NEWTON Q&A: ASK THE LLAMA

This Column first appeared in volume 19 of DEVELOP (the Apple technical journal for developers)

Q *I'm having trouble with the protoRoll. I have a protoApp with a protoRoll at the bottom with a couple items in it. (Note that I'm not using the protoRollBrowser proto.) It compiles OK, but w download to the Newton, nothing shows up. In fact, when I use the inspector to look at the vie hierarchy, the protoRoll doesn't show up at all. The other views are fine. What am I doing wro*

A The protoRoll doesn't show up because of the setting of the viewFlags of the ROM prototype: t vApplication and vClipping flags are set, but not the vVisible flag. If the protoRoll were the bas template of your application, the vApplication flag would be sufficient to make it visible.

In your case, the protoRoll is a child of your base application template. Since it isn't visible (vVisible isn't set), the system doesn't create a runtime view frame for the child. You could get system to create the runtime view by declaring the protoRoll to be the base template, but this sti wouldn't show the protoRoll.

To make the protoRoll visible, add a viewFlags slot to the protoRoll and check the vVisible flag You may or may not want to uncheck the vApplication flag. If you uncheck it, the system will n longer send scroll and overview messages (viewScrollUpScript, viewScrollDownScript, viewOverviewScript) to the protoRoll, so it will appear to be broken. But you can support these messages in your base application view and just pass them on to the protoRoll as needed. If you leave the vApplication flag checked, protoRoll will get the scroll events.

Q *My print format never seems to get called, ever. I don't get a printNextPageScript or even a viewSetupFormScript. I'm not using ROM_coverPageFormat because I don't ever want to prin cover page. How can I get this to work?*

A The answer to your problem is in your question. A print (or fax) format must proto to ROM_coverPageFormat; it's not optional (as the manual implies). It may help to know that ROM_coverPageFormat is really misnamed. The generation of a cover page is controlled by a s in your format. The proto should be called something like ROM_allThePrintingAndFaxingBehaviorProto, but that would be verbose :-)

Q *I would like to add a [button|view|Llama] to the [Notepad|Calendar|Cardfile|etc.]. How can I do safely?*

A This is a simple one: you can't. If you add any element to a built-in application, you take the ch that your application will break in future releases of the MessagePad. Also note that adding llam to MessagePad will theoretically cause a multidimensional implosion. ("Don't cross the llamas, . . beams." — LlamaBusters)

 Q *I've n*

some peculiar behavior in the Compile function and am wondering if it might be a bug. The pro is with special characters and string objects. When Compile is passed a string object containing special characters rather than a literal string with Unicode codes, the result is incorrect. This exa works as expected:

```
x:= Compile("{msg: \"A string with special character \u00A5\u\"}";
y:= :x();
```
—> y is {msg: "A string with special character ¥"}

This example doesn't work as expected:

```
a:= "A string with special character \u00A5\u";
x:= Compile(a);
y:= :x();
```
*—> y is {msg: "A string with special character *"} where * is some character other than the exp "¥".*

Can you explain what's going on here?

A The problem is that you're using illegal NewtonScript syntax in the second example. If you use the inspector instead of Compile for this example, it would be like typing

```
A string with special character \u00A5\u
```

and then hitting Enter. This would result in a syntax error from NewtonScript. What you proba want is the equivalent of typing

```
"A string with special character \u00A5\u"
```

into the inspector. This is done with the following call to Compile:

```
x := Compile("\"A string with special character \\u00A5\\u\"");
call x with ();
```
—> #4415F49 "A string with special character ¥"

Note that the escape characters (\) for the Unicode string are themselves escaped. If you don't d this, you'll be putting the actual Unicode characters into the string being compiled, which is probably not what you want. Although your first example worked, you could easily get a case where not escaping the escape characters could bite you.

Q *In the communications input spec below, why does the call to UpdateStatus fail? UpdateStatus method in my base view, and the whole endpoint is in my base view, so why can't the input sp find the method?*

```
tMessage: {
   inputForm: 'string,
   endCharacter: unicodeCR,

   InputScript: func(endpoint, data)
   begin
      :UpdateStatus(data);
      endpoint:SetInputSpec(GetMessage);
   end;
}
```

A The call to UpdateStatus fails because it's a message send that uses full inheritance to find the method. That means the system will look in the current context (that is, self), then check the pro chain, and then check the parent chain. However, the current context is not what you think it is an input spec, the current context is the frame that defines the input spec. In this case, it's the GetMessage frame you define.

Since the GetMessage frame has no proto or parent pointer, the message send fails. There's a second problem waiting to happen: the call to SetInputSpec will also fail, because the symbol GetMessage isn't valid in this context.

The solution is to get a reference to your base view (or other view which contains or inherits the UpdateStatus message). The usual way to do this is to add an _parent slot to your endpoint at ru time during initialization. Now your InputScript can use endpoint._parent to find the base view follows:

```
InputScript: func(endpoint, data)
begin
   endpoint:UpdateStatus(data);
   endpoint:SetInputSpec(endpoint.GetMessage);
end;
```

If you really want to use a simple message send (for example, :UpdateStatus), you could add a _parent slot to the input spec. This may be useful in situations where you have several input scr that rely on a dynamic inheritance mechanism. That is, you change what the _parent slot of the input spec points to on the fly.

Q *I have a large amount of static data in my application. I'd like to use Project Data to edit this data it won't fit. What can I do?*

A You must have an old version of Newton Toolkit. As of version 1.0.1, the 32K limit is gone. Y could use another text editor to edit the Project Data file. You could also use the Load command load anothe Newto

ript source file.

As an example, assume you had a file called MyData.f in the same directory as your project and this file contained the script that defined your constant data structures. You could use the Load command like this:

```
// This line appears in your Project Data file.
// Load in the data file and use the HOME compile-time variable
// to get the path to the project folder.
Load(HOME & "MyData.f");
```

Q *Did you know that "gullible" is not in the Newton dictionary?*

A It is now.

Q *How can I figure out how much space my package and data will take on a card? I really want m application to fit on a 1-meg card.*

A The short answer is, you can't. The long answer is, load your packages and soups after comple erasing the card. To completely erase the card, open up preferences and then insert the card. Be the card is loaded, you'll get a chance to erase it.

Look at the difference in the free space on the card. Use the value in the card dialog. The value the remove package picker is the uncompressed size. You must erase the card before you check free space difference.

Q *I have an input spec that receives data and places it into a queue. When I get data, I set a flag in base view (DataInQ) that indicates data is available. I know the data is getting sent, but my inpu specs never seem to get called. What's going on?*

A The chances are that your base view has some code like this:

```
myBase.WaitForData := func()
   while Not DataInQ do nil;
```

You may have more statements in the loop, and you may be using **repeat** instead of **while**, bu you probably have a loop that waits for the DataInQ flag to be set. The problem is that you're n giving control back to the NewtonScript thread so that it can process the pending InputScript ca (from your input spec).

If you really need to wait for data, you can use either an idle script or a repeating delayed action The idle script will be significantly easier to implement. You should make the delay on your idl script enoug give ti

to the Newton. Also note that the Newton is a battery-powered device, and excessive use of thi kind of programming tends to drain the users — I mean, batteries.

Q *I have an array of text elements called MyFirstArray in my Project Data file. I want to set the te clParagraphView that I open to an item in this array. The clParagraphView has a slot (strRef) th references MyFirstArray[0]. The first element appears as the clParagraph's view. There are fou buttons on the base view, and depending on which button is tapped I want a different element c array to be the clParagraph's text. When I try replacing MyFirstArray[0] in strRef during the viewSetupFormScript, I get as text "MyFirstArray[1]", not the text this represents. Here's the c in SetupFormScript in the clParagraph:*

```
SetValue(self, 'strRef, "MyFirstArray["&tempslot&"]");
```

tempslot is a slot in the base view where I store a value depending on which button is tapped. W the problem?

A The basic answer is that your SetValue statement is incorrect. This statement sets strRef to the string "MyFirstArray[" concatenated with the string representation of tempslot concatenated wit "]". What you really want is the string that's in MyFirstArray at the position defined by tempslo that statement would be

```
SetValue(self, 'strRef, MyFirstArray[tempslot]);
```

But there are better ways to do this. Which method you use depends on when you set the text o clParagraphView. If you set up things at open time, use the viewSetupFormScript, but just assi directly to the text slot:

```
clParagraphView.viewSetupFormScript := func()
   text := MyFirstArray[tempslot] ;
```

Remember that SetValue will also dirty the view and call RefreshViews. This isn't something y want to happen when you Open a view.

The other case is that the clParagraphView is already open. In this case, you can use a SetValue statement to set the text slot directly, instead of setting a strRef slot.

One other note: If the user can edit the strings you place in a clParagraphView, you must Clone string. Otherwise you can get a "tried to modify read only object" error.

Q *How long does it take to train a llama to be a competent NewtonScript programmer?*

A About four weeks, but the hooves get in the way of really fast coding.

he unofficial mascot of the Developer Technical Support group in Apple's Personal Interactive Electronics (PIE) division. Send your Newton-related questions to NewtonMail DRLLAMA or AppleLink DR.LLAMA. The first time we use a question from you, we'll send you a T-shirt.

Thanks

to our PIE Partners for the questions used in this column, and to jXopher, Todd Courtois, Bob Ebert, Mike Engber, Kent Sandvik, Jim Schram, Maurice Sharp, and Scott ("Zz") Zimmerman for the answers.

Have more questions?

Need more answers? Take a look at PIE Developer Info on AppleLink.