NEWTON Q&A: ASK THE LLAMA

Q *I was wondering why my autopart is taking up so much heap space after it's installed. The InstallScript and RemoveScript are quite small:*

```
InstallScript := func(partFrame, removeFrame) begin
if NOT GetGlobals().(kGlobalDataSym) exists then
   GetGlobals().(EnsureInternal(kGlobalDataSym)) := {};
   GetGlobals().(kGlobalDataSym).(EnsureInternal(kAppSymbol))
      := GetLayout("MyTool.t");

   end;

      RemoveScript := func(removeFrame) begin
         local NGP := GetGlobals().(kGlobalDataSym);
         if hasSlot(NGP, kAppSymbol) then
            RemoveSlot(NGP, kAppSymbol);
      end;
```

The template in MyTool.t is only a simple proto with a few slots in the base view: a symbol, viewBounds, viewFlags, viewJustify, viewFormat, viewClickScript, and three methods. When installed, the autopart takes up about 640 bytes of heap space. Is this because of the physical representation in the extras drawer?

A A minimal autopart with no RemoveScript will take up about 240 bytes. One with a RemoveScript will take 350 bytes or more, depending on the size of the RemoveScript. The 240 bytes are used by the system to keep track of the package. This includes the name, the extras drawer entry, and any symbols that you passed to EnsureInternal.

Trying your code showed that it used 444 bytes. Of course, we don't have the MyTool.t layout in the package, but that doesn't matter since you don't call EnsureInternal on the layout. Using 444 bytes isn't out of line considering the minimum size of an autopart.

Of course, you could make your code a little smaller. For example, in your RemoveScript you check that kGlobalDataSym is a known global variable. This isn't required, since RemoveSlot will do the right thing if kAppSymbol isn't in your global data frame. Note that you may want to make sure your global data frame exists:

```
RemoveScript := func(removeFrame)

RemoveSlot(GetGlobalVar(kGlobalDataSym), kAppSymbol);
```

Also, you use GetGlobals, which is a deprecated function for Newton OS 2.0. You should be using GetGlobalVar, DefGlobalVar, and UnDefGlobalVar.

Q *I'm using a protoPeoplePicker in Newton OS 2.0 and it keeps showing an "Untitled Person." Is this an opportunity to create someone or is there really a gremlin inside the machine?*

A There are only two reasons for the appearance of the "Untitled Person" entry. One is that you mistakenly entered that person in your card file, which can happen if you edit an entry and accidentally erase the name. The more likely reason is that this is your hacking Newton device. If so, you probably didn't go through the Setup application and set an owner. "Untitled Person" is the default owner of the machine. We recommend that you set up a default hacking MessagePad, back it up using NBU (Newton Backup Utility), and then use that backup to create development units.

Q *I've read the article in Newton Technology Journal on package freezing and I'm still a bit confused. Can you confirm that the following questions and answers are right?*

• *Does my form part still have a slot in the root after being frozen? No.*

• *Is the RemoveScript called when it's frozen? Yes.*

• *Is the InstallScript called when it's unfrozen? Yes.*

• *Can I prevent freezing/unfreezing? No.*

• *Can I get a message indicating freezing vs. pulling the card? No.*

• *What happens when the user tries to put away an item routed to a frozen application? A "can't find application" error.*

A Congratulations, you understand package freezing correctly. And you win an all-expenses paid visit to your nearest Green Giant food processing plant.

Q *There are times in my application when I want to perform the same operation on a whole bunch of soup entries. Right now the Xmit form of the calls takes quite a while and results in a lot of messages flying around. Is there a better way to do this?*

A Yes. You can use nil for the argument that tells the system which application is performing the change. This tells the system not to transmit the change notification. Then you can use XmitSoupChange to send a whatThe notification. This will inform other applications that something changed, but not give specifics of the change. As an example, here's a code snippet to remove all entries in a given cursor:

```
// create a function we can map over local myRemoveFunction :=
func(entry) EntryRemoveFromSoupXmit(entry, nil);

// remove the entries MapCursor(removeCursor, myRemoveFunction);

// now inform registered soups that something has changed
XmitSoupChange(kSoupName, kAppSymbol, 'whatThe, nil);
```

Q *We're having a problem with compiled NewtonScript code. We have a function that takes an int as a parameter. We use the int type indicator in the function definition. However, it's possible for the parameter to be nil. For compiled code this results in a type mismatch error. Is there a workaround for this?*

A There is a way to work around this problem; however, you'll pay a performance penalty. It's much better to redesign your API to accept only integers. If you do want a workaround, you can use the type-checking functions to make sure the parameter is an integer before you use it like one. Here's some code that will work:

```
func native(x) begin
local int intx;

   if IsInteger(x) then begin
      // now you can use intx and it'll be faster than using x
      intx := x;
   end else begin
      // do whatever else is appropriate; it isn't an integer
```

```
      ...
   end;
end;
```

Q *In our application we sometimes have to show the user a big error message, which unfortunately is too large for the Notify mechanism. Is there a way we can add a button to the Notify slip? If not, is there some other mechanism we can use?*

A The answer is simple: just don't let your user generate such a large error. But seriously, there is no way to modify the slip that comes up from Notify. Here are two ways to solve your problem:

• Have the Notify message tell the user to click on some user interface element in your application for more information. This is the recommended solution.

• Instead of Notify, use a dialog to inform the user of the error. Since you control the dialog, you control which buttons are in it. You could set up such a dialog with AsyncConfirm.

Q *We need a way to find out whether a particular view inherits from a particular proto. For example:*

```
aView := {_proto: anotherView,

      // more slots...}

anotherView := {_proto: protoFloater,

      // more slots...}
```

Is there a function I could call that would take aView and return true if it's an instance of protoFloater?

A There is no built-in function that will do this, but it's relatively simple to write:

```
func(frame, prot) begin
   while (frame AND (frame <> prot))
      frame := frame._proto;
   return (frame <> nil)
end;
```

Q *How do I define a pickerDef column to be “lastname, firstname” in a single column?*

A You can specify a Get method in your pickerDef and modify your columns appropriately. As an example, take a look at the protoListPicker sample on the Newton Developer CD. One of the subsamples is called ListPickerSoup. The default is to display the first item in the first column and the second item in the second column. The original pickerDef is defined as follows (from pickerDef.f):

```
DefConst('kMyBasicSoupDataDef, {
_proto: protoNameRefDataDef, // required
validationFrame: nil, // used if editing is supported
name: "Random Data ", // name at top left of picker if
                     // foldersTabs are present
HitItem: func(tapInfo, context) begin
   context:ThingChosen(tapInfo);
end,
});
```

Then in myListPicker in the listPickerSoup.t layout file, the pickerDef slot is:

```
{_proto: kMyBasicSoupDataDef, // defined in the pickerDef.f file
   class: 'nameRef, // always include
   columns: [
      // Column 1
      {
         fieldPath: 'first, // field to display in column
         tapWidth: 100, // width for checkbox & name combined,
                        // offset from the right margin
         doRowHilite: true,
      },
      // Column 2
      {
         fieldPath: 'second, // field to display in column
         tapWidth: 0, // width from preceding column to
                     // right bounds
         doRowHilite: true,
      },
   ],
}
```

To modify this sample to show one column in the format “second, first”, you would add the following Get method to kMyBasicSoupDataDef:

```
Get: func(object, fieldPath, format) begin
   if fieldPath = 'second AND
      (format = 'text OR format = 'sortText) then
   begin
      local realData := EntryFromObj(object);
      if realData then // format is "second, first"
         return (realData.second & ", " & realData.first);
      else
         return "- -";
      end else
         inherited:?Get(object, fieldPath, format);
end
```

This Get method will return the correct string for a column that displays the slot 'second. It will also sort on a string that’s in the format “second, first”.

The other thing you need to do is modify the columns definition. Simply remove the first column, so that the pickerDef in myListPicker looks like this:

```
{_proto: kMyBasicSoupDataDef, // defined in the pickerDef.f file
   class: 'nameRef, // always include
   columns:
   [
      // Column 2
      {
         fieldPath: 'second, // slot to display in column
         tapWidth: 0, // width from preceding column to
                     // right bounds
         doRowHilite: true,
      },
   ],
}
```

Q *I have a pick list that takes quite a while to create. I’d like to use a weak array so that I don’t have to keep creating the list. That way I get garbage collection for free. But I don’t want to make the array “weak” until after the pick list has been opened by the*

picker so that items don't accidentally get garbage collected. How do I turn a regular array into a weak array? Or will this work at all?

A You can't turn a regular array into a weak array; an array is one or the other. But using a weak array should work, with these minor modifications to your code: Create a slot in your picker (say myWeakArray) and initialize it to a weak array. Create your regular array of pick items as usual. Let the user pop up the picker; then in either the pickCancelledScript or the pickActionScript, set the first element of myWeakArray to the array of list items. Next time you want to construct the pick list, check for the first element of myWeakArray. If it exists, you have your pick list; if not, create a new one.

Q *I'm using one of the newt name views to select a name. Whenever the people picker comes up, it's viewing "All Names." How can I programmatically change the default folder used by the picker?*

A You need to change the Picker method of the newt name flavor as follows:

```
Picker: func()
   protoPeoplePopup:New('|nameRef.people|, nil, self,
      {labelsFilter: <symbol-for-desired-folder>});
```

The final argument to the New method is a frame of options. Each slot/value pair is used to set up a slot/value pair in the protoPeoplePicker. So grab the symbol for the default folder that you want and set the labelsFilter of the protoPeoplePopup.

Q *What is the path to true enlightenment and wisdom?*

A Simple: Buy and read all the books that claim to show you such a path. As you read, make a list of the major points. Take that list, cross off the contradictions, take the inverse of what's left, and then get a life. Alternatively, go and code another thousand lines of NewtonScript.

The llama is
the unofficial mascot of the Developer Technical Support group in Apple's Newton Systems Group. Send your Newton-related questions to

dr.llama@newton.apple.com. The first time we use a question from you, we'll send you a T-shirt.

Thanks
to jXopher Bell, Bob Ebert, David Fedor, Ryan Robertson, Jim Schram, Maurice Sharp, and Bruce Thompson for these answers.

If you need more answers,
take a look at the Newton developer Web page, at http://www.devworld.apple.com/dev/newtondev.shtml.