Chapter 13
Packages

13.1 Contexts

When writing a large program, it is easy to accidentally use one symbol for two different quantities in different parts of the program. This leads to difficult-to-find bugs. This is especially true if parts of the program are written by different persons (in particular, when some packages from the standard library, or third-party packages, are used). To avoid such problems, contexts are used.

In *Mathematica*, a full symbol name consists of two parts: the *context* and the *short name*. Two symbols in different contexts may have the same short name. For example, the global symbol x and the symbol x in the contexts a and b are unrelated.

In[1] := x = 1; a`x = 2; {x, a`x, b`x}

Out[1] = {1, 2, b`x}

Contexts may be nested. Here the variable x lives in the context b which lives in the context a (and thus is unrelated to the global context b used above).

In[2] := a`b`x

Out[2] = a`b`x

The current default context is held in the variable $Context. It is used when a new symbol is created without specifying its context.

In[3] := $Context

Out[3] = Global`

When a symbol without an explicit context is used, it is being searched in contexts specified in $ContextPath.

In[4] := $ContextPath

Out[4] = {PacletManager`, QuantityUnits`, WebServices`, System`, Global`}

Built-in symbols live in the context System`.

In[5] := {Context[x], Context[a`x], Context[b`x], Context[a`b`x], Context[Pi]}

Out[5] = {Global`, a`, b`, a`b`, System`}

You can change $ContextPath using standard list functions. The function Remove removes symbols from the system.

In[6] := Clear[x, a`x]; Remove[a`x, b`x, a`b`x]

A. Grozin, *Introduction to Mathematica® for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-319-00894-3_13, © Springer International Publishing Switzerland 2014

13.2 Packages

Mathematica comes with a library of packages extending its built-in functionality. A package can be loaded by

In[7] := <<Quaternions`

Now this context is prepended to $ContextPath.

In[8] := $ContextPath

Out[8] = {Quaternions`, PacletManager`, QuantityUnits`, WebServices`, System`, Global`}

Short names will be searched in this context first, possibly shadowing variables and functions from Global`.

In[9] := Context[Quaternion]

Out[9] = Quaternions`

The package Quaternions lives in the directory Quaternions, which lives in the standard library directory.

Many additional packages are available at http://library.wolfram.com/infocenter/MathSource/. You can download them and install somewhere in your $Path.

You can instruct *Mathematica* to load a package whenever any function defined in it is used.

In[10] := DeclarePackage["NumericalCalculus`", {"EulerSum", "NLimit", "ND", "NSeries", "NResidue"}]

Out[10] = NumericalCalculus`

In[11] := ND[x^2, x, 1.0]

Out[11] = 2.

13.3 Writing Your Own Package

Begin, End

Begin["*a*`"] changes the default $Context; all symbols defined after this will live in this context.

In[12] := Begin["*a*`"]

Out[12] = *a*`

In[13] := $Context

Out[13] = *a*`

In[14] := z = 0; Context[z]

Out[14] = *a*`

End[] restores the previous $Context.

In[15] := End[]

Out[15] = *a*`

In[16] := $Context
Out[16] = Global`
In[17] := *a`z*
Out[17] = 0

BeginPackage, EndPackage

BeginPackage["*a*`"] sets $Context and changes $ContextPath in such a way that only the contexts *a*` and System` are available.
In[18] := BeginPackage["*a*`"]
Out[18] = *a*`
In[19] := $Context
Out[19] = *a*`
In[20] := $ContextPath
Out[20] = {*a*`, System`}
In[21] := *u* = 1;

EndPackage[] restores the previous $Context; ContextPath gets its old value prepended by *a*`, so that symbols defined after BeginPackage[*a*`] remain available.
In[22] := EndPackage[]
In[23] := $Context
Out[23] = Global`
In[24] := $ContextPath
Out[24] = {*a*`, NumericalCalculus`, Quaternions`, PacletManager`,
QuantityUnits`, WebServices`, System`, Global`}
In[25] := *u*
Out[25] = 1
In[26] := Clear[*z*, *u*]

A Typical Package

A simple package looks like this.
In[27] := FilePrint["APackage.m"]
BeginPackage["APackage`"]
f::usage = "f squares its argument"
Begin["`Private`"]
g[*x*_] := *x*^2
f[*x*_] := Expand[*g*[*x*]]
End[]
EndPackage[]
After BeginPackage["APackage`"], publicly available functions and variables of the package are introduced, usually by assignments to their usage messages.

Implementation of the package is done in the context APackage`Private`, which may contain additional functions and variables (not seen by users of the package).

In[28] := <<APackage`

In[29] :=?*f*

Out[29] = f squares its argument

In[30] := *f*[*a*+*b*]

Out[30] = $a^2 + 2ab + b^2$