docs.md)

Release Process (https://livecode.com/resources /release-process/)

User Groups (https://livecode.com/resources /user-groups/)

Support (https://livecode.com /resources/support/)

Directory (https://livecode.com /directory/)

LiveCode Services (https://livecode.com/services/)


Contact us (https://livecode.com /about/contact/)

Awards (https://livecode.com /about/awards/)

/livecode-9-5-goes-9-5- into-testing/ goes- Introducing LiveCloud (https://livecode.com /introducing- livecloud/) Visitors in LiveCode (https://livecode.com /visitors-in- livecode/) WordLib Update Released (https://livecode.com /wordlib- update- released/)

(https://livecode.com/resources/guides/beginners-guide/)
(page/Stackoverflow/RunRevLtd)
/livecode-9-5-goes-9-5- into-testing/ Introducing LiveCloud (https://livecode.com /introducing- livecloud/) Visitors in LiveCode (https://livecode.com /visitors-in- livecode/) WordLib Update Released (https://livecode.com /wordlib- update- released/)

 Sitemap (/sitemap)

Terms (/terms) Privacy Policy (/privacy-policy) EULA (/eula)

Brought to you by LiveCode Ltd, Registered in Scotland, No. SC200728