



## **NewtCard Handbook**

**February 6, 1998**

©NS BASIC Corporation, 1998.

77 Hill Crescent

Toronto, Canada M1M 1J3

(416) 264-5999

This manual and the software described in it are copyrighted, with all rights reserved. Under the copyright laws, this manual may not be copied, in whole or in part, without the written consent of NS BASIC Corporation. Under the law, copying includes translating into another language or format.

Every effort has been made to ensure that the information in this manual is accurate. NS BASIC Corporation is not responsible for printing or clerical errors. Specifications are subject to change without notice.

MessagePad, Newton and the Newton logo are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Icons copyrighted by and used with the permission of Apple Computer, Inc. All rights reserved.

Mention of third party products and their trademarks is for informational purposes only and constitutes neither an endorsement or recommendation. NS BASIC Corporation assumes no responsibility with regard to the performance or use of NewtCard or these products.

#### Canadian Cataloguing In Publication Data

Henne, George W.P., 1954-

Schettino, John C. Jr., 1961-

Schettino, Elizabeth O., Ph.D., 1961-

NewtCard Handbook

Includes index.

ISBN XXXXXXXXXX

1. BASIC (Computer program Language). 2. Newton (Computer) - Programming. I. Title.

QA76.8.N48H4 1997                      005.265                      C94-931542-7

# L I C E N S E   A G R E E M E N T

PLEASE READ THIS LICENSE CAREFULLY BEFORE USING THE SOFTWARE. BY USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, PROMPTLY RETURN THE PRODUCT TO THE PLACE WHERE YOU OBTAINED IT AND YOUR MONEY WILL BE REFUNDED.

1. License. The application, demonstration, system, and other software accompanying this License, whether on disk, in read only memory, or on any other media (the "Software"), the related documentation and fonts are licensed to you by NS BASIC Corporation ("NSBC"). You own the media on which the Software and fonts are recorded but NSBC and/or NSBC's Licensor(s) retain title to the Software, related documentation and fonts. This License allows you to use the Software and fonts on a single Newton Product (which, for purposes of this License, shall mean a product bearing Apple's Newton logo), and make one copy of the Software and fonts in machine-readable form for backup purposes only. You must reproduce on such copy the NSBC copyright notice and any other proprietary legends that were on the original copy of the Software and fonts. You may also transfer all your license rights in the Software and fonts, the backup copy of the Software and fonts, the related documentation and a copy of this License to another party, provided the other party reads and agrees to accept the terms and conditions of this License.
2. Restrictions. The Software contains copyrighted material, trade secrets and other proprietary material and in order to protect them you may not decompile, reverse engineer, disassemble or otherwise reduce the Software to a human-perceivable form. You may not modify, network, rent, lease, load, distribute or create derivative works based upon the Software in whole or in part. You may not electronically transmit the Software from one device to another or over a network.
3. Termination. This License is effective until terminated. You may terminate this License at any time by destroying the Software and related documentation and fonts. This License will terminate immediately without notice from NSBC if you fail to comply with any provision of this License. Upon termination you must destroy the Software, related documentation and fonts.
4. Export Law Assurances. You agree and certify that neither the Software nor any other technical data received from NSBC, nor the direct product thereof, will be exported outside the United States except as authorized and as permitted by the laws and regulations of the United States. If the Software has been rightfully obtained by you outside of the United States, you agree that you will not reexport the Software nor any other technical data received from NSBC, nor the direct product thereof, except as permitted by the laws and regulations of the United States and the laws and regulations of the jurisdiction in which you obtained the Software.
5. Government End Users. If you are acquiring the Software and fonts on behalf of any unit or agency of the United States Government, the following provisions apply. The Government agrees: (i) if the Software and fonts are supplied to the Department of Defense (DoD), the Software and fonts are classified as "Commercial Computer Software" and the Government is acquiring only "restricted rights" in the Software, its documentation and fonts as that term is defined in Clause 252.227-7013(c)(1) of the DFARS; and (ii) if the Software and fonts are supplied to any unit or agency of the United States Government other than DoD, the Governments' rights in the Software, its documentation and fonts will be as defined in Clause 52.227-19(c)(2) of the FAR or, in the case of NASA, in Clause 18-52.227-86(d) of the NASA supplement to the FAR.
6. NS BASIC will replace at no charge defective disks or manuals within 90 days of the date of purchase. NS BASIC warrants that the programs will perform generally in compliance with the included documentation. NS BASIC does not warrant that the programs and manuals are free from all bugs, errors or omissions.
7. Disclaimer of Warranty on Software. You expressly acknowledge and agree that use of the Software and fonts is at your sole risk. The Software, related documentation and fonts are provided "AS IS" and without warranty of any kind and NSBC and NSBC's Licensor(s) (for the purposes of provisions 7 and 8, NSBC and NSBC's Licensor(s))

shall be collectively referred to as "NSBC") EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NSBC DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE SOFTWARE AND THE FONTS WILL BE CORRECTED. FURTHERMORE, NSBC DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE AND FONTS OR RELATED DOCUMENTATION IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY NSBC OR A NSBC AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU (AND NOT NSBC OR AN NSBC AUTHORIZED REPRESENTATIVE) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

8. Limitation of Liability. Because software is inherently complex and may not be free from errors, you are advised to verify the work produced by the Program. UNDER NO CIRCUMSTANCES INCLUDING NEGLIGENCE, SHALL NSBC BE LIABLE FOR ANY INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES THAT RESULT FROM THE USE OR INABILITY TO USE THE SOFTWARE OR RELATED DOCUMENTATION, EVEN IF NSBC OR A NSBC AUTHORIZED REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU. In no event shall NSBC's total liability to you for all damages, losses, and causes of action (whether in contract, tort (including negligence) or otherwise) exceed the amount paid by you for the Software and fonts.

9. Allocation of Risk: You acknowledge and agree that this Agreement allocates risk between you and NSBC as authorized by the Uniform Commercial Code and other applicable law and that the pricing of NSBC's products reflects this allocation of risk and the limitations of liability contained in this Agreement. If any remedy hereunder is determined to have failed of its essential purpose, all limitations of liability and exclusions of damages as set forth in this Agreement will remain in effect.

10. Support. NSBC may, at its option, provide support services at its standard fees for such services. Such support services will be governed by the limitations of liability under this Agreement.

11. Additional Restrictions: Any upgrade or enhancement of the program subsequently supplied by NSBC may only be used upon the destruction of the prior version, and shall be governed by the terms of this Agreement.

12. Controlling Law and Severability. This License shall be governed by and construed in accordance with the laws of the United States and the State of Delaware, as applied to agreements entered into and to be performed entirely within Delaware between Delaware residents. If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License shall be enforced to the maximum extent permissible so as to effect the intent of the parties, and the remainder of this License shall continue in full force and effect.

13. Complete Agreement. This License constitutes the entire agreement between the parties with respect to the use of the Software, related documentation and fonts, and supersedes all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by a duly authorized representative of NSBC.

# C O N T E N T S

I. Introduction .....	1
1.1 Handbook Overview.....	1
1.2 About HyperCard and NewtCard.....	2
NewtCard.....	3
NewtCard and the Newton .....	4
1.3 System Requirements .....	4
Newton System Compatibility.....	5
1.4 Installation.....	5
Preparing to Install on the Newton.....	5
Preparing to Install on a Storage Card.....	6
Installing The NewtCard Package .....	7
Installing Additional Packages.....	8
Entering Your Registration Number .....	9
1.5 Conventions Used in this Handbook .....	10
2. The Browsing Environment.....	13
2.1 Interacting With NewtCard.....	13
2.2 Building Stacks with NewtCard.....	16
Creating a New Stack .....	16
Saving Changes.....	17
Opening an Existing Stack .....	17
Routing a Stack .....	18
The Home Stack.....	19
2.3 Working With Stacks .....	19
When Arriving.....	20
When Leaving .....	21
2.4 Working with Backgrounds.....	22
Creating Backgrounds.....	22
Background Info .....	23
Background Artwork .....	24
Background Buttons & Fields .....	24
Deleting a Background.....	25
2.5 Working With Cards.....	26
Creating Cards .....	26
Card Info .....	26
Card Artwork.....	27
Card Buttons & Fields.....	28

Deleting a Card .....	28
3. The Drawing Environment .....	29
3.1 The Drawing Tool Palette .....	30
The Pointer Tool .....	31
The Text Tool .....	32
The Line Tool .....	33
The Rectangle Tool .....	33
The Rounded Rectangle Tool .....	34
The Oval Tool .....	34
The Freehand Tool .....	34
The Polygon Tool .....	35
The Stamp Tool .....	35
The Shapes Tool .....	37
The Fill Tool .....	37
The Pen Tool .....	38
3.2 Drawing environment menus .....	38
Editing items .....	38
Setting the Text Font .....	39
Arranging Items .....	39
3.3 Editing with the Pen .....	40
3.4 Exiting and Saving .....	40
4. The Buttons & Fields Environment .....	41
4.1 Setting the User Level .....	42
4.2 Creating Buttons and Fields .....	44
4.3 Editing Buttons and Fields .....	47
Pen Edits .....	48
Using the Edit Slip .....	48
4.4 Changing Order and Deleting .....	52
4.5 Saving Changes and Returning to Browser .....	52
5. Building Stacks Without Scripting .....	53
5.1 Stack Planning .....	53
Layers .....	53
Backgrounds and Cards .....	54
Stack Goals .....	54
5.2 The Recipe Stack .....	55
Making a Title Card .....	56
The Recipe Background .....	59
Using the Stack .....	62
6. Scripting with NewtCard .....	65
6.1 When to Script .....	65

6.2 Where to Script.....	66
6.3 How to Script.....	67
Adding a Script.....	68
Deleting a Script.....	69
Adding a Stack Script.....	70
Adding Button and Field Scripts .....	72
Adding a Background Script .....	75
Getting Ready To Test.....	75
Testing the Stack.....	76
7. NewtCard Utilities .....	77
7.1 Stacks as Packages.....	77
Creating Packages .....	77
Installing Packaged Stacks.....	78
7.2 Sharing Packaged Stacks .....	79
8. NewtCard Reference.....	81
9. Differences between NewtCard and NS BASIC .....	127
9.1 New Features in NewtCard.....	127
Frame References .....	127
Button and Field References .....	128
Button and Field References on Other Cards.....	129
THE and THIS.....	129
Card, Background, and Stack Properties.....	129
Calling Functions in Cards and Fields.....	130
9.2 Unsupported Features of NS BASIC.....	131
9.3 NS BASIC Features that are Changed .....	131
9.4 Button and Field Properties .....	131
INDEX.....	135
USER'S COMMENT FORM .....	137





---

## I. Introduction

Welcome to NewtCard. NewtCard is designed to help you capture, store, and organize all kinds of information. It is a simple yet powerful environment that can be used to create solutions for almost any situation. The name NewtCard is derived from Apple's HyperCard product for the Macintosh. Like HyperCard, NewtCard uses Stacks, Backgrounds, Cards, Buttons, and Fields to organize and present information.

There is a text file named README.TXT on the supplied disk that contains any late-breaking information about NewtCard, including updates to the Handbook. Please read it before installing NewtCard.

Sample Stacks are provided with NewtCard for you to study and use. You can tailor these sample Stacks to your particular needs. There are several packages on the supplied disk in a folder named EXAMPLES that contain the Stacks used in this Handbook.

### I.1 Handbook Overview

NewtCard is easy to learn and use, but there are a lot of features. This handbook is organized so that all of the features of NewtCard are explained before any examples are given. You may want to skip these initial sections and jump right into creating your first Stack on page 53. If

you do you'll need to refer back to the previous sections to learn more about a particular tool or menu option as it is used in the example, since it won't be explained in detail in the example.

If you'd like to get started using NewtCard right away, read the Installation section on page 5, and then turn to page 53 where you'll build your first Stack.

You should be somewhat familiar with the basics of operating a Newton device before you start using this Handbook. That is, you should know about opening applications in the Extras Drawer, using the stylus (pen) and other Newton features. If you are not comfortable with these terms, review the Newton Handbook. Newton devices (MessagePads and eMates) are collectively referred to as "Newtons" throughout this Handbook.

A basic understanding of operating a desktop computer (Macintosh or IBM Compatible) is needed to install the NewtCard software.

## **1.2 About HyperCard and NewtCard**

HyperCard is a wonderful authoring environment for creating information retrieval, database, and other applications on a Macintosh. It remains one of the most popular programming environments for novice and professional programmers, as well as non-programmers.

In HyperCard, information is organized as a Stack of Cards. This is similar to a Stack of flashCards, or index Cards, or a rolodex. It is a simple yet powerful way to store and retrieve information. The "Hyper" part of HyperCard refers to its ability to create hyper-links between different Cards in a Stack.

Although NewtCard is not exactly HyperCard it implements many of the same concepts and can be used for the same purposes as HyperCard. A Newton is very different from a Macintosh and NewtCard reflects those differences.

## NewtCard

Within NewtCard you'll find the same familiar authoring environments as in HyperCard, except tailored to the Newton. Stacks are containers for entire applications or collections of information. Backgrounds contain related groups of Cards and provide a single layout or backdrop for those Cards. Cards hold information much like records in a database. Cards and Backgrounds contain Buttons (things that are used to navigate and interact with the Stack) and Fields (things that store and present information). You can attach scripts to Buttons & Fields, Cards, Backgrounds, and Stacks to create sophisticated applications in NewtCard. Unlike HyperCard, NewtCard uses the NS BASIC programming language within these scripts.

Cards and Stacks are linked via Buttons, much like links on the World-Wide-Web, so that connections between related information are easily navigated.

NewtCard Corporation maintains a World Wide Web page at <http://www.nsbasic.com>. If you have a Web browser check this site for important announcements, technical information, and example NewtCard Stacks. You will also find many example NS BASIC programs that you can adapt for use within NewtCard.

## NewtCard and the Newton

Your Newton is your personal information repository. In other words, it's where you keep your stuff. NewtCard lets you organize and access your information any way you want. With it you can create Stacks that act like databases, store images, or act like other Newton Applications. If you're just getting started you'll be able to create many useful and interesting things with little or no programming. NewtCard uses NS BASIC as its scripting language, so you can create powerful applications that take advantage of the power in your Newton.

NewtCard is always available right on your Newton to capture and refer to information or to create new Stacks, wherever you are. You don't need to learn a complex new language just to take advantage of the powerful features built into your Newton, rather you can use NewtCard's powerful authoring environment and the NS BASIC language to create exactly what you need.

### **I.3 System Requirements**

In order to install NewtCard you need a Newton device running version 2.1 or later of the NewtonOS, a desktop computer (Macintosh or PC Compatible,) the Newton Connection Utilities or another package installer, and a cable that can be used to connect the Newton to the desktop computer.

## Newton System Compatibility


NewtCard version 1.0 is compatible with Newtons running the 2.1 version of the operating system. This includes both the MessagePad 2000, 2100, and eMate 300.

### **1.4 Installation**

NewtCard is supplied on a software disk. You must install it onto your Newton using a package installer. The example shown here uses the Newton Connection Utilities (NCU) to install the NewtCard package. If you're not using NCU then follow the directions in your package installer software manual when installing NewtCard.

NewtCard can be installed on your Newton or on a storage Card.

#### Preparing to Install on the Newton

Before you attempt to install the software on the internal memory of your Newton, check the available memory in your Newton. Open the Extras Drawer and tap . In the list that displays, tap "Memory Info". Verify that the free memory displayed under the name "Internal" is at least 300k. If you have less than this amount, you should remove some information from your Newton or consider installing NewtCard on a storage Card. Refer to your Newton Handbook's Managing Memory section for more information on removing data and packages from your Newton.

If you have a storage Card installed in your Newton, open the Extras Drawer and tap Card.

Verify that the checkbox “Save new info and packages on this Card” is not checked.



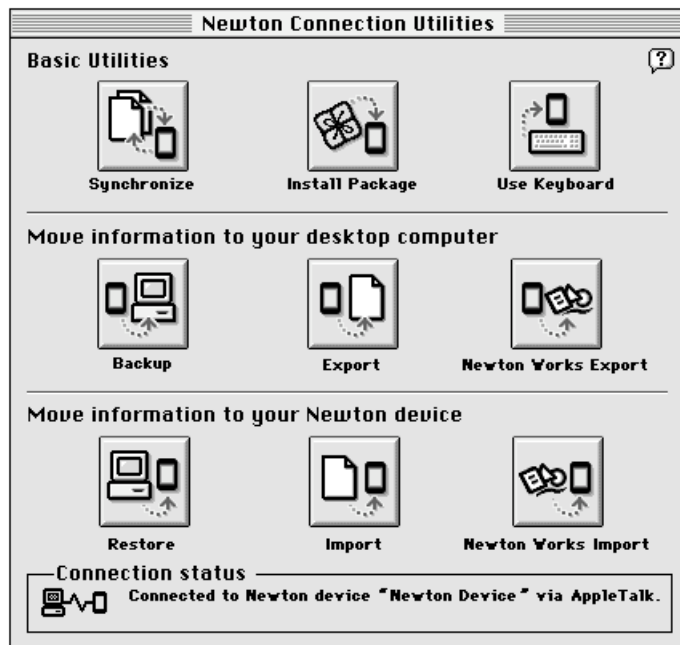
### Preparing to Install on a Storage Card

Before you attempt to install the software, check the available memory on your Card. Open the Extras Drawer and tap “Card”. Verify that the free memory displayed is at least 300K. If you have less than this amount, you should remove some information from your Card or consider installing NewtCard on another storage Card. Refer to your Newton Handbook's Managing Memory section for more information on removing data and packages from your storage Card.

Verify that the checkbox “Save new info and packages on this Card” is checked.

## Installing The NewtCard Package

1 Attach your Newton to your desktop computer with an appropriate cable. Insert the NewtCard disk into your disk drive. Start the Newton Connection Utilities software on your desktop computer.



2 The NCU indicates that it is ready for you to open a connection from your Newton. Open the Extras Drawer (if it is not already open.)

3 Tap "Dock."

**4** Choose the kind of connection you are using. If you are using a “Macintosh LocalTalk” connection select the computer’s name from the list of choices.

**5** Tap “Connect.”

**6** Choose “Install Package” From the NCU Window or the Newton Menu.

The NCU software opens a window on your desktop computer where you can select a package to install. Select “NEWTCARD.PKG” from the disk drive containing the NewtCard disk.

**7** After your connection kit indicates the installation was successful, you’ll see the NewtCard icon in the Extras Drawer.



### Installing Additional Packages

NewtCard includes several other packages that provide additional capabilities. These packages include:

NewtCard Draw (NewtCardDraw.pkg (Mp2K) needs 160K of storage) adds support for the drawing environment for the MP2x00. You must install this package when using NewtCard on a MP2x00.

PackMan (PackMan.PKG needs 80K of storage) adds support for turning Stacks into Newton packages. See page 19 and page 77.



XPort Lite (XPNC.PKG needs 160K of storage) adds support for uploading packaged Stacks to your desktop. See page 79.

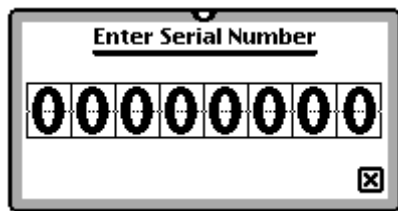
SOUNDS is a directory containing several packages. You can install these packages on your Newton to add additional sounds that may be used in NewtCard.

STACKS is a directory containing several packages. Each package is a packaged Stack that you can install on your Newton. In addition to a sample Home Stack, there are other example Stacks including those used in this handbook.



You may install any of these packages on your Newton or a Card, regardless of where you install the NewtCard package. Follow the same procedures as outlined above to install each package in the desired location. These packages are filed in the “Extensions” folder of the Extras Drawer once they are installed.

### Entering Your Registration Number

The first time you start NewtCard on your Newton, you will be asked to enter your Product Registration Number.



This number is printed on the outside of the back cover of this Handbook or on the bottom of the box, as well as on the Product Registration Form. Enter your serial number into the slip and then tap the close box.

Open the Extras Drawer and tap the NewtCard icon. The initial registration screen displays, along with the on-screen keyboard. Use the keyboard to tap in your Product Registration Number. You may use the  key to make corrections. Once the number is correctly entered, tap the return  key. Your copy of NewtCard is now installed and ready to use.

## I.5 Conventions Used in this Handbook

The following notation conventions are used in this Handbook:

**KEYWORDS** Capital letters indicate NewtCard keywords, symbols, and other text that must be typed exactly as shown. For the purposes of this manual, uppercase text indicates a required part of the Statement syntax. NewtCard is case-insensitive: keywords are accepted with either uppercase letters, lowercase letters, or any mixture of the two. A keyword such as GO may be entered into your Stacks as go, Go, or GO.

*placeholders* Italic text indicates a placeholder for types of information that you must supply. In the following Statement, *destination* is italicized to show that the GO Statement requires a destination:

GO *destination*

In an actual Stack Statement, *destination* must be replaced with a specific destination, such as:

```
GO FIRST
```

**Examples** This Monaco typeface indicates example scripts and information that is printed in the NewtCard Message window. The following example shows a line from a NewtCard script:

```
20 GO CARD "Overview"
```

**User Input** A bold Monaco typeface indicates something entered by the user in response to a NewtCard prompt or in the Scripting environment. It distinguishes between an on-screen prompt and user input when both appear in the same example. For instance, **John** is entered in response to the "Enter Your Name:" prompt:

```
Enter Your Name:  
? John
```

**[Optional]** Brackets indicate that the enclosed items are optional. In the following example, brackets are used to show that using a second item with the GO Statement is optional:

```
GO destination [name]
```

Both of these GO Statements are legal, since GO accepts one or two items:

```
GO LAST  
GO CARD "Instructions"
```

|            The vertical bar indicates that the items are mutually exclusive. In the following example the bar indicates that the RUN command can either be used with a file name or a line number:

RUN [*fileName* | *lineNumber*]

Underlined     Underlined text indicate that the items are environment variables. In the following example, underlining is used to indicate that USERLEVEL is an environment variable:

USERLEVEL is used to control the amount of control the user has within the Browsing environment (see page 42).

## 2. The Browsing Environment

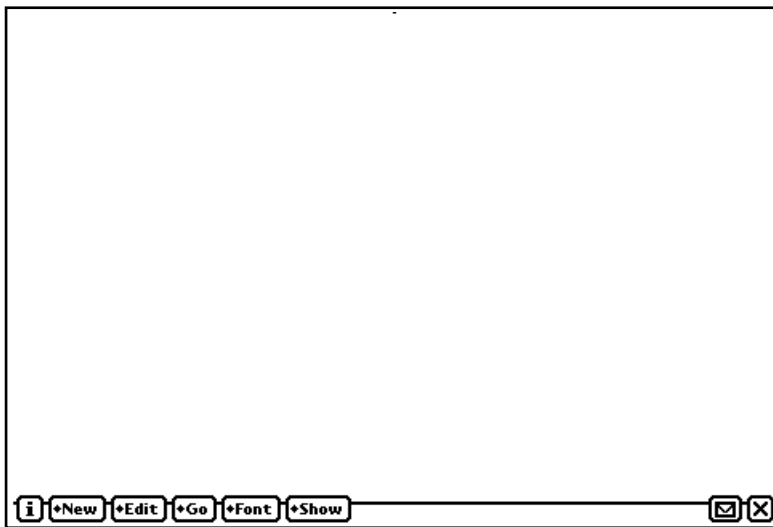
### 2.1 Interacting With NewtCard

NewtCard provides several powerful environments for creating and using Stacks on the Newton. Begin working with NewtCard by opening the Extras Drawer and tapping the NewtCard icon.





**NewtCard**


After briefly displaying an introduction screen the NewtCard Browsing environment is displayed:



The first time you launch NewtCard it creates a new, Untitled Stack unless you've already installed the sample Home Stack. If you have installed the sample Home Stack, that Stack is opened instead. Each time you launch NewtCard it returns you to the same Stack that was showing when you last exited.


The initial view is the Browsing environment. This environment lets you use and navigate within a Stack. The view contains a menu bar across the bottom of the screen. This menu bar can be hidden by pressing  - spacebar. Press it again to show the menu bar.

Use the  menu to create a new Stack, or to make a new Background or Card within the current Stack.

Use the  menu to delete, cut (same as delete only a copy is saved on the clipboard), or copy the current Card to the clipboard, or paste a Card from the clipboard. In addition to these Card actions, the Edit menu contains the undo/redo action, text cut/copy/paste/clear, and the Background edit toggle.

When editing, you can either edit the Card or the Background, depending on the current selection of the Background Edit menu item. Whenever the Background toggle is on, the menu bar displays with a cross hatch pattern underneath to help remind you that you are editing the Background, as shown below.




Use the  menu to navigate within and across Stacks. The Back option returns you to the Card you were viewing before the current Card. You may go Back up to 32 Cards. The Title Card option displays the Card

with the same name as the Stack. There are options for going to the First, Previous, Next, and Last Card in Stack. The Jump to Card option goes to a Card using a Card name or number that you enter. The Jump to Stack option opens a Stack, closing the current Stack. The Find option searches the Cards of the current Stack using Newton Find facility, and the Message option opens a Message Window where you can enter an NS BASIC command or statement that is executed immediately.

Use the **Font** menu to control the default font used in Fields. It contains three sections. The first section lists all the installed font Faces, the second lists the font Sizes, and the third lists the font Styles. Select the desired Face, Size, and Style to use for text you enter.

Use the **Show** menu to switch between the three environments of NewtCard. The Browser option enables navigation. The Drawing option enables the drawing environment on the current Card or Background. The drawing environment is described beginning on page 29. The Buttons & Fields option enables the Buttons & Fields environment on the current Card or Background. The Buttons & Fields environment is described beginning on page 41. The Stack Info option displays the Stack information slip, where you can change the name of the Stack, view the number of Backgrounds and Cards in the Stack, and customize how the Stack behaves. The Background Info option displays a slip that gives similar information and options for the current Background. The Card Info option displays a slip showing information and options for the current Card.

Use  to Print, Fax, or Beam the current Stack. The Duplicate option creates a copy of the Stack with a different name. The Delete option deletes the current Stack. NewtCard then tries to open the Home Stack. If that Stack does not exist it creates or opens the Untitled Stack. The PackIt option creates a package containing the Stack, so you can exchange it with other Newton owners.

We'll go into more detail on each of these menu items in the next section.

## 2.2 Building Stacks with NewtCard

A Stack is a container for Cards, Backgrounds, and scripts. It's a bit like a Newton package in that it acts like a program. The first step in building a Stack is to make a new Stack in NewtCard in the browsing environment.

### Creating a New Stack

Begin by tapping the New menu and selecting the Stack option. The New Stack slip displays.



Enter a name for the Stack. You may use names that include spaces or other special characters, up to 15 characters. Once you've entered a name and pressed return (or tapped OK) the Stack Info slip displays. The



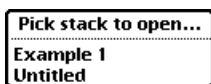
Stack Info slip is covered in the next section. For now just close the slip by tapping the close box. Once the slip closes NewtCard displays the Browsing environment. Whenever you create a new Stack, NewtCard creates a single Background (named “BG\_0”) and a single Card (without a name) in that Background. Since this is a new Stack there are no Drawings, Buttons, or Fields on either the Card or the Background, so even though the Card and Background are displayed you don’t see anything except the menu bar.

## Saving Changes

NewtCard saves all changes you make whenever you change environments, move to a new Card, Background, or Stack, or exit NewtCard. You don’t need to do anything to save changes.

## Opening an Existing Stack


There are two ways to open an existing Stack. If you press the Overview button (on the eMate keyboard or the MessagePad button bar) a list of all Stacks displays in the upper left corner of the screen.




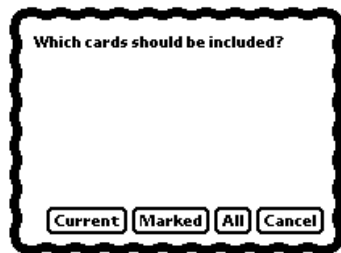
You can also use the Go menu’s Jump to Stack option to open an existing Stack. Selecting it opens a slip where you can choose the Stack from a list or enter the Stack name.

Opening a Stack saves any changes to the current Stack and closes it.


## Routing a Stack


You use the  menu to Print, Fax, Beam, Delete, Duplicate, or Pack the current Stack.


Printing and Faxing Cards: Select the Print or Fax options from the  menu. A notification slip asks you to indicate what to Print or Fax.




Once you've made a selection the standard Newton routing slip displays. The Stack Prints or Faxes to your selected location.

Beaming a Stack: Select the Beam option from the  menu. The Stack beams to the receiving Newton once the infrared connection is established.

Deleting a Stack: Select the Delete option from the  menu. Tap OK to delete the Stack, or the close box to cancel the delete. When you delete a Stack NewtCard always tries to open the Home Stack. If that Stack does not exist it opens the Untitled Stack, creating it if necessary.

Duplicating a Stack: Select the Duplicate option from the  menu. A new copy of the Stack is created with the same name as the current Stack, with “copy” appended to the name. The current Stack remains open.

Packaging a Stack: You can create packages that contain Stacks. This is handy if you want to post your Stack on a World-Wide-Web page for others to download, or you wish to otherwise distribute your Stacks to multiple Newtons. It’s also another way to back up or archive a Stack.

Select the PackIt option from the  menu. A package containing the Stack is created with the same name as the current Stack, with “.stk” appended to the name. The package is placed in the Extras Drawer in the Unfiled folder. Use the XPort Lite utility to copy the package off of the Newton. The current Stack remains open.

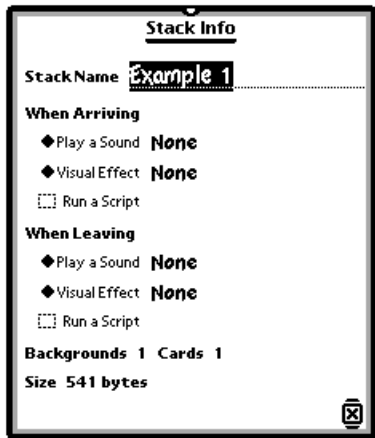
### The Home Stack

If you create a Stack named Home, NewtCard opens that Stack rather than creating or opening the Untitled Stack whenever you Delete a Stack. There is an example Home Stack included on the Disk. To install this Stack install the package named "HOME.STK" onto your Newton using NCU, and then launch the package. This installs the Home Stack onto your Newton. You may delete the package once you’ve run it. You can do this before or after installing NewtCard.

## **2.3 Working With Stacks**

A new Stack contains a single Background and a single Card. The next step in creating a useful Stack is to customize it.

The Stack Info slip displays when you create a new Stack, or by tapping the Stack Info option in the Show menu.



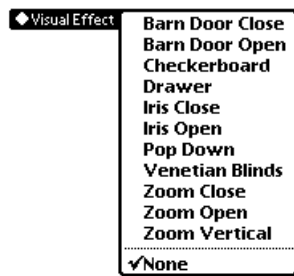
The slip shows the number of Backgrounds and Cards within the Stack, as well as the Stack’s size. Use the Stack Info slip to control how the Stack behaves when it is opened and closed.

### When Arriving

This section determines what NewtCard does whenever this Stack is opened.

The Play a Sound picker determines which sound (if any) is played when the Stack is opened. All sounds installed on your Newton are displayed in this pick list.

The Visual Effect picker determines how the display changes from what was displayed before and the first Card of this Stack.



Each effect's name describes how the display changes from the old Stack to this Stack. If you are unsure what the effect does, try it out!

The Run a Script checkbox enables or disables the NS BASIC script associated with opening a Stack. If you check it, the NS BASIC programming environment displays, with the module associated with the open Script action loaded. The NS BASIC programming environment is covered beginning on page 65.

You may choose to use all three options if you wish. If you do, the sound is played first, the visual effect is performed next, and finally the script is run.

### When Leaving

The same actions are possible when closing the Stack. A Stack is closed whenever a different Stack is opened or you close NewtCard. Choose any sound and visual effect desired, and enable and write a script to run when the Stack closes exactly as you did above.

## 2.4 Working with Backgrounds

Stacks contain one or more Backgrounds. Backgrounds contain Cards. Each Background may contain Drawings, Buttons, and Fields. The contents of the Background are displayed for each Card in that Background.

You can think of each Background as a particular layout for editing records in a database. Then each Card in the Background would act like a record in the database. By using several Backgrounds, it is possible to store many different types of information in a single Stack. You might want to have a home inventory Stack that contained Backgrounds for household items (including a serial number), Music (including Title and Artist) and another for Artwork (including Author, Date, and Media.) Each Background contains specific Buttons & Fields for the kind of information needed. All the Cards in the Stack are related (they all have to do with items in the home) but they don't all store the same information.

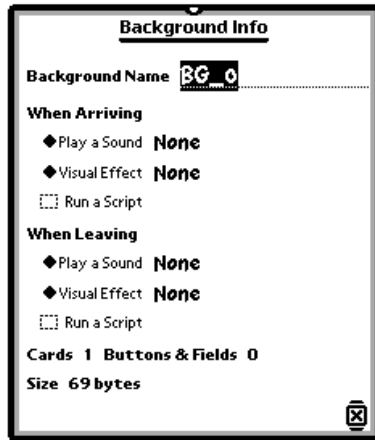
### Creating Backgrounds

Use the Background option of the New menu to create a new Background. Whenever you create a new Background, NewtCard adds a single new Card to that Background. The Background is given a default name ("BG\_0", "BG\_1", etc.) and the Background Info slip displays. You can change the Background name to something more meaningful (such as "Music" or "Artwork",) but try to keep the name short, since you'll probably use it in scripts you write for the Stack. Every Background in a Stack must have a unique name.

## Background Info

Unlike Stacks, there is no way to specifically open a Background. A Background is opened whenever a Card with that Background displays, and the Card's Background is not already the current Background. A Background is closed whenever a Card using a different Background displays or whenever you close NewtCard.

Use the Background Info slip to control the actions performed whenever the current Background is opened or closed. Open the slip by selecting Background Info from the Show menu.



The slip shows the number of Cards, Buttons & Fields in the Background. The When Arriving and When Leaving options are the same as for the Stack Info slip.

## Background Artwork

Each Background may contain artwork or drawings. Artwork includes lines, shapes, and icons you draw in NewtCard, pictures you download to the Newton (via the Web, NCU, or other programs,) and pictures you copy from other packages (such as NewtWorks.) Enable Background editing by selecting Background from the Edit menu, if it is not already selected. Then select the Drawing environment by selecting Drawing from the Show menu. See page 29 for a description of the Drawing environment.

## Background Buttons & Fields

Each Background can contain zero or more Buttons & Fields. These Buttons & Fields display on each Card in the Background. Any information entered into a Background Field is saved with the Card that was displayed when that information was entered. to edit the Buttons & Fields on a Background, enable Background editing by selecting Background from the Edit menu, if it is not already selected. Then select the Buttons & Fields environment by selecting Buttons & Fields from the Show menu. See page 41 for a description of the Buttons & Fields environment.



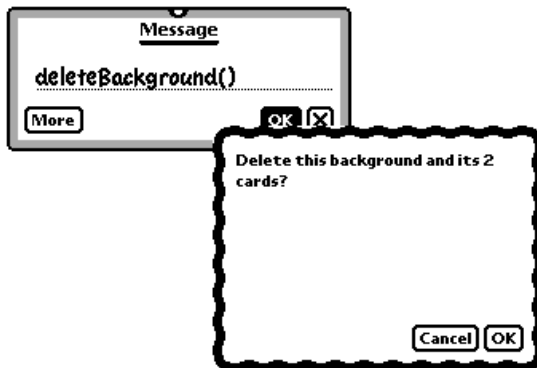
## Deleting a Background

There are two ways to delete the current Background. Whenever you delete the last Card in a Background a confirmation slip displays.



Tap OK to delete the last Card and the Background, or Cancel to leave the last Card and Background in the Stack.

If the Background has many Cards in it by you still wish to delete it, use the `deleteBackground()` function. Open the Message window by selecting the Message option from the Go menu. Enter `deleteBackground()` into the Message window and press return. A confirmation slip displays.



Tap OK to delete all the Cards and the Background, or Cancel to leave all the Cards and the Background in the Stack.

## 2.5 Working With Cards

Cards are similar to records in a database. Like Backgrounds, they can contain artwork as well as Buttons & Fields. They store the information you enter in Background and Card Fields automatically, and recall and display them. A Card exists in a single Background, in a specific Stack.

### Creating Cards

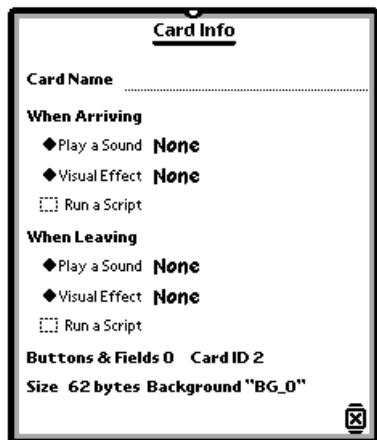
Use the Card option of the New menu to create a new Card. When you create a new Card this way it is not named by default, and the Card Info slip displays. You can change the Card name to something more meaningful but, as with Background names, try to keep the name short since you'll probably use it in scripts. Every Card that has a name must have a unique name. Every Card also has a unique ID number, so you need not use a name to refer to a Card.

A Card with the same name as the Stack is considered the Title Card. Other than being able to navigate to this card using the Go menu Title Card option this Card is the same as any other in the Stack.

### Card Info

A Card is opened whenever it displays, and is closed whenever a different Card displays or you close NewtCard.

Use the Card Info slip to control the actions performed whenever the current Card is opened or closed. Open the slip by selecting Card Info from the Show menu.



The screenshot shows a window titled "Card Info" with a close button in the top right corner. Inside the window, there is a section for "Card Name" with a dotted line for input. Below this is a section titled "When Arriving" with three options: "Play a Sound" set to "None", "Visual Effect" set to "None", and "Run a Script" with an unchecked checkbox. A similar section titled "When Leaving" follows, also with "None" selected for sound and visual effects, and "Run a Script" unchecked. At the bottom, it displays "Buttons & Fields 0" and "Card ID 2". The very bottom line shows "Size 62 bytes" and "Background 'BG\_0'", with a small icon to the right.

The slip shows the number of Buttons & Fields in the Card, as well as the Background it is in and its ID number. The When Arriving and When Leaving options are the same as for the Stack and Background Info slips.

### Card Artwork

Each Card may also contain drawings or artwork, just like Backgrounds. Card artwork is drawn on top of any Background artwork, Buttons & Fields. Disable Background editing by making sure that Background is not selected in the Edit menu. Then select the Drawing environment by selecting Drawing from the Show menu. See page 29 for a description of the Drawing environment.

## Card Buttons & Fields

Each Card can contain zero or more Buttons & Fields. These Buttons & Fields display only on the current Card. They may overlap the artwork of the Card, as well as artwork, Buttons & Fields on the Background. Any information entered into a Card Field is saved with the Card that was displayed when that information was entered. To edit the Buttons & Fields on a Card, be sure Background editing is not selected in the Edit menu. Then select the Buttons & Fields environment by selecting Buttons & Fields from the Show menu. See page 41 for a description of the Buttons & Fields environment.

## Deleting a Card

There are three ways to delete a Card. The first two are to use the Edit menu's Delete Card or Cut Card options. Use Cut Card if you wish to paste the Card into a different Stack. The third way is to use the deleteCard() function. Open the Message window by selecting the Message option from the Go menu. Enter deleteCard() into the Message window and press return. Recall that deleting the last Card in a Background also deletes that Background, after displaying a confirmation slip (see page 25.) If the Card being deleted is not the last Card in a Background, then no confirmation displays.

---

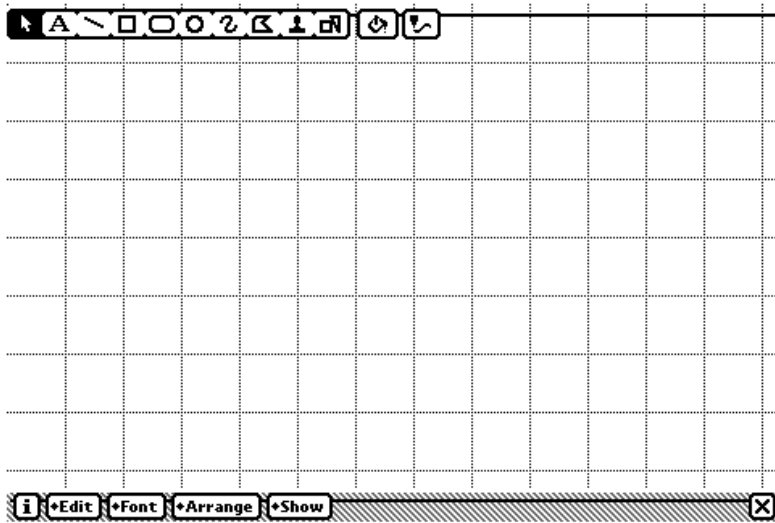
### 3. The Drawing Environment

Stacks are containers for entire applications or collections of information. Stacks can contain several Backgrounds and Backgrounds can contain related groups of Cards. Cards and Backgrounds contain Buttons, Fields and artwork (or drawings). To create the artwork you use the Drawing environment. This creates a layer that displays underneath any Buttons & Fields. Use the Drawing environment to create any artwork that act as the backdrop for the Background or Card.

Artwork consists of images copied from other Newton applications, from your desktop computer, from the World Wide Web, or created within NewtCard. You can use the Drawing environment to create artwork directly or to arrange artwork from other sources.

To place artwork in the Background layer select Background from the Edit menu and then select Drawing from the Show menu.

The NewtCard Drawing environment is shown:



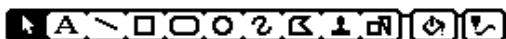
The gray area at the bottom indicates that the artwork is to be placed on the Background layer. The Drawing Tool Palette appears in the upper left hand side of the screen.

### 3.1 The Drawing Tool Palette

The NewtCard Drawing environment allows you to create shapes, lines and artwork that appear on every card for a particular Background. If you want a specific piece of artwork to be seen for every Card in a Background, this is the place to do it. You can also create artwork that appears on a particular Card. The only difference is that instead of selecting Background from the Edit menu, you make sure that Background is disabled and then you select Drawing from the Show

menu. Next go to the Card that you wish to place artwork on. In either case you use the Drawing environment to create and edit drawings or to paste in artwork from the Newton clipboard.

The Drawing Tool palette appears in the upper left hand corner of the screen.



To use a tool, just tap it. We'll look at each tool's function next.

### The Pointer Tool



The Pointer tool allows you to pick out a single item of artwork or a group of artwork pieces. To select a single item tap the item. To select a group of unconnected items press and hold the pen outside one of the desired items. When a sound is heard and a large filled circle appears under the pen encircle the items with the pen. A fat line appears around the items. Each selected item is shown with several selection handles. For lines, the handles appear at the end points. For everything else, the handles appear at the four corners of a bounding rectangle that fully encloses the item. Once one or more items are selected you can perform editing tasks on these selected items such as moving, resizing, or grouping.

To move a selected item, tap any portion of the item that is not covered by a selection handle and drag the item to the new location. To resize an item (or to change the endpoints of a line) drag the desired selection

handle to a new location. As you drag a handle a light-gray rectangle or a light gray version of the shape being resized displays, showing you the effects of the resize as you make it.

## The Text Tool



The Text tool allows you to enter alphabetic letters and numeric digits. To do this select the Text tool and place the pen where you want the letters or numbers to appear. Begin typing. This creates a text item, which is a block of text that can be selected, moved, and rotated. The text word-wraps within the bounds of the rectangle defining its size. You can change the size of the rectangle using the Selection tool to select an existing text item and then drag one of the selection handles.

Use the Font menu to select the desired Font Face, Size, and Style of text for the entire text item before you enter text. You can change the Face, Size, and Style of an existing text item using the Selection tool. Tap a text item to select it, and then make selections from the Font menu. You can also mix several different Font Faces, Sizes, and Styles within a text item by selecting the Text tool, tapping a text item, and then selecting just the portion of the text within that item. Use the Font menu to select the desired Font Face, Size, and Style of the selected text.



## The Line Tool



The Line tool lets you draw straight lines of any angle. To do this, select the Line tool and place the pen where you want the line to begin. Drag the pen to the place that you want the line to end and pick up the pen. A single straight line appears from the starting to the ending points. Use the Pen tool (see page 38) prior to using the Line tool to select the desired thickness and shade of the line drawn.

## The Rectangle Tool



The Rectangle tool draws rectangles when you select it and drag the pen across the screen. To easily create rectangles place the pen in the upper left hand corner and drag to the lower right hand corner for the desired rectangle. To draw an exact square hold down the Shift key while dragging. Use the Fill tool (see page 37) and the Pen tool prior to using the Rectangle tool to select the desired fill color or pattern, and thickness and shade of the rectangle drawn.

## The Rounded Rectangle Tool



The Rounded Rectangle tool draws rectangles with rounded corners. You create a rounded rectangle the same way you create a regular rectangle except that you select the Rounded Rectangle tool from the palette first. Use the Fill tool and Pen tool to control the fill and line style of rounded rectangles.

## The Oval Tool



The Oval tool draws ovals and circles. To create ovals select the tool and then drag the pen across the screen. To create circles hold down the shift key while dragging the pen. Use the Fill tool and Pen tool to control the fill and line styles of ovals.

## The Freehand Tool



The Freehand tool draws freehand lines. Use the Pen tool to control the line style of the lines.

## The Polygon Tool

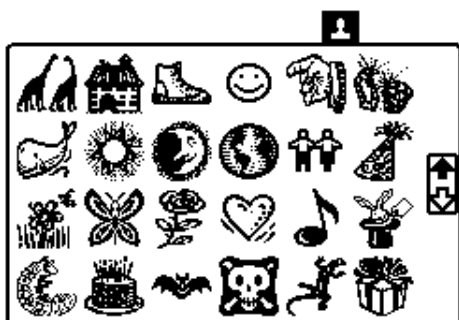
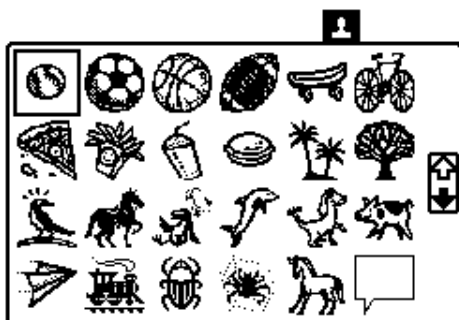


The Polygon tool draws multisided artwork. You tap the pen at several points to create the points of the polygon. Double-tap the last point to complete the polygon. Use the Pen Shapes tool to control the line style of the lines.

## The Stamp Tool



The Stamp tool allows you to select from the following two slips of artwork (Samples shown from an eMate 300):



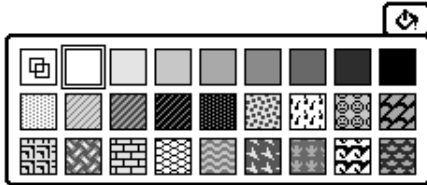
To move from one slip to the other use the up and down arrows on the right side of the slip. To place a stamp, simply select it by tapping. The slip disappears. You then tap the place on the Background or Card where you want the stamp to appear. You may use the selected stamp as many times as you wish.

## The Shapes Tool



The Shapes tool cleans up lines, ovals, triangles, and rectangles you draw freehand. Draw the desired shape and it will be replaced with a straight line, oval, triangle, or rectangle, based on what you draw. Use the Fill tool and Pen tool to control the fill and line style of the shapes you draw.

## The Fill Tool



The Fill tool sets the fill pattern or color for an item. It works on rectangles, ovals, and polygons. To draw an item with a fill pattern, first select the desired tool, then tap the Fill tool, and tap the desired fill pattern or color. Then create the item. It will be created with the selected fill. To change an existing item's fill pattern or color, use the Selection tool to select the item, then use the Fill tool to select the new fill pattern or color.

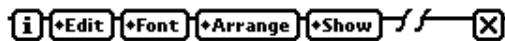
## The Pen Tool



The Pen tool sets the pen shade and size for the lines of an item. It works on lines, rectangles, ovals, polygons, and freehand drawings. To draw an item with a specific pen shade and size pattern, first select the desired tool, then tap the Pen tool, and tap the desired shade and size. Then create the item. It will be created with lines that are the selected shade and size. To change the shade and size of lines in an existing item, use the Selection tool to select the item, then use the Pen tool to select the new pen shade and size.

### **3.2 Drawing environment menus**

The menu bar for the drawing environment contains menus for editing the items in the drawing, setting the font for text items, and for arranging items.



#### Editing items

The Edit menu contains the usual Undo, Cut, Copy, Paste, and Clear options, as well as a Select-All and Duplicate option.

## Setting the Text Font

Use the Font menu to control the font used in text items. It contains three sections. The first section lists all the installed font Faces, the second lists the font Sizes, and the third lists the font Styles. Select the desired Face, Size, and Style to use for text you enter, or to change selected text.

## Arranging Items

As you draw items, each new item is drawn on top of all the items drawn before it. Each item exists separate from all other items.

Use the Arrange menu to move a selected item in front of or behind another item. The Move Forward option moves the selected item closer to the top. The Move to Front option moves it to the very top of all items. The Move Backwards and Move to Back options move an item towards or to the bottom.

You can also use the Arrange menu to align items. Select two or more items and then tap the Align option. A slip displays offering two options. You can align or distribute items horizontally using the options in the Top to Bottom pick list, and align or distribute items vertically using the Left to Right pick list. You can try several different settings. Tapping the close box on the slip accepts the changes, tapping Revert undoes them.

The Arrange menu lets you Flip an object Horizontally or Vertically, as well as Rotate it in 90 degree increments to the Left or Right. Note that you can rotate text items to generate labels that run sideways.

Finally, use the Arrange menu to Group several items into a new group item. Select two or more items and then tap the Group option. The group can now be moved or resized all at once. To split apart a group item select it, then tap Ungroup.

### **3.3 Editing with the Pen**

Use the standard Newton gestures to select multiple items. This includes holding the pen down until the selection noise is heard and then encircling several items, or using the Selection tool and holding the Shift key down while tapping additional items. Delete items using the Scrub gesture. To be sure of which item you're deleting, select it using the Selection tool and then scrub over it.

Items can be cut to the clipboard by dragging them to the edge of the screen. You can paste cut or copied artwork and text by dragging the clipping from the edge of the screen to the desired location.

### **3.4 Exiting and Saving**

Your changes are saved when you switch to the Browser or Buttons & Fields environments. Select the desired environment from the Show menu to leave the Drawing environment.



## 4. The Buttons & Fields Environment

Backgrounds and Cards contain a Buttons & Fields layer. This layer displays on top of any artwork on the Background or Card. You use this Buttons & Fields layer to create the interface and the storage for the Background or Card.

Buttons are Newton interface elements that are used to specifically to perform actions. These include checkboxes, picture, popup, and text buttons, lists of text, and other Buttons. Fields are Newton interface elements that are used to store and display information. These include draw, text, and label Input fields, as well as datePicker, gauge, slider, and other fields. Use the Buttons & Fields environment to create and edit the Buttons & Fields on a Card or Background.

Every Button & Field has several properties that control how and where it displays, as well as how it appears. See page 131 for a complete description of every Button & Field property. In addition to properties, Buttons & Fields can have actions that include moving to a new Card, Stack, or Application, playing a sound, performing a visual effect, and running a script.

Within the Button & Field layer, buttons are displayed in a specific order. This order determines which Button or Field is visible when two more overlap. It also determines the order Buttons & Fields are visited when using the Tab key to move within a Card or Background.


## 4.1 Setting the User Level

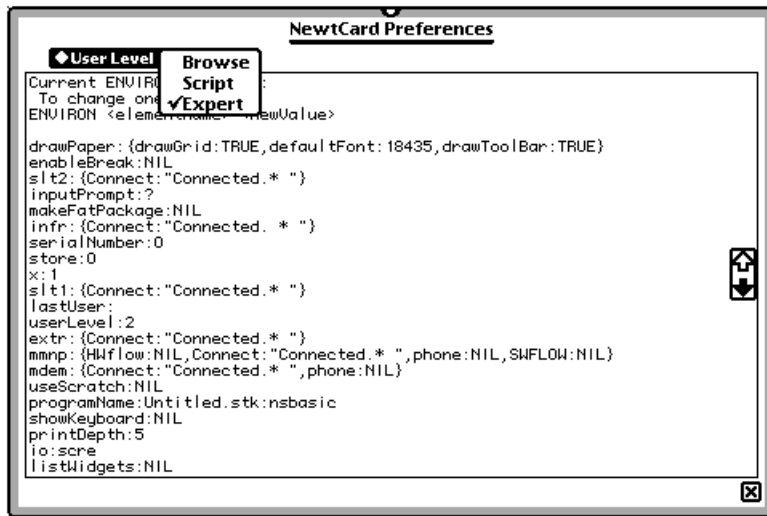
NewtCard has three user levels called Browse, Script, and Expert. These user levels reduce or expand the fields available in the Buttons & Fields environment.

The Browse user level removes all options for creating new Buttons and Fields, although you can still edit and delete existing fields in a Background or Card.

The Script user level only allows creation of the most commonly used Buttons and Fields.

The Expert user level allow creation of all Buttons and Fields.

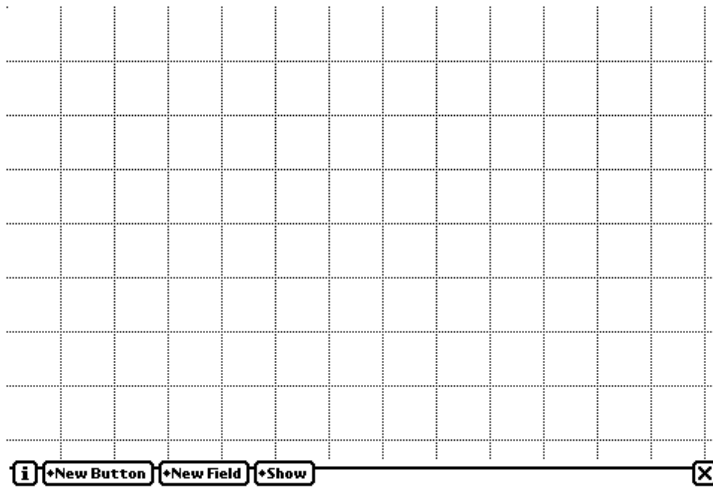
You set the user level using the Prefs option of the  menu. The Prefs slip displays. This slip displays all of the NewtCard settings, and includes a User Level picker where you can select the new user level.



If the user level is set to Browse then the ENVIRON settings are not displayed. The ENVIRON settings are described in the NS BASIC Handbook.

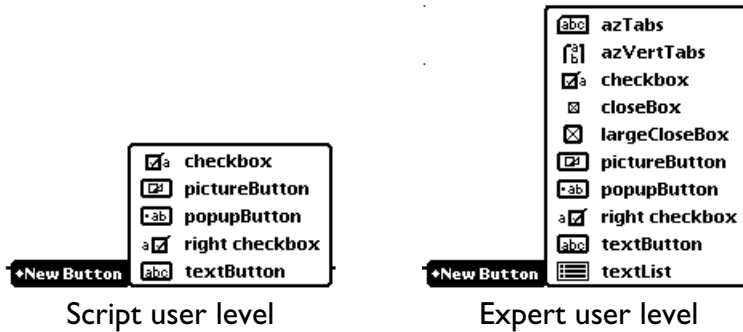
## 4.2 Creating Buttons and Fields

Select Buttons & Fields from the Show menu to use the Buttons & Fields environment. If you want to create elements on the background, remember to enable the Background option from the Edit menu prior to selecting the Buttons & Fields environment. The Buttons & Fields environment for a Card is shown below.



You can both create and edit Buttons & Fields in this environment. For the most part, what you see in the Buttons & Fields environment is what you get when you browse the Stack. The grid of dots does not display in your application. They are shown to help you align Buttons & Fields. Also, the menu bar appears in the same place as the browser menu for the Stack. Avoid placing Buttons or Fields in the menu bar area.

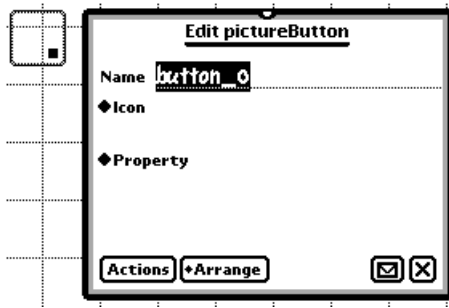
Let's add some Buttons & Fields to this Card. You create a new Button by tapping New Button and then selecting the desired Button from the menu:



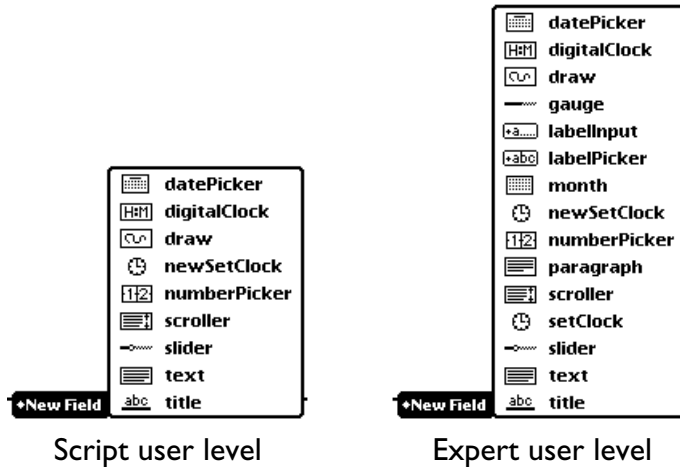
The New Button menu shows a small icon for each button, representing the style of Button to create. A complete description of the different Buttons is provided in the NewtCard Reference section.

Select `pictureButton` from the menu. A new Button is added to the Card. A new Buttons or Field is selected automatically, which means it displays with a dotted rectangle surrounding it. This rectangle represents the size of the of the Button or Field. The little black square in the lower right corner is called a `resize handle`, and is only displayed when it is selected.


The Edit slip for the Button displays when it is created.



You create new Fields using the New Field menu. It also shows a small icon representing the Field that is created by each menu option.

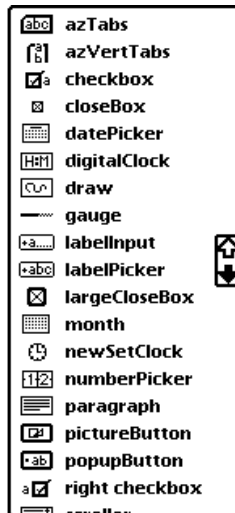


Selecting a Field from this menu adds a new field to the Card or Background, and opens its Edit slip. When you create new Buttons & Fields using these two menus, the new Button or Field is created in a default location.

A third way to create a new Button or Field is to use the  insert gesture. This pops open a scrolling menu showing all the Buttons & Fields.



Script user level



Expert user level

Select the desired Button or Field and it is created with its upper left corner located where you made the insertion gesture.

## 4.3 Editing Buttons and Fields

You edit Buttons & Fields by using the pen or by changing property values using the Edit slip.

## Pen Edits

Select a Button or Field by tapping it with the pen. Delete a Button or Field by using the standard Newton scrub-out gesture. Move a selected Button or Field to a new location on the screen by placing the pen anywhere inside the selection outline and dragging. Resize a Button or Field from the lower right corner by placing the pen on the resize handle and dragging it. Just the handle moves to indicate the new lower right corner for the Button or Field. Lift the pen and the new size is used.

Since both moving and resizing a Button or Field is just changing its `viewBounds` property, you can control the exact placement by using the Edit slip to edit the `viewBounds` property. In fact, you can perform all these edits and more by using the Edit slip.

## Using the Edit Slip

Use the Edit slip to edit the Button or Field name, properties, and Actions, as well as to arrange it within the Buttons & Fields layer or delete it. As you make changes to a Button or Field by using the Edit slip its display is updated in the Buttons & Fields environment automatically after one second of inactivity.

## **Editing the Name**

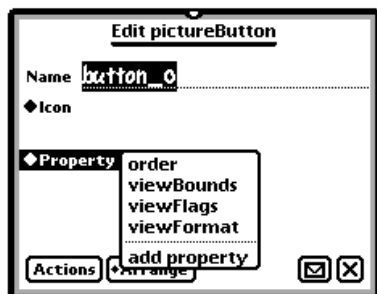
The Button or Field name can be changed to better describe its function. Names must begin with a letter, may contain underscores (`_`), hyphens (`-`), and numbers, and cannot have any spaces. By default each new Button or Field is named `button_X` or `field_X`, where `X` is the number of the Button or Field. The Button or Field name is used in scripts when



you wish to change or access specific properties of a Button or Field. If you don't need to access properties from a script then you can leave the default name as is. If not, you should give each Button & Field a descriptive name, using the same rules for names as for variables. See "Scripting with NewtCard" for examples of accessing property values from within a script.

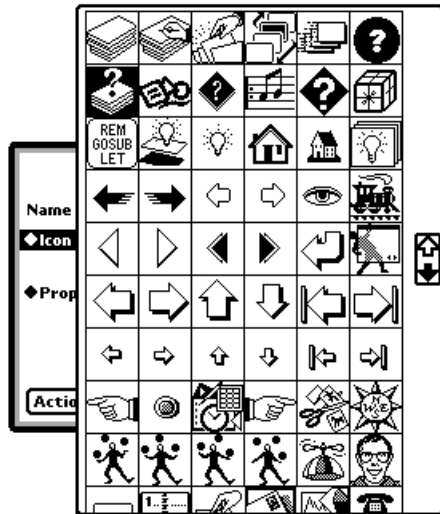
## Editing Properties

Use the property picker in the Edit slip to edit any property of a Button or Field. All the properties are described beginning on page 131. In addition, each Button & Field reference page lists the properties it supports. Fortunately, the Buttons & Fields environment knows the most common properties for a given Button or Field. When you select a Button or Field its Edit slip displays, customized for the particular type of Button or Field. Tap Property in the Edit slip to display the list of available properties, and then tap a property to edit it.



The Edit slip displays one or more fields or checkboxes to edit the property. For example, the viewBounds property editor shows four text entry boxes for the left, top, right, and bottom viewBounds values. The pictureButton Button also includes an Icon picker, so you can select the desired icon for the Button.

Tap it and all the possible icons for the Button are displayed.



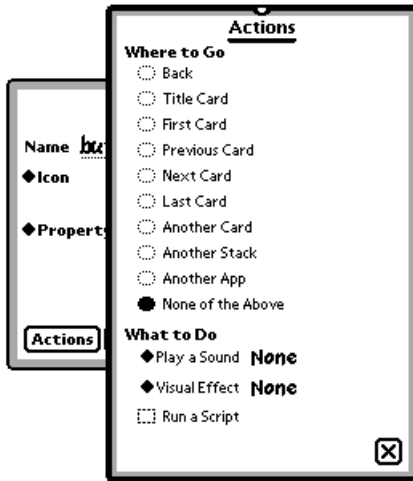
Use the scroll arrows to see the rest of the icons. Tap one of the icons to use that icon for the pictureButton.

## Editing Actions

Every Button & Field can perform a number of Actions. These Actions occur whenever a Button is tapped, and whenever a Field's value changes. Note that for the draw and text Fields this happens each time

you add, edit, or delete a drawing or word in the Field. If you're using the keyboard to enter text into a text field, the action is triggered after every keystroke!

The Action button opens the Actions slip.




Use the choices in Where to Go to navigate to different Cards, Stacks, or Applications. This is usually used for Buttons. For example, you can create Previous, Next, Back, and Home Buttons and use their Where to Go Actions to link them to the appropriate Cards and Stacks. Recall that the Title Card is the Card with the same name as the Stack. The Another Card, Stack, and App choices open a slip where you can enter the Card name or ID, Stack name, or Application name.

Use the choices in What to Do to play a sound, perform a visual effect, or run a script exactly as you do for a Stack, Background, or Card.

## 4.4 Changing Order and Deleting

A Button or Field can be moved forwards or backwards in the front-to-back ordering by using the Arrange menu. Moving it forward draws it before other Buttons & Fields, moving it backward draws it after others. This is important when two or more Buttons or Fields overlap.

The Edit slip includes a  button. Use this to duplicate or delete the Button or Field.

## 4.5 Saving Changes and Returning to Browser

Your changes are saved when you switch to the Browser or Drawing environments. Either tap the Close Box or select the desired environment from the Show menu to leave the Buttons & Fields environment.

## 5. Building Stacks Without Scripting

NewtCard lets you do quite a bit without writing a single script. This section presents an example Recipe Stack that you can create in about 15 minutes without any scripts. The next section builds a Stack that uses several scripts.

### 5.1 Stack Planning

It pays to think about your Stacks for a little while before you start creating. Since each Card in a NewtCard Stack has several different layers, it's important to understand how the layers interact when displaying each Card in the Stack.

#### Layers

When NewtCard displays a Card, it actually shows four separate layers one on top of the other. Anything in a layer on top of another layer covers everything below it. The layers are Card Buttons & Fields, Card Drawing, Background Buttons & Fields, and Background Drawing.

## Backgrounds and Cards

Stacks contain Backgrounds. Backgrounds contain Cards. One way to think of a Stack is like a stack of index cards. If all the cards were white 3x5s, then it would be like a Stack with a single Background. If the stack of index cards contained cards of several different colors, then that would be like a Stack with several Backgrounds.

A Background provides both a Drawing layer and a set of Buttons & Fields that are displayed for each Card created in it. Background fields act like a form - each Card holds whatever information is entered into the Background fields.

## Stack Goals

Depending on what you want your stack to do, you'll need to use Backgrounds and Cards differently. For a Database of information, where one or more forms are all you need, create a Background for each form. On that Background draw whatever artwork you'd like for the form, and then add Buttons & Fields to provide the form itself. Each Card you add in a particular Background will then act as one record for the given form. These Cards will not need anything in their Drawing or Buttons & Fields layers. If your Stack contains more than one Background (i.e., several forms) you can add a Title Card that can quickly navigate to each Background. This Title Card will need its own Background, and will contain several Card Buttons and perhaps some Drawings.

A very different kind of Stack is one where each Card contains its own Drawings. When used this way, a Stack can act like a Scrapbook where you paste Drawing on each Card. You might still want to provide some Buttons and Drawings on the Background, to make using the Stack easier.

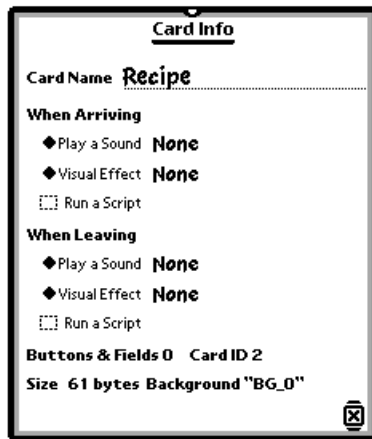
A third kind of Stack is more like a regular Newton package. This type of Stack uses each Background as a different view or layout for the package, and includes quite a few NS BASIC scripts to perform different actions. For example, you could create a Financial Calculator Stack.

## **5.2 The Recipe Stack**

In this section you'll create a simple Stack containing two Backgrounds and one Title Card. Launch NewtCard if it's not already running. Make sure you're in the Browser by tapping Show, Browser. Next tap the New menu and select the Stack option. Enter the name Recipe and press return. The Stack Info slip opens. Just close this slip by tapping the Close Box.

## Making a Title Card

Whenever you create a new Stack, NewtCard makes a new Background and a new Card in that Background. Let's use this Background and Card as our Title Card for the Stack. Open the Card Info slip by tapping Show and selecting the Card Info option. Enter Recipe in the Card Name field and close the slip.



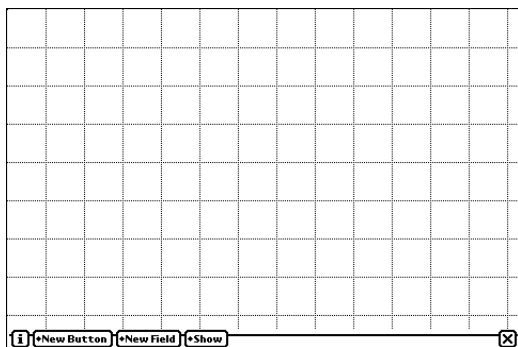
Next add some Buttons and Artwork or Drawings to this Card.

## **Adding Buttons**

We'll add a Button to link this Card to the Home Stack, and A Button linking it to the first Recipe Card in the Stack. Switch to the Buttons & Fields environment for the Card. Make sure you're not editing the Background by tapping the Edit menu. The Background option should not be checked. If it is, tap it, if it is not then just tap anywhere outside

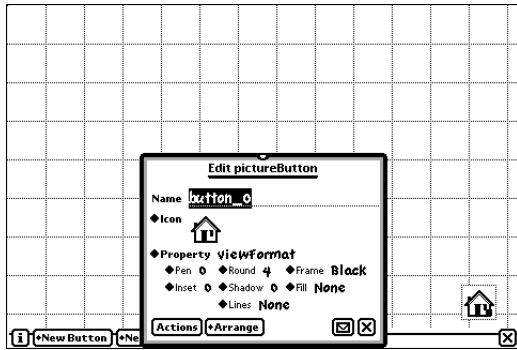


the menu to close it. Now open the Buttons & Fields environment by selecting that option from the Show menu. The Buttons & Fields environment displays.



Create a new pictureButton by tapping New Button and selecting the pictureButton option from the list. A new Button is created in the upper left corner of the screen, and the Button Edit slip displays. Tap the Icon picker and select the Home icon from the popup. Tap the Property picker and select the viewFormat property. Tap the Pen picker and select 0. This turns off the border around the pictureButton. Tap the Actions button to open the Actions slip. Tap the Another Stack radio button under Where to Go, and enter Home in the Stack Name field of the Jump to Stack slip that opens. Close the slip. Tap the Close Box on

the Actions slip. Drag the pictureButton to the lower right corner of the screen. When you're done dragging the Button your screen should look something like this:

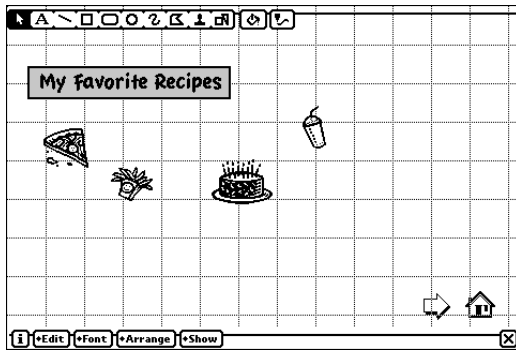


Next, create another pictureButton that will link to the first Recipe in the Stack. Tap New Button and again select pictureButton from the list. This time, select the forward arrow icon for the Button. Select the viewFormat property and set the Pen to 0. Tap the Actions button, and in the Actions slip tap the Next Card radio button under Where to Go. Close the Actions slip, and drag the Button so that it is a bit to the left of the Home Button. Close the pictureButton Edit slip.

To test the Stack, switch to the Browser environment by selecting Browser from the Show menu. Tap the Home Button. The Home Stack opens. As long as you're here, you could add a Button to the Home Stack that links to the Recipe Stack, using the same approach you did to create the Home button. Open the Recipe Stack again by pressing the Overview key, and selecting Recipe from the list.

## Adding Drawings

Switch to the Drawing environment by tapping Show and selecting the Drawing option. Add some text and some drawings to the Card to spice it up a bit! You can draw whatever you want, or copy this example:



## The Recipe Background

Next, we'll create a new Background that will be used by all the Recipe Cards in the Stack. Switch to the Browser environment, and create a new Background by tapping New and selecting the Background option. Notice that this enables Background editing for you. In the Background Info slip that displays select the Trill sound and the Zoom Open visual effect under When Arriving. Close the slip.

## Adding Buttons

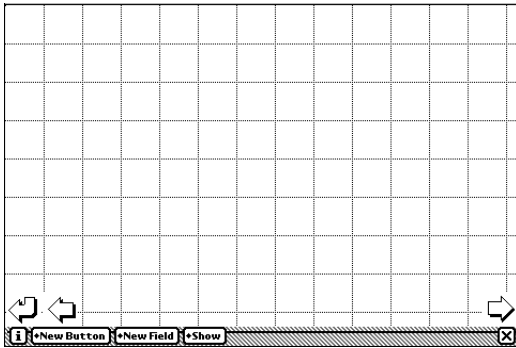
Switch to the Buttons & Fields environment. Add three Buttons to this background to make moving around in the Recipe Stack easier. They will be just like the picture Buttons on the Title Card, so we won't cover their creation step by step.

Create a pictureButton with a return arrow icon and a Pen of 0 in its viewFormat property. In the Actions slip for the Button select the Title Card radio button under Where to Go. Drag the Button to the lower left corner of the screen.

Create another pictureButton with a previous arrow icon and a Pen of 0 in its viewFormat property. In the Actions slip for the Button select the Previous Card radio button under Where to Go. Drag the Button to the lower left corner of the screen, just to the right of the other Button.

Create another pictureButton with a next arrow icon and a Pen of 0 in its viewFormat property. In the Actions slip for the Button select the Next Card radio button. Drag the Button to the lower right corner of the screen.

When you're all done your screen should look something like this:



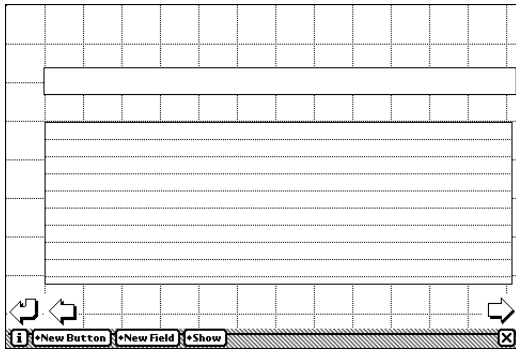
## Adding Fields

Next, add the fields that will hold the recipe information on each Card. We'll use one Text Field to hold the name of the recipe, and another for the recipe itself.

Create a new Text Field by tapping New Field and selecting the Text option. Change the viewFont property in the Edit slip. Select the Casual Family, Bold Face, and 18 Size. Resize the new Field by dragging the resize handle. You want to make the field wide and only about one line tall. Change the viewFormat property to get rid of the lines in the Field. Tap Lines and select None. To make a bit of space between the box and the text inside it tap Inset and select 2. Drag the field so it is about a half inch from the top and left edge of the screen. Select the Text property and delete the default value of Text Box.

Create another Text Field, and change its viewFont property in the Edit slip. Select the Espy Family, Normal Face, and 12 Size. Drag the field so it is about a quarter inch below the other one. Resize the new Field by dragging the resize handle. You want to make the field about as wide as the other one, and as tall as possible without covering the Buttons at the bottom of the screen. You may need to drag the Edit slip out of the way before you resize the Text Field. Select the Text property and delete the default value of Text Box.

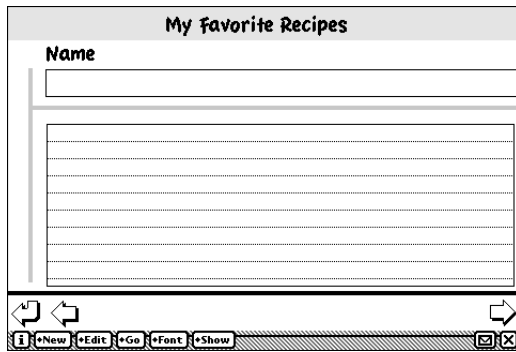
When you're all done your screen should look something like this:



## Adding Drawings

As you did with the Title Card, you can spice up this Background with some drawings. Add some lines and boxes to separate the Buttons and Text Fields, and some Text to label the Fields.

Here is an example that uses three lines, a rectangle, and some text:



## Using the Stack

Switch to the Browsing environment. Turn off Background Editing. Tap the return arrow Button and the Title Card displays. Tap the next arrow Button and you're back to the first recipe Card. Add information into this Card. To make a new Card tap New and select Card. Close the Card slip. A new, blank recipe card displays. Enter another recipe here. You can add as many recipes as you'd like. Use the next and previous arrows to move among the recipe Cards. You can also use the Find key to search for a recipe.

Not bad for a few minutes work! The Recipe Stack is a good example of a simple database in NewtCard. There's a lot more you can do, even with this Stack. If you have a couple very favorite recipes you could name the Cards they are on, and make links to them on the Title Card using pictureButtons.





---

## 6. Scripting with NewtCard

NewtCard is a powerful programming environment. In addition to simple Stacks without any scripts, you can use NewtCard to create applications that perform complex tasks. To do this, you add scripts to your Stack that consist of NS BASIC statements.

### 6.1 When to Script

There are several things you'll probably want to do with Buttons that are not possible just by using the Where to Go radio buttons in the Actions slip. In addition to performing more complex navigation, you might want to provide Buttons that create new Cards, delete Cards, or initiate a Find operation. These are all simple scripts to add, and are a good place to get started with scripting.

Another time scripts are used is to create computed Fields in a form. These Fields display information that is generated by combining other Fields, or by calculating a value from other Fields. A variation on these scripts are input validation scripts. These scripts check the values entered into Fields to make sure they fall within acceptable ranges. For example you may have a Text Field that is used to enter a quantity of an item. This Field can be checked by a script to make sure that the value entered is a number, and is within a specific range.

You also need to use scripts when you create custom applications with NewtCard. These applications use NewtCard to present an interface and to gather and store information, but they contain scripts that perform application-specific functions. One example of this is a database Script that generates reports summarizing the information in the database to create one or more reports. Another is an order-entry Stack that would use scripts to look up an item in an inventory Stack, retrieve the cost of the item, update the quantity on hand, and generate an invoice for an order.

Clearly there are many different reasons to create scripts.

## 6.2 Where to Script

Scripts can be attached to Buttons & Fields, Cards, Backgrounds, and Stacks.

For Buttons & Fields a script is run whenever the Button or Field's actions are triggered. This happens whenever a Button is tapped, and whenever the value in a Field is changed by the user. A large portion of the scripts you'll write in NewtCard will be for Buttons & Fields. Buttons are the best way to initiate actions. You might have a Button that generates a report, looks up an item in another Stack, opens a Newton Application, or performs some other action. Fields may be used to compute values in other Fields, or they may need to have their values validated whenever they change.

Cards, Backgrounds, and Stacks can run scripts whenever they are opened (When Arriving) or closed (When Leaving).

This leads to a very natural approach to scripting. Use the Stack's When Arriving script to perform initialization actions for the Stack. This might include going to a particular Card, setting some variables to initial values, or opening one or more files. Use the Stack's When Leaving script to perform clean up actions for the Stack, such as closing files.

If you use Backgrounds as different layouts for your Stack, you might need to perform similar initialization and clean up actions When Arriving and When Leaving a Background.

Cards can also perform specific functions in a NewtCard Stack. For example, the Title Card might need to perform specific operations whenever it is opened or closed.

Once a Card displays, the Button and Field scripts are used for navigation, data access and generation, and to initiate application-specific functions.

This all sounds more complex than it often is. You can add just a few scripts to a Stack to round out the built-in actions. You can also create full-fledged Newton applications in NewtCard.

## **6.3 How to Script**

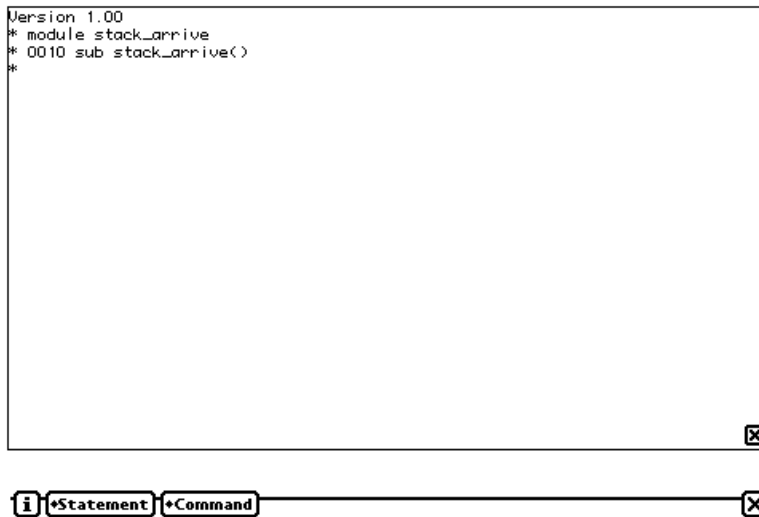
A NewtCard script is written using NS BASIC. This section is not intended to teach you NS BASIC. You should consult the NS BASIC handbook to learn how to program in NS BASIC.

## Adding a Script

Let's make a new Stack that uses Scripts. A Flash Card Stack is a great way to help practice basic math skills. We'll create a Stack that has a single math problem on each Card. When an answer is entered on the Card it is checked for correctness, and if it is correct the next Card is displayed. If it is not a sound is played.

The Stack will have two modes. A setup mode allows the user to create new Cards containing problems. A test mode lets the user try and answer each question. We'll add a When Arriving script to the Stack that initializes a Stack variable to let us know we're in test mode.

Create a new Stack. Name the Stack FlashCards. When the Stack Info slip appears tap the Run a Script Checkbox in the When Arriving section. The Scripting environment displays.



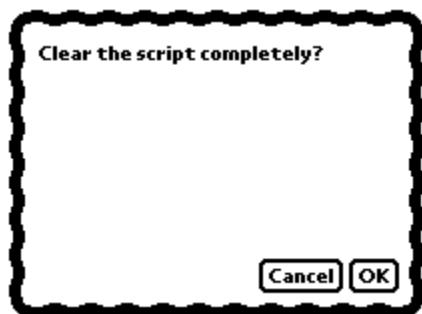
We'll come back to this in just a moment. Tap the small Close Box to leave the Scripting environment. Notice that Run a Script is checked in the Stack Info slip. If you tap it again, you'll re-enter the Scripting environment. Do that now.

### Deleting a Script

Even though you did not add any statements to the script before, you did create and attach a script. The only way to delete an attached script (even an empty one) is to open it in the Scripting environment. Once in the environment, enter the NEW command at the prompt:

```
* module stack_arrive  
* NEW
```

A Notification displays:



Tap OK to delete the script. The Scripting environment closes, and the Stack Info slip (with Run a Script unchecked) displays.

## Adding a Stack Script

Tap the Run a Script Checkbox in the When Arriving section of the Stack Info slip, and the Scripting environment is again displayed.

NewtCard enters two lines that create the correct script for this action:

```
* module stack_arrive
* 0010 sub stack_arrive()
*
```

The MODULE command identifies the specific module you're editing.

When the Stack is opened the subroutine named `stack_arrive()` is called, so NewtCard creates the subroutine definition for you. The final `*` is where you begin to enter the statements of your script. Whenever the Stack is opened we need to move to the Title Card and set a stack variable. Enter the following lines into the NewtCard Scripting environment:

```
20 GO TITLE
30 LET stack.setupmode=CARD BUTTON setupMode
40 END SUB
```

Those four lines do all the work. Line 20 displays the Title Card. Line 30 sets a Stack variable named `setupmode` to the value (`TRUE` or `NIL`) of the Checkbox named `setupMode` on the Title Card. If `setupmode` is `NIL` we know we're in test mode. If it is `TRUE` we know we're in setup mode. Close the Scripting Environment. Tap Show, Card Info and change the name of the first card to FlashCards. That makes the first card the Title Card.

Switch to the Buttons & Fields environment and add a pictureButton. Tap Actions in the Edit slip, and tap Next Card under Where to Go. Close the Actions slip. Add a checkbox Button and name it `setupMode`. Tap Actions in the Edit slip. Tap Run a Script, and enter the following script into the Scripting environment:

```
20 LET stack.setupmode=CARD BUTTON setupMode
30 END SUB
```

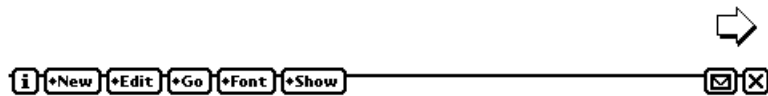
This updates the Stack variable `setupmode` whenever the user taps the Checkbox.

Close the Scripting environment, and then close the Action slip and Edit slip. Close the Buttons & Fields environment. Switch to the Drawing Environment and add some text to the Title Card (for now, you can just put the phrase “Flash Cards!” somewhere on the Card).

When you're all done the Title Card should look something like this:

Flash Cards!

☐ Setup Mode



Tap Show, Browser. Now tap New, Background. This new Background is where we'll add the Flash Card Cards. Name the Background theFlashCards.

### Adding Button and Field Scripts

Each Flash Card contains five buttons and fields. A Text Field displays the problem. A Text Field accepts the answer. A textButton is used to get the answer if the user is stuck, and a pictureButton skips on to the next Card without entering an answer. The last item is a Text Field that holds the correct answer. We'll hide this field when in test mode.



Enter the Buttons & Fields environment and create these five items in the Background. When you create the problem Text Field, set it's viewJustify property so that the Horizontal value is Right. Name the answer Text Field theAnswer. Name the correct answer Text Field theCorrectAnswer. You may arrange them any way you wish, or copy the layout shown below:

Text Box

Text Box

Help

Text Box



We need to add a script to one field and one Button. Tap the answer Text Field, then Tap Actions in the Edit slip.

Tap Run a Script, and enter the following script:

```
20 IF NOT stack.setupmode THEN
30 IF BACKGROUND FIELD theCorrectAnswer = BACKGROUND
FIELD theAnswer THEN
40 GO NEXT
50 ELSE
60 BEEP 3
70 END IF
80 END IF
90 END SUB
```

This tests the entered value against the correct value each time the user changes it, when not in setup mode. If it's correct the next card is displayed. Otherwise a crumple sound indicates the value entered was incorrect.

Close the Scripting environment, and then close the Action slip. Select the Text Button then Tap Actions in the Edit slip. Tap Run a Script, and enter the following script into the Scripting environment:

```
20 NOTIFY (4, "Hint", "The Answer is: " & BACKGROUND
FIELD theCorrectAnswer)
30 END SUB
```

This displays a notification slip with the correct answer whenever the Button is tapped.

Close the Scripting environment and the Actions slip. Select the pictureButton. Tap Actions in the Edit slip, and tap Next Card under Where to Go. Close the Actions slip. This lets the user skip a problem by tapping the pictureButton.

## Adding a Background Script

Next we need to add a Script to the Background. Select Show, Background Info and tap the Run a Script Checkbox in the When Arriving section. Enter the following Script:

```
20 IF stack.setupmode THEN
30   SHOW BACKGROUND FIELD theCorrectAnswer
40 ELSE
50   HIDE BACKGROUND FIELD theCorrectAnswer
60 END IF
70 END SUB
```

This shows or hides the correct answer depending on the mode we're in.

## **Getting Ready To Test**

Close the Scripting environment and the Background Info slip. Toggle Background editing off. Switch to the Browser, and then go to the Title Card. Check the Setup Mode checkbox, and then tap the pictureButton. You should see the first flash card. Enter a problem into the problem

Text Field, and then enter the correct answer in the correct answer Text Field. Delete the value in the answer Text Field. When you're done your card should look like this:

$2 * 4 =$		Help
	8	



If you want to add additional flash cards, Select Edit, Copy Card and then Edit, Paste Card. Enter the new problem and answer into the new Card. You may add as many cards as you wish.

## Testing the Stack

Go back to the Title Card, and uncheck the setup mode checkbox. Tap the pictureButton to move to the first flash card. Enter a value into the answer Text Field. Each time you type a digit the value is tested against the correct answer. If the value you enter is correct, the next card is displayed. If not, a crumple sound is played.

Congratulations! You've just created a Stack that uses Scripts.

---


## 7. NewtCard Utilities

NewtCard includes two powerful utilities that let you package and distribute your Stacks to other Newton users. The PackMan package lets you turn your Stacks into Newton Packages, and XPort Lite lets you upload those packages to your desktop computer.

### 7.1 Stacks as Packages

A NewtCard Stack is a collection of Cards, Artwork, Scripts, and Data, stored in a single Newton Soup. A Newton Soup is a lot like a File in a desktop computer. You can see all the Soups on your Newton by looking in the Storage folder of the Extras drawer. If you want to send a Stack to another Newton user, you need to package it first using the PackMan utility. You might also want to create packages of Stacks to back them up on your desktop computer.


#### Creating Packages

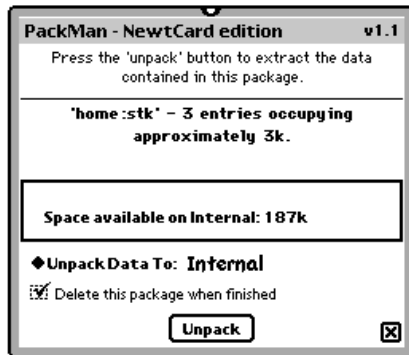
To create a package from a Stack, open the Stack in NewtCard. In the Browsing environment, select the PackIt option from the  menu. A package containing the Stack is created with the same name as the current Stack, with “.stk” appended to the name. The package is placed in the Extras Drawer in the Unfiled folder.

## Installing Packaged Stacks


The NewtCard disk contains several sample Stacks, packaged using PackIt. To install one of these Stacks, or any other Stack packaged with PackIt, you must first install the package on your Newton. See page 5 for an overview of using Newton Connection Utilities to install a package. You will need roughly twice as much free storage space as the size of the package file. Let's install the sample Home Stack from the NewtCard disk.

Install the package named HOME.STK onto your Newton using NCU. Once the package is installed, you only need to launch it to install the

Stack it contains. Open the Extras drawer and locate  in the Unfiled folder. Tap it and the PackMan slip displays:




The slip identifies the Stack, and tells you how much space it will need. Select the desired location (Internal or a Card). You can also indicate that you want to delete the package once the Stack is installed by checking that option in the slip. This is usually a good idea. Once you've


made all your selections tap  and the Stack is installed. If a Stack with the same name already exists a confirmation slip is displayed. Choose to overwrite or cancel the unpack operation. If you enabled package deletion, another confirmation slip displays after the Stack is unpacked asking you to confirm the deletion of the package.

Once a Stack is installed you can open it in NewtCard in the usual way. It is fully usable and editable.

## 7.2 Sharing Packaged Stacks

Once you've created a packaged Stack using PackIt, you can beam it to another Newton or upload it to your desktop computer using XPort Lite.

To beam a packaged Script, open the Extras drawer and select the package you created with PackIt. Then select the Beam Icon option from the  menu. The package will be beamed when another Newton accepts the Beam.

To upload a packaged Script to your desktop computer you must first install the XPort Lite package onto your Newton. Once you've done that, to upload a packaged Stack open the Extras drawer and select the package you created with PackIt. Then select the X-Port option from the  menu. In the routing slip that displays you can choose to X-Port the package now or just place it in the Out Box. Connect your Newton to your desktop computer, run the desktop XPort Lite application, and chose X-Port Now. The package is transferred to your desktop computer.





---

## 8. NewtCard Reference

NewtCard includes a number of Buttons and Fields that you may use in your Stacks. The NewtCard scripting language is based on NS BASIC, but contains several additional Functions and Statements that you may use in your scripts. This chapter contains an entry for each of the Buttons, Fields, Functions, and Statements in NewtCard in alphabetical order. The NS BASIC Handbook contains a complete reference section describing all the Functions and Statements in NS BASIC.

Each entry in the Reference chapter consists of the following information:

<b>Name</b>	<b>Category</b>
-------------	-----------------

---

ITEM parameters

#### **DESCRIPTION**

This section describes the ITEM and its parameters. Details concerning the uses of ITEM are given, as well as any constraints on its use.

#### **EXAMPLE**

A small program that uses ITEM is listed here.

#### **OUTPUT**

This section shows the results of running the Example program.

#### **RELATED ITEMS**

A list of zero or more NewtCard Commands, Statements, Functions, Buttons, or Fields that are related to ITEM. You may often gain a better understanding of ITEM by reviewing the related items.

`ANSWER(prompt, options)`

**DESCRIPTION**

`ANSWER` displays a standard Newton modal dialog slip. The *prompt* string is displayed along with several buttons, depending on the value of *options*. *Options* may be one of:

'okCancel: OK and Cancel buttons are displayed. If the user taps OK TRUE is returned, otherwise NIL is returned.

'yesNo: Yes and No buttons are displayed. If the user taps Yes TRUE is returned, otherwise NIL is returned.

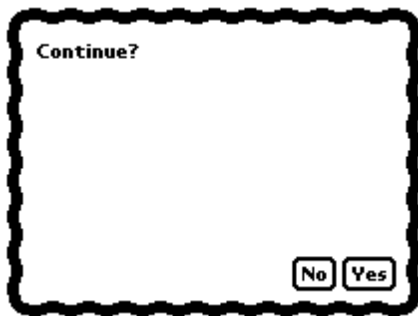
An array of strings: Each string in the array is used as the label for a button. The index of the tapped button is returned.

An array of frames: Each frame contains two fields. The `text` field contains the string to use as the button label. The `value` field contains the value that is returned when the corresponding button is tapped.

**EXAMPLE**

```
theAns = ANSWER("Continue?", 'yesNo)
theAns = ANSWER("Delete File?", 'okCancel)
theAns = ANSWER("Pick one:", ["Blue", "Red", "Green"])
theAns = ANSWER("Pick another:", [{text:"Blue",
value:'blue}, {text:"Red",value:'red}, {text:"Green",
value:'green}]
```

## OUTPUT



## RELATED ITEMS

NOTIFY

**DESCRIPTION**

The AZTABS and AZVERTTABS Buttons display the standard Newton selection tabs in the horizontal and vertical orientations.

These Buttons contain the following properties:

viewBounds: a record specifying the top, left, bottom, and right edge of a frame enclosing the select tab.

curIndex: The currently selected index (from 0) in the selection tab.

**EXAMPLE SCRIPT**

```
20 NOTIFY(4, "Selection Tab Tapped", "Item: " & CARD
BUTTON theButton)
30 END SUB
```

**OUTPUT****RELATED ITEMS**

SHOW, HIDE

**DESCRIPTION**

The CHECKBOX and RIGHTCHECKBOX Buttons display a small square box that shows a check mark when selected. The check box can be toggled by the user by tapping on it or the label. CHECKBOX displays the label followed by the check box, and RIGHTCHECKBOX shows the check box followed by the label.

These Buttons contain the following properties:

viewValue: TRUE to display a checkmark.

text: The label text.

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

To change the value of a Card checkbox in a script, use the following expression:

```
_widgets.myCheckBox.toggleCheck().
```

For a checkbox on the Background, use:

```
_bgcd.background._widgets.myCheckBox.toggleCheck()
```

### EXAMPLE

```
20 IF CARD BUTTON theCheckbox THEN
30   checked = "is checked"
40 ELSE
50   checked = "is not checked"
60 END IF
30 NOTIFY (4, "Checkbox Tapped", "Item: " & checked)
30 END SUB
```

### OUTPUT

☐ Checkbox  
right checkbox ☒

### RELATED ITEMS

SHOW, HIDE, WINDOW

**DESCRIPTION**

The CLOSEBOX and LARGECLOSEBOX widgets display the standard Newton close box (small and large sizes) in the lower right hand corner of the Newton screen.

These Buttons contain the following property:

viewBounds: a record specifying the top, left, bottom, and right edge of a frame enclosing the close box.

**EXAMPLE**

```
20 NOTIFY (4, "Close Box Tapped", "Bye!")
30 END SUB
```

**OUTPUT****RELATED ITEMS**

SHOW, HIDE, WINDOW



DESCRIPTION

The DATEPICKER Field displays the standard Newton date picker. The date or dates can be selected as in the Dates application.

The Field contains the following properties:

selectedDates: An array of integers (from the TIME() function) representing the selected dates. The first date determines which month is displayed. The default value is the current date.

noSelection: TRUE if DATEPCIKER is display-only.

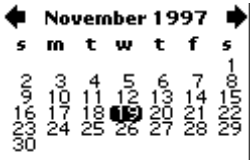
singleDay: TRUE if only a single day may be selected.

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

EXAMPLE

```
20 NOTIFY(4, "Date Selected", DATETIME(CARD FIELD theField[0]))
```

OUTPUT



RELATED ITEMS

HIDE, SHOW, TIME, MONTH

DELET CARD()

**DESCRIPTION**

DELET CARD deletes the current Card from the Stack. If it is the last Card in the current Background then a conformation message is displayed asking if the Background should be deleted. If the user taps No then the Card is not deleted. If the Card is deleted DELET CARD returns TRUE, otherwise it returns NIL.

**EXAMPLE**

DELET CARD()

**OUTPUT**

<The current Card is deleted>

**RELATED ITEMS**

DELET BACKGROUND, NEW CARD

DELETEBACKGROUND()

**DESCRIPTION**

DELETEBACKGROUND deletes the current Background and all its Cards from the Stack. Prior to deleting the Background a conformation message is displayed asking if the Background and all its cards should be deleted. If the user taps Cancel NIL is returned.

**EXAMPLE**

DELETEBACKGROUND()

**OUTPUT**

<The current Card is deleted>

**RELATED ITEMS**

DELETECARD

**DESCRIPTION**

The DIGITALCLOCK Field displays the standard Newton time picker. The time can be selected as in the Dates application.

The Field contains the following property:

**time:** An integer (from the TIME() function) representing the selected time. The initial value of this property determines the initial display. The default value is the current time.

You may also use these properties: viewBounds, viewFlags, viewFont.

**EXAMPLE**

```
20 NOTIFY(4, "Time Changed", "New Time is: " &  
TIMESTR(CARD FIELD theField,0))
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW, TIMESTR, TIME

**DESCRIPTION**

The DRAW Field provides a user entry area that accepts ink drawing. The input may be recognized as shapes (this is the default) by setting the following in the viewFlags property: vVisible, vClickable, vGesturesallowed, and vShapesallowed. To accept just plain ink drawings set Visible, vClickable, vGesturesallowed, and vStrokesallowed

The Field contains the following properties:

viewChildren: An array of records describing the drawing.

You may also use these properties: viewBounds, viewFlags, viewFormat.

**EXAMPLE**

```
20 NOTIFY(4, "Drawing Changed", "")
```

**OUTPUT****RELATED ITEMS**

POINTSTOARRAY, SHOW, HIDE

See the POINTSTOARRAY NS BASIC Reference section entry for an example of extracting the x,y coordinates of the shapes and strokes drawn in a DRAW Field.

FIND *value* [, *foundRecord* [,*fileName*]

**DESCRIPTION**

FIND invokes the standard Newton Find action (i.e., the “find” button) and searches for *value*. What is searched depends on *fileName*. If *fileName* is omitted, the current Stack is searched. If *fileName* is supplied, then that file is searched. If more than one match is found, then the standard Find result slip is displayed so the user may select one of the matching records. If *foundRecord* is supplied, it should be a valid variable name. The matching record will be placed in the variable. When using FIND on the current Stack, the found Card is displayed as the current Card.

**EXAMPLE**

FIND "Tomato"  
Finds all Cards in the current Stack with  
Tomato in any field.

FIND "Smith",myVar,"names"  
Finds an entry in Names with Smith in any  
field, and copies the record to the variable  
myVar.

**RELATED ITEMS**

FUNCTION *functionName*(*args*) *expression*

DEF FN *functionName*(*args*) = *expression*

### DESCRIPTION

FUNCTION and DEF FN define a user function. *FunctionName* is a valid NS BASIC variable name and *expression* is a valid NS BASIC expression or NewtonScript code. *args* are parameter variables that are used in *expression*. User functions retain their values in the same manner as any other variable. Use of functions can greatly speed up your code.

**Note:** To use NewtonScript in *expression*, you'll need a NewtonScript Manual. *Programming for the Newton*, by McKeehan and Rhodes and published by AP Professional is a good source of NewtonScript documentation.

To call a user function, use:

*functionName*(*args*)

### EXAMPLE

```
20 REM FUNCTION Example
30 DEF FNS(starttime)=(TICKS()-starttime)/60
40 FUNCTION tot(b) BEGIN LOCAL x:=0; FOR i:=0 TO
LENGTH(b)-1 DO x:=x+b[i]; x END
50 LET iterations=1000
60 a=ARRAY(iterations, 25)
70 GOSUB 100 //sum using NS BASIC loop
80 GOSUB 180 //sum using function
```

```
90 STOP
100 REM sum using NS BASIC loop
110 tm=TICKS()
120 x=0
130 FOR i=0 TO LENGTH(a)-1
140 x=x+a[i]
150 NEXT i
160 PRINT "Method 1:", fns(tm)
170 RETURN
180 REM sum using function
190 tm=TICKS()
200 x=tot(a)
210 PRINT "Method 2:", fns(tm)
220 RETURN
```

### OUTPUT

```
Method 1: 15
Method 2: 0.1
Stop at 0080
*
```

### RELATED ITEMS



**DESCRIPTION**

The GAUGE Field provides a display of a relative value (i.e., the battery gauge). You can set the initial value of the GAUGE, and update the value within a script.

The Field contains the following properties:

viewValue: The current setting (0-100% filled).

You may also use these properties: viewBounds, viewFlags, viewFormat.

**EXAMPLE**

```
20 SET CARD FIELD theField to 50
```

**OUTPUT****RELATED ITEMS**

SHOW, HIDE, PROGRESS, SLIDER

GETFIELD(*card*, *context*, *field*)

GETFIELD gets the value of a Card or Background Field on a Card. *card* is either a string containing the name of the Card or a number corresponding to the Card ID. *context* is either 'card to set a Card Field or 'background to set a Background Field. *field* is the name of the Field to get. GETFIELD returns the contents of the specified Field if it succeeds, or NIL if the specified Card could not be found. GETFIELD does not display the Card specified by *card*.

You can use the simpler Field reference when getting the value of a Card or Background field of the current Card. See page 128 for a description of Field references.

#### EXAMPLE

```
value1 = GETFIELD(2, 'background, 'myField)
value2 = GETFIELD("Summary", 'card, 'myTotal)
```

#### OUTPUT

*value1* is set to the contents of the Field named *myField* on the Background of Card ID 2, or NIL if there is no Card ID 2. *value2* is set to the contents of the Field named *myTotal* on Card "Summary", or NIL if there is no Card named "Summary".

#### RELATED ITEMS

SETFIELD

GO *location* [*locationName*]

### DESCRIPTION

GO moves to a new Card, Application, or Stack. Use *location* to select the new location:

BACK - return to previously displayed Card.

TITLE - go to Card with the same name as the current Stack.

FIRST- go to first Card in the Stack.

PREV - go to the Card proceeding the current Card, or the last Card if the current Card is the first Card.

NEXT - go to the Card following the current Card, or the first Card if the current Card is the last Card.

LAST - go to the last Card.

CARD *locationName* - go to the Card specified by *locationName*. Either the Card name or ID may be specified.

APP *locationName* - run the application named *locationName*.

STACK *locationName* - open the Stack named *locationName*.

### EXAMPLE

```
GO TITLE
GO APP "calculator"
GO CARD 5
GO STACK "Recipe"
```

### OUTPUT

<The new Card is displayed>

### RELATED ITEMS

HIDE *item* | *itemList*

**DESCRIPTION**

HIDE hides the button or field *item* or list of buttons or fields *itemList*.  
To show buttons and fields use the SHOW Statement.

**EXAMPLE**

```
HIDE CARD BUTTON theButton  
HIDE CARD FIELD field_0, BACKGROUND FIELD field_1
```

**OUTPUT**

The specified buttons and fields are hidden

**RELATED ITEMS**

SHOW

**DESCRIPTION**

The LABELINPUT Field provides a label with a text entry line. The Field may also contain a pick-list. If it does, then a small diamond is displayed in front of the label. Tapping the label displays the pick-list. Tapping an item in the list enters it into the text entry line.

The Field contains the following properties:

`entryFlags`: recognition flags for the entry field, as used in `viewFlags`.

`label`: The label text

`labelFont`: The label font

`text`: The initial entry field value

`entryLine.text`: The user entered or updated entry field value

`labelCommands`: The optional pick list (an array of strings)

`curLabelCommand`: The initial selection from the optional pick list

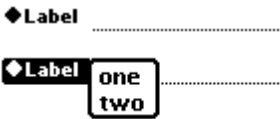
`viewValue`: The current selection from the optional pick list

You may also use these properties: `viewBounds`, `viewFlags`, `viewFont`, `viewFormat`.

**EXAMPLE**

20 NOTIFY(4, "LabelInput Changed", "New value is: " &  
CARD FIELD theField)

**OUTPUT**



**RELATED ITEMS**

HIDE, SHOW

## DESCRIPTION

The LABELPICKER Field provides a label with a text display line. The Field also contains a pick-list. A small diamond is displayed in front of the label. Tapping the label displays the pick-list. Tapping an item in the list displays it next to the label.

The Field contains the following properties:

text: The label text

labelCommands: The pick list (an array of strings)

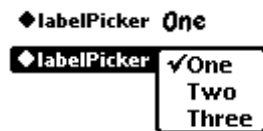
viewValue: The current selection from the pick list

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

## EXAMPLE

```
20 NOTIFY(4, "LabelPicker Changed", "New value is: " &  
CARD FIELD theField)
```

## OUTPUT



## RELATED ITEMS

HIDE, SHOW

DESCRIPTION

The MONTH field provides a display of a single month. The days of the month can be selected as in the Dates application.

The Field contains the following properties:

selectedDates: An array of integers (from the TIME() function) representing the selected dates. The first date determines which month is displayed. The default value is the current date.

noSelection: TRUE if DATEPCIKER is display-only.

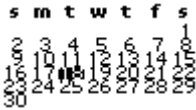
singleDay: TRUE if only a single day may be selected.

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

EXAMPLE

```
20 NOTIFY(4, "Date Selected", DATETIME(CARD FIELD
theField[0]))
```

OUTPUT



RELATED ITEMS

HIDE, SHOW, TIME, DATEPICKER



NEWCARD(*cardInfo*)

### DESCRIPTION

NEWCARD creates a new Card in the current Background and then displays that Card. The function returns the ID of the new Card. If NEWCARD is called within a script, the new card is created and the ID is returned.

You can supply some of the properties of the new Card in *cardInfo*. If *cardInfo* is {} then no properties are set in the new Card. You may supply one or more of the following fields for *cardInfo*:

*name* - the name for the new Card.

*arriveSound* - the symbol for the sound to play when arriving.

*leaveSound* - the symbol for the sound to play when arriving.

*arriveEffect* - the effect when arriving

*leaveEffect* - the effect when leaving.

You can access these values from the current Card by using *card.name*, *card.arriveSound*, etc.

### EXAMPLE

```
NEWCARD({name:"This Old Card", arriveSound: 'funBeep'})
```

### OUTPUT

<The new Card is displayed>

### RELATED ITEMS

DELETECARD

**DESCRIPTION**

The NEWSETCLOCK Field provides the standard Newton clock face for time display and entry. The clock face is drawn scaled to the supplied viewBounds. The contents of the Field are returned as an integer (from the TIME() function) representing the selected time.

The Field contains the following properties:

hours: The current setting of the hour hand.

minutes: The current setting of the minute hand.

You may also use these properties: viewBounds, viewFlags, viewFormat.

**EXAMPLE**

```
20 NOTIFY(4, "Time Changed", "New Time is: " &  
TIMESTR(CARD FIELD theField,0))
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW, SETCLOCK, WINDOW

NOTIFY(*level*, *header*, *message*)

**DESCRIPTION**

NOTIFY displays a standard Newton notification box containing the header and message specified. Script execution continues after the notice is displayed. The function returns a frame. If the user closes the notification display, the `seenByUser` field of the frame is set to `TRUE`.

Use *level* to control how the notification is displayed by setting it to one of the following values:

- 1: Put message in log, do not notify user.
- 2: Display message is pending Star. Tapping Star displays message.
- 3: Display message immediately and beep.
- 4: Display message immediately, but don't beep.

**EXAMPLE**

```
NOTIFY(4, "Demo Program", "There has been an unexpected error")
```

## OUTPUT



## RELATED ITEMS

ANSWER

**DESCRIPTION**

The NUMBERPICKER Field displays the standard Newton number picker. A number can be entered by tapping on the number display.

The Field contains the following properties:

**value:** An integer representing the selected number. The initial value of this field determines the initial display.

**minValue:** The minimum allowed value.

**maxValue:** The maximum allowed value. This number is used to determine how many digits to display. Seven digits are shown if **maxValue** is not specified.

**showLeadingZeros:** TRUE to display them, NIL to hide them.

**viewBounds:** The width is calculated automatically based on **maxValue**. The left value is then calculated from the supplied right value. The height (bottom - top) should be 32.

You may also use these properties: **viewFlags**.

**EXAMPLE**

```
20 NOTIFY(4, "Number Changed", "New Value is: " & CARD  
FIELD theField)
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW, WINDOW

**DESCRIPTION**

The PARAGRAPH widget provides a text display area that does not scroll.

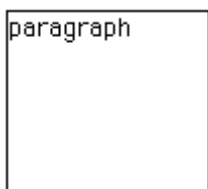
The Field contains the following properties:

text: The text displayed

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

**EXAMPLE**

SET CARD FIELD theField TO "New Text!"

**OUTPUT****RELATED ITEMS**

HIDE, SCROLLER, SHOW, TEXT

**DESCRIPTION**

The PICTUREBUTTON button displays a standard Newton button with an icon. The button highlights correctly when tapped.

You may use these properties with a PICTUREBUTTON: viewBounds, viewFlags, viewFormat.

**EXAMPLE**

```
20 NOTIFY (4, "PictureButton Tapped", "Bye!")
30 END SUB
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW, TEXTBUTTON

**DESCRIPTION**

The POPUPBUTTON Button provides a label with a text display line. The Button also contains a pick-list. A small diamond is displayed in front of the label. Tapping the label displays the pick-list. Tapping an item in the list displays it next to the label.

This Button contains the following properties:

labelCommands: The pick list (an array of strings).

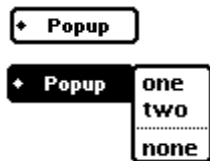
viewValue: The current selection from the pick list.

text: The label text.

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

**EXAMPLE**

```
20 NOTIFY (4, "Popup Button Tapped", "Item: " & CARD  
BUTTON theButton)  
30 END SUB
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW



RENAMESTACK(oldName, newName)

RENAMESTACK renames an existing stack. *oldName* is a string containing the existing name of the Stack, with :stk appended. *newName* is the new name for the Stack, also with :stk appended.

**EXAMPLE**

RENAMESTACK("Untitled:stk", "MyFirstStack:stk")

**OUTPUT**

The name of the Stack, as shown in the overview and Extras drawer, is changed from Untitled to MyFirstStack.

**RELATED ITEMS**

**DESCRIPTION**

The SCROLLER Field provides a text entry area that scrolls. When the user wishes to enter new text, they tap on the mountain icon. The Field will expand to fill the entire Newton screen, and the user can enter text. Tapping the mountain icon again shrinks the Field back to its original size. The scroll arrows scroll the Field in either view.

The Field contains the following properties:

`text`: The initial value

`boxTitle`: The title on the edit box. If the value of `boxTitle` is "" then the box title is omitted.

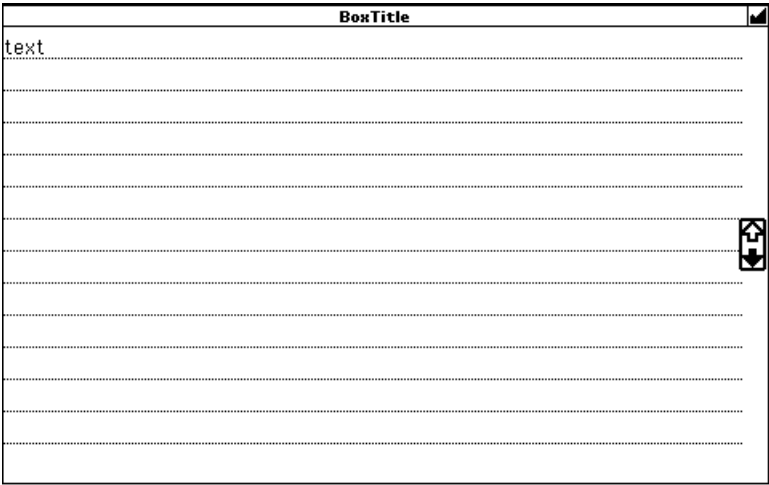
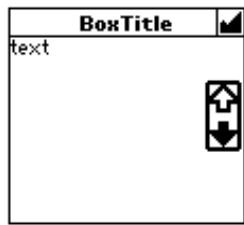
`editOK`: TRUE if the user can edit the text. If the value of `editOK` is NIL then the mountain icon is omitted.

You may also use these properties: `viewBounds`, `viewFlags`.

**EXAMPLE**

```
20 NOTIFY (4, "Scroller Changed", "New value is: " &  
CARD FIELD theField)  
30 END SUB
```

OUTPUT



RELATED ITEMS

HIDE, PARAGRAPH, SHOW

SET *variable* TO *value*

**DESCRIPTION**

SET is another way to perform assignments. It behaves exactly the same as the following LET Statement:

LET *variable* = *value*

**EXAMPLE**

```
20 SET x TO 5
30 LET x= x+ 1
40 SET x TO x - 10
50 PRINT X
```

**OUTPUT**

```
-4
*
```

**RELATED ITEMS**

LET

**DESCRIPTION**

The SETCLOCK Field provides a clock face for time display and entry. The clock face is always drawn such that it uses a 64x64 pixel area. You must be sure that your supplied viewBounds provides an area of this size. The contents of the Field are returned as an integer (from the TIME() function) representing the selected time.

The Field contains the following properties:

hours: The current setting of the hour hand.

minutes: The current setting of the minute hand.

You may also use these properties: viewBounds, viewFlags, viewFormat.

**EXAMPLE**

```
20 NOTIFY(4, "Time Changed", "New Time is: " &  
TIMESTR(CARD FIELD theField,0))
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW, NEWSETCLOCK

SETFIELD(card, context, field, value)

SETFIELD sets the value of a Card or Background Field on a Card. *card* is either a string containing the name of the Card or a number corresponding to the Card ID. *context* is either 'card to set a Card Field or 'background to set a Background Field. *field* is the name of the Field to set. *value* is the new value to store in the Field. SETFIELD returns true if it succeeds, or NIL if the specified Card could not be found. SETFIELD does not display the Card specified by *card*.

You can use the simpler Field reference when setting the value of a Card or Background field of the current Card. See page 128 for a description of Field references.

#### EXAMPLE

```
isOk1 = SETFIELD(2, 'background, 'myField, "New  
Value")  
isOk2 = SETFIELD("Summary", 'card, 'myTotal, 12)
```

#### OUTPUT

isOk1 and isOk2 will be set to true if there is a Card with ID 2 and a Card with the name "Summary". The Field named myField on the Background of Card ID 2 has its contents set to "New Value", and the Field named myTotal on Card "Summary" has its contents set to 12.

#### RELATED ITEMS

GETFIELD

SHOW *item* | *itemList*

**DESCRIPTION**

SHOW displays the button or field *item* or list of buttons or fields *itemList*. To hide buttons and fields use the HIDE Statement.

**EXAMPLE**

```
SHOW CARD BUTTON theButton  
SHOW CARD FIELD field_0, BACKGROUND FIELD field_1
```

**OUTPUT**

The specified buttons and fields are displayed

**RELATED ITEMS**

HIDE

**DESCRIPTION**

The SLIDER Field provides a gauge that the user can set. The value of the Field is a number from 0 (slider all the way to the left) to 100 (slider all the way to the right).

This Button contains the following properties:

viewValue: The current setting of the slider from 0 to 100.

text: The label text.

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

**EXAMPLE**

```
20 NOTIFY (4, "Slider Changed", "Value: " & CARD FIELD  
theField)  
30 END SUB
```

**OUTPUT****RELATED ITEMS**


GAUGE, HIDE, SHOW



STATUSBAR:HIDE()  
STATUSBAR:SHOW()

STATUSBAR:HIDE() hides the status bar at the bottom of the screen. Once hidden, the close button for NewtCard is not accessible. You must provide some means of closing the Stack or of turning the status bar back on.

STATUSBAR:SHOW() re-displays the status bar.

The keyboard combination -spacebar also toggles the status bar on and off

**EXAMPLE**

STATUSBAR:HIDE()

**OUTPUT**

<The status bar is hidden>

**RELATED ITEMS**

**DESCRIPTION**

The TEXT Field provides a text entry area that does not scroll. Hand written entry in this area will be recognized and converted into text. The `viewFlags` property can be used to indicate which recognition should be attempted.

The Field contains the following properties:

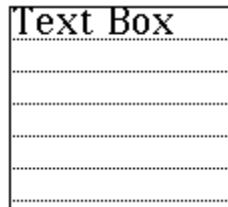
`text`: the text displayed and entered by the user

`viewLineSpacing`: spacing of the lines, in pixels

You may also use these properties: `viewBounds`, `viewFlags`, `viewFont`, `viewFormat`.

**EXAMPLE**

```
20 NOTIFY(4, "Text Changed", "New text is: " & CARD  
FIELD theField)
```

**OUTPUT****RELATED ITEMS**

HIDE, PARAGRAPH, SCROLLER, SHOW

**DESCRIPTION**

The TEXTBUTTON Button displays a standard Newton button with a text label. The button hilites correctly when tapped.

The Button contains the following properties:

text: the button label to display.

You may also use these properties: viewBounds, viewFlags, viewFont, viewFormat.

**EXAMPLE**

```
20 NOTIFY (4, "TextButton Tapped", "Bye!")
30 END SUB
```

**OUTPUT****RELATED ITEMS**

HIDE, SHOW, PICTUREBUTTON

**DESCRIPTION**

The TEXTLIST Button displays list of strings that may optionally include checkboxes for multiple selections and scroll arrows. If multiple selections are allowed the value returned is an array of numbers (from 0) representing the selected items in the list. If not the value is a single integer representing the selected item.

The Button contains the following properties:

`listItems`: an array of strings representing the list to display

`useScrollers`: when TRUE, show Newton scroll arrows if list does not fit within `viewBounds`. When NIL the user may drag the pen to scroll.

`useMultipleSelections`: when TRUE display checkboxes and allow more than one item to be selected

`scrollAmounts`: an array of three integers defining how scrolling should work. The first integer is the scroll amount, in pixels, when the user taps the scroll arrows. The second integer is the scroll amount when the user double-taps the arrows, and the third is the scroll amount if the user holds the pen down on an arrow.

`selection`: the last item selected

`selectedItems`: if multiple selections are allowed, this is an array of the indexes of the selected items.

You may also use these properties: `viewBounds`, `viewFlags`.

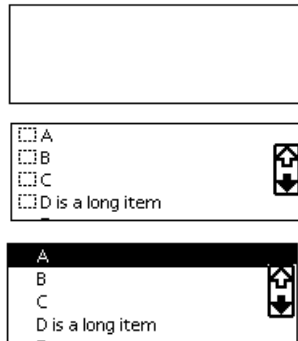
To update the display of `listItems` values after you've changed them in a script, use the expression:

```
_widgets.theButton:setupList()
```

### EXAMPLE

```
20 NOTIFY (4, "Text List Selection", "Selected: " &  
CARD BUTTON theButton)  
30 END SUB
```

### OUTPUT



### RELATED ITEMS

HIDE, SHOW

**DESCRIPTION**

The TITLE Field displays a text label formatted as a standard Newton title.

The Field contains the following properties:

`text`: the text of the title.

You may also use these properties: `viewBounds`, `viewFlags`, `viewFont`.

**EXAMPLE**

SET CARD FIELD theField to "Sample Title"

**OUTPUT**

**title**

**RELATED ITEMS**

HIDE, SHOW

## 9. Differences between NewtCard and NS BASIC

NewtCard's scripting language is based on NS BASIC, and almost all of the NS BASIC language is supported. There are a few differences that are outlined here. The first section of this chapter discusses new language features found only in NewtCard. The second section details those portions of NS BASIC that do not work in NewtCard, and the final section lists those portions of NS BASIC that behave differently in NewtCard.

### 9.1 New Features in NewtCard

NewtCard's scripting language includes several new constructs that make Scripting easier. Additionally, there are new forms of variables that you can use to store information in Stacks, Backgrounds, and Cards.

#### Frame References

NewtCard includes a new way to refer to the fields in a frame. In addition to NS BASIC's dot notation, you may use the more english like syntax *field OF frame*. The example below shows two equivalent scripts:

```
20 REM NS BASIC Style
30 myFrame = {name: "Smith", age: 22}
40 myFrame.age = myFrame.age + 1
```

```
20 REM NewtCard Style
30 SET myFrame TO {name: "Smith", age: 22}
40 SET age OF myFrame TO age OF myFrame + 1
```

## Button and Field References

You may retrieve or set the contents of a Button or Field by supplying its full name in an expression. The general form of the Button or Field expression is:

CARD | BACKGROUND BUTTON | FIELD *itemName*

For example:

```
SET CARD FIELD theText to "New Text"
SET isChecked TO BACKGROUND BUTTON theCheck
```

It is not necessary to use only BUTTON with Buttons, either BUTTON or FIELD may be used to access a Button or Field. The property value that is set or retrieved by the contents expression varies. Refer to each Button and Field entry in the reference section to see which property is accessed.

You can access specific properties of a Button or Field by using a property reference. The general form of the Button or Field property reference is:

*property* OF CARD | BACKGROUND BUTTON | FIELD *itemName*

For example:

```
SET text OF CARD FIELD theText to "Hello!"
SET isChecked TO viewValue of BACKGROUND BUTTON
theCheck
```



## Button and Field References on Other Cards

The SETFIELD and GETFIELD functions provide access to Buttons and Fields residing on Cards other than the current card.

## THE and THIS

You may use THE and THIS in any expression. Both are simply ignored, but can improve the readability of your scripts. For example:

```
SET THE name OF THE CARD FIELD myField TO "theField"
```

## Card, Background, and Stack Properties

You can examine and change the values properties of the current Card, Background, and Stack. Any changes you make to the values is saved with the Card, Background or Stack. For example:

```
Stack.markedCards=[]
```

un-marks all Cards in the current Stack.

```
PRINT Card.cdWidgets
```

Displays a frame containing fields for each Button and Field on the Card.

You may also create your own Card, Background, and Stack properties. These properties save their values. If you create a Card property, it is only available when that Card is the current Card. The same holds true for Background and Stack properties. A Stack property behaves like a global variable because any value stored there is accessible by all Scripts in the Stack.

Create a Card property using this expression:

SET *Card.property* TO *value*

or

SET *property* OF *CARD* TO *value*

You access the value store in the property using either *Card.property* or *property* OF *Card*. Make sure that the name you select for *property* is not already in use.

Likewise, Background and Stack properties are created and accessed using the following expressions:

*Background.property*  
*property* OF Background  
*Stack.property*  
*property* OF Stack

### Calling Functions in Cards and Fields

To call a card Button or Field function from a NewtCard script, use the following expressions:

*\_widgets.myCheckBox:toggleCheck()*  
*\_widgets.myPicker:setupList()*

If the Button or Field is on the background, preface the expression with *\_bgcd.background.*, as shown below:

*\_bgcd.background.\_widgets.myCheckBox:toggleCheck()*

## 9.2 Unsupported Features of NS BASIC

The following Statements and Commands are not supported in NewtCard: BYE, CHAIN, HWINPUT, LOAD, MAKEPACKAGE, REPLACE, REVUP, SAVE, TRACE, and WAIT.

Widgets (which are the NS BASIC form of Buttons and Fields) do not support the GOTO and GOSUB properties. Use Action Scripts instead.

## 9.3 NS BASIC Features that are Changed

The NOTIFY Function has changed, and is documented in the reference section. The WDRAW Statement works as documented in the NS BASIC handbook, but only for PARAGRAPH and DRAW Fields. When using WDRAW in NewtCard, use the field identifier in the statement, as shown below:

```
WDRAW CARD FIELD myDrawField, [makeoval(5,5,20,20)]
```

## 9.4 Button and Field Properties

In NS BASIC Buttons and Fields are called Widgets. The properties of Widgets are documented in the NS BASIC Handbook Reference section in the entry for WINDOW. The elements of the *windowSpec* used in a WINDOW Statement correspond to the properties of a Button or Field. All possible properties are described below:

```
viewBounds: {top: position1, left: position2, bottom: position3,  
             right: position4}
```

ViewBounds defines the location and size of the Button or Field. A Newton screen is 480 pixels wide by 320 pixels high.

viewFlags:      vApplication + vFloating + vClickable +  
                  vGesturesallowed + vSingleunit + vCharsallowed +  
                  vLettersallowed + vPunctuationallowed + vShapesallowed  
                  + vStrokesallowed + vCapsrequired + vNumbersallowed +  
                  vNamefield + vPhonefield + vDatefield + vTimefield

viewFlags defines the special characteristics of the Button or Field. Not all combinations are valid. Each characteristic is described below.

vApplication	TRUE to make window receive application messages, NIL otherwise
vFloating	TRUE to make window float over all others, NIL for normal window stacking
vClickable	TRUE if the window accepts pen taps
vGesturesallowed	TRUE to accept Newton gestures such as scrub.
vSingleunit	TRUE to accept only one word
vCharsallowed	TRUE to use word recognition
vLettersallowed	TRUE to use letter-by-letter recognition
vPunctuationallowed	TRUE to accept punctuation
vShapesallowed	TRUE to recognize Boxes, Lines, and Circles
vStrokesallowed	TRUE to accept digital ink
vCapsrequired	TRUE to capitalize first letter of each word
vNumbersallowed	TRUE to accept numbers
vNamefield	TRUE if this is a name field
vPhonefield	TRUE if this is a phone field
vDatefield	TRUE if this is a date field
vTimefield	TRUE if this is a time field

**Note:** Not all combinations of these values are valid. If you try a combination that does not look as expected, then you've found an invalid combination.

viewFont:     {family: *fontName*, face: *fontFace*, size: *fontSize*}

viewFont defines the font to be displayed in the Button or Field. *fontName* is the name of the font you wish to be used in the window. Possible fonts on the Newton are 'espy, 'geneva, 'newyork, or 'handwriting. **Note:** The ' sign is required.

*fontFace* is the style of the font: 0 for plain, 1 for bold, 2 for italics, 4 for underline, 8 for outline, 128 for superscript, 256 for subscript. *fontSize* may be 9,10,12,14 or 18.

**Note:** Not all combinations of *fontName*, *fontFace* and *fontSize* are valid. If you try a combination that does not look as expected, then you've found an invalid combination.

viewFormat:     *frameColor* + *fillColor* + x\*vfpen + y\*vfshadow + z\*vfround

viewFormat defines the visual format of the Button or Field. If viewFormat is 0, then the Button or Field is transparent. *frameColor* is the color (pattern) of the Button or Field border. *frameColor* may be one of:

vfFrameWhite	vfFrameLtgray
vfFrameGray	vfFrameDkgray
vfFrameBlack	vfFrameMatte (thick gray bordered by black)

*fillColor* is the color of the contents of the window. *fillColor* may be one of:

vfFillWhite      vfFillLtgray      vfFillBlack  
vfFillGray      vfFillDkGray

*x\*vfPen* sets the width of the border in pixels. *X* should be between 0 and 15. *Y\*vfShadow* sets the width of the shadow in pixels. *Y* should be between 0 and 3. *Z\*vfRound* is the corner radius, in pixels. *Z* should be between 0 and 15.

viewJustify:      *justifyCode*

viewJustify defines the type of justification used for the text displayed in the Button or Field.

text:

text contains the current text displayed in the Button or Field.

drawing:

drawing contains the current graphic displayed in the Button or Field using WDRAW.

# INDEX

<b>A</b>		<b>C</b>		Shapes	37
Adding a Script	68	Card		Stamp	35
ANSWER	83	Artwork	27	Text	32
AZTABS	85	Buttons & Fields	28	<b>F</b>	
AZVERTTABS	85	Creating	26	Fields	
<b>B</b>		Deleting	28	DATEPICKER	89
Background		Info	26	DIGITALCLOCK	92
Artwork	24	When Arriving	27	DRAW	93
Buttons & Fields	24	When Leaving	27	LABELINPUT	101
Creating	22	CARD BUTTON	128	LABELPICKER	103
Deleting	25	CARD FIELD	128	MONTH	104
Edit Toggle	14	CHECKBOX	86	NEWSETCLOCK	106
Info	23	CLOSEBOX	88	NUMBERPICKER	109
When Arriving	23	Commands		PARAGRAPH	110
When Leaving	23	FIND	94	SCROLLER	114
BACKGROUND BUTTON	128	Module	70	SETCLOCK	117
BACKGROUND FIELD	128	<b>D</b>		SLIDER	120
Beaming a Stack	18	DATEPICKER	89	TEXT	122
Buttons		DEF FN	95	TITLE	126
AZTABS	85	DELETEBACKGROUND	91	Fill Tool	37
AZVERTABS	85	DELETECARD	90	FIND	94
CHECKBOX	86	Deleting a Script	69	Frame reference	127
CLOSEBOX	88	desktop computer	2, 4, 7	Freehand Tool	34
LARGECLOSEBOX	88	DIGITALCLOCK	92	FUNCTION	95
PICTUREBUTTON	111	DRAW	93	Functions	
POPUPBUTTON	112	Drawing		ANSWER	83
RIGHTCHECKBOX	86	Aligning Items	39	DELETEBACKGROUND	91
TEXTBUTTON	123	Arranging Items	39	DELETECARD	90
TEXTLIST	124	Drawing Tools		GETFIELD	98, 129
Buttons & Fields		Fill	37	NEWCARD	105
Actions	50	Freehand	34	NOTIFY	107, 131
Changing Order	52	Line	33	RENAMESTACK	113
Deleting	52	Oval	34	SETFIELD	118, 129
Edit Slip	48	Pen	38	STATUSBAR:HIDE	121
Editing properties	49	Pointer	31	STATUSBAR:SHOW	121
Icon	50	Polygon	35	<b>G</b>	
Nameing	48	Rectangle	33	GETFIELD	98, 129
		Rounded Rectangle	34		

Global Variable	129	<b>P</b>	Starting NS BASIC	13
GO	99		Statements	
Go Back	14	PackIt	DEF FN	95
<b>H</b>		PARAGRAPH	FUNCTION	95
HIDE	100	Pen Tool	GO	99
Home Stack	19	PICTUREBUTTON	HIDE	100, 119
HOME.STK	19	Pointer Tool	SET	116
		Polygon Tool	SHOW	100, 119
		POPUPBUTTON	WDRAW	131
		Printing and Faxing Cards	STATUSBAR:HIDE	121
<b>I</b>			STATUSBAR:SHOW	121
Installing		<b>R</b>	SUB	70
On a Storage Card	6	Rectangle Tool	Subroutine	70
On the Newton	5	RENAMESTACK	Definition	70
<b>L</b>		RIGHTCHECKBOX		
LABELINPUT	101	Rounded Rectangle Tool	<b>T</b>	
LABELPICKER	103		TEXT	122
LARGECLOSEBOX	88	<b>S</b>	Text Tool	32
Line Tool	33	Script	TEXTBUTTON	123
		Adding	TEXTLIST	124
		Deleting	THE	129
<b>M</b>		SCROLLER	THIS	129
Menu Bar	14	Serial Cable	TITLE	126
Hiding and Showing	14	SET	Title Card	56
Message Window	25, 28	SETCLOCK		
MODULE	70	SETFIELD	<b>U</b>	
MONTH	104	Shapes Tool	User Level	42
		SHOW		
		SLIDER	<b>W</b>	
<b>N</b>		Stack	WDRAW	131
NCU	5	Deleting	Web Site	3
NEWCARD	105	Duplicating	Widgets	131
NEWSETCLOCK	106	Home	WWW	3
Newton Connection Utility	5	Info		
Notation Conventions	10	When Arriving	<b>X</b>	
NOTIFY	107, 131	When Leaving	XPort Lite	19
NUMBERPICKER	109	Opening		
		Packaging		
<b>O</b>		Saving		
OF	127	Untitled		
Oval Tool	34	stack_arrive()		
		Stamp Tool		



## USER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Please let us know if you would like a reply. Return to:

NS BASIC Corporation  
77 Hill Crescent  
Toronto, Canada M1M 1J3  
fax (416) 264-5888

Page	Comments