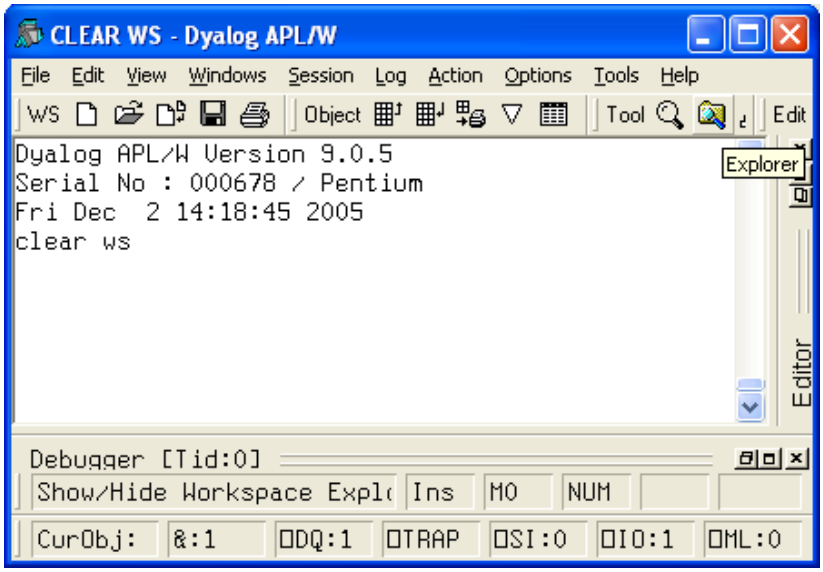
Module4: The Session Object

There is only one object of `Type Root`, called `#`, and there is only one object of `Type Session`, called `⎕SE`. `⎕SE` itself has properties, events and methods, and can support certain types of children, just like other objects. In the development version of Dyalog the child content of `⎕SE` is loaded from a DSE file when APL starts. In the runtime version `⎕SE` is empty of children but still retains its innate characteristics. First we shall approach the Session object from the point of view of a user of the development environment. Then we shall explore its contents from the point of view of an object-oriented programmer.

§ 4.1 Using the Session Object

§§ 4.1.1 Immediate Execution Mode of ⎕SE

An *APL session* is, traditionally, the environment wherein one writes and executes APL code. In the APL 1 and APL 2 eras of **Timesharing**, the session was considered to be the period of time during which ones terminal was connected and signed on to a mainframe computer running APL. Now, in APL 3 and APL 4 eras of **Windows**, the session is the window in which one executes APL expressions and defines APL functions. (See *Vector Vol.3 No.3 p97* for an earlier vision of APL3.)



The window is live. Expressions are, as ever, executed as soon as the **Enter** key is pressed. (**Enter** initiates execution of the code on the line inhabited by the cursor.) Immediate execution now extends to GUI objects. Objects are displayed as soon as they are created. Objects respond to mouse and keyboard when created in the development environment in the same way as they would in the runtime environment under `⎕DQ`. This takes the Dyalog APL Session to a new level of sensitivity as regards the modern GUI understanding of immediate execution mode.

§§ 4.1.2 Tracer and Editor of ⎕SE

In the days of timesharing, one of the most important keystrokes to learn in APL was how to stop execution. Every second of execution time cost real money, so you had to learn **Ctrl+C** early in your career. Now, on a PC, runaway code is less frightening so **Ctrl+Break** is, perhaps, less important than:

- **Ctrl+Enter** initiates step by step execution of code on the line …
- **Shift+Enter** opens for editing the program located at the cursor (or suspended function when the cursor is in column 1)

- **Ctrl+Shift+Enter** jumps over the current line in the tracer (cf **Ctrl+Shift+Backspace)**

Other significant keys, such as **Esc**, can be found in the current DIN input file whose name and location can be discovered from the result of

```
      #.GetEnvironment¨'aplkeys' 'aplk'
```

4.1.2.1 Investigate, using Notepad.exe, the content of your current .DIN file. Use the function *KEYPRESS* in supplied workspace ..\WS\UTIL.DWS to investigate further.

The GUI implementation of the APL Session ought to follow, as far as is reasonable, the general Windows standards. For more information, see the voluminous and detailed Dyalog APL *User Guide*.

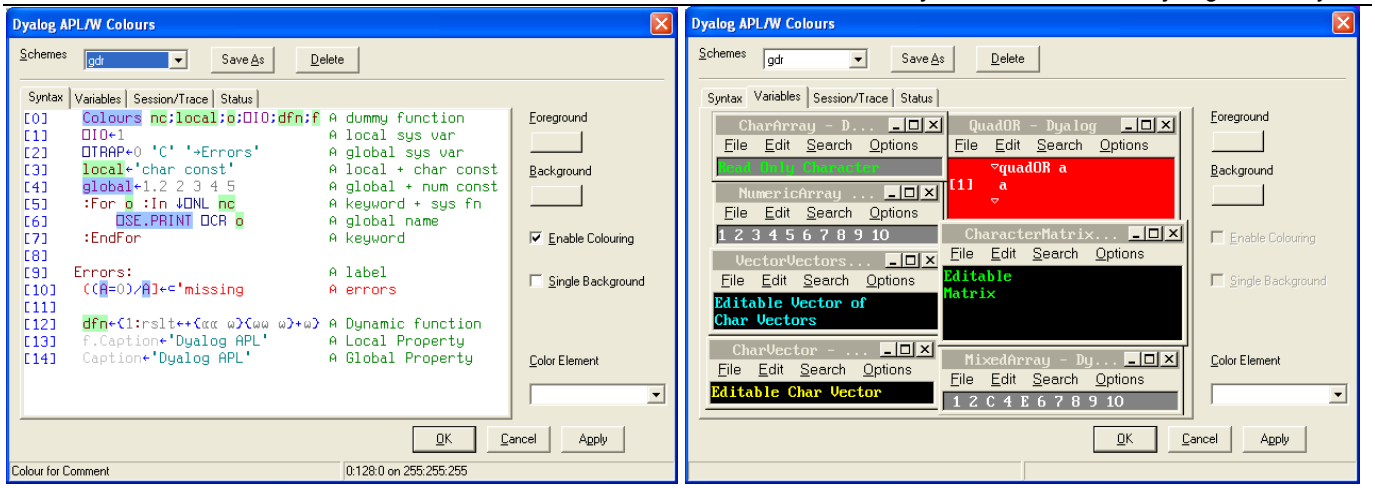
§§ 4.1.3 Choosing syntax Colours

Syntax coloring is a valuable aid to reading and writing APL programs. The choice of colours should make some sort of sense to the programmer, you. It is particularly important to choose nice meaningful colours for global and local names. There are many possible criteria. Very pale background colours for some of the elements seem to work well.

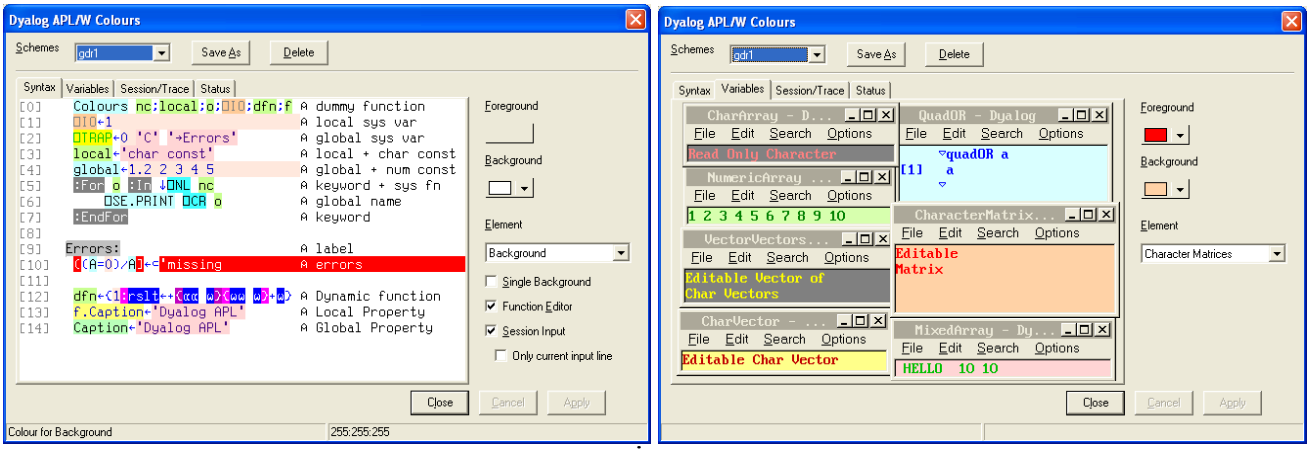
4.1.3.1 How would *you* colour the various syntactic elements in APL?

Below is a suggested scheme whose philosophy rests on proposed correlation between name class (⎕*NC*) and the average electromagnetic frequency of the colour. Elements of class 1, 2, 3 and 4 are aligned with the spectrum of colours: Black Red Orange Yellow Green Cyan Navy Violet White, or any intermediate colours/intensities. (For this purpose name class 9 is regarded as name class 2.) Unclassified APL symbols may be placed in any suitable position in the 'power' spectrum from labels (1) to operators (4).

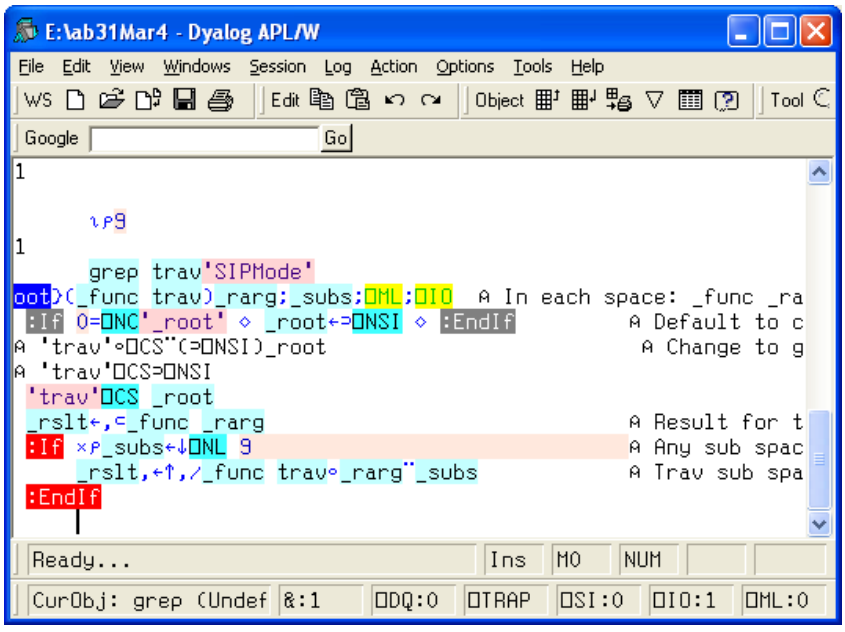
Class	Syntax Colour	Syntax Element
0	Gray (192)	Line Numbers
1	Black	Labels
1	Dark Red	Control Structures
	Red	Error - Unmatched parentheses, quotes, and braces
2.1	Orange	Character constants
2.1	Dark Yellow	Numeric constants
2	Black on Pale Red	Localised System Variables
2	Red on Black	Global System Variables
2.4	Black on Pale Yellow	Local GUI Property
2.4	Yellow on Gray	Global GUI Property
2∨3	Black on Pale Green	Local Names (functions and esp[ly] variables)
0	Dark Green	Comments
3∨2	Black on Pale Blue	Global Names (variables and esp[ly] functions)
3	Black on Pale Cyan	System functions
3.2	White on Light Navy	D-Fn name
3∨4	Navy	Primitives (functions/ops/special…)
4.2	White on Light Violet	D-Op (monadic)
4.2	Violet	D-Op (dyadic)
	White	Background



or



The main Session window can also be coloured. See [Options][Colours…][Session Input] check button.



APL code for the Google bar was kindly given free to dyalogusers@yahoogroups.com by Norbert Jurkiewicz and is discussed a little in Module 8.

§ 4.2 Inside the Session Object

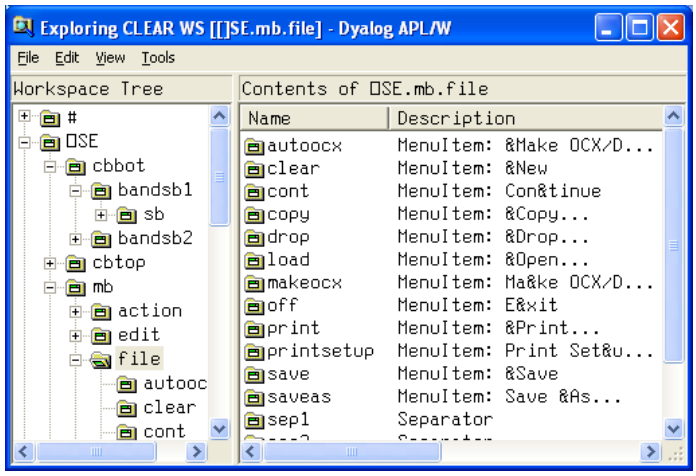
§§ 4.2.1 Exploring the Workspace and ⎕SE

The ⎕SE window appears to be built from a multi-line Edit object or a RichEdit object just as the Grid object appears to be built from a matrix of Label or Edit or Combo .. objects. Normally we can only explore these superobjects to the point allowed by the object programmer who decides (as we shall see later) which properties, which methods and which events to expose to the outside world. These lists can be found from the PropList, MethodList and EventList properties of ⎕SE. (Note that [Options][Object Syntax][Expose Session Properties] or registry parameter PropertyExposeSE have to be set in order to expose GUI names for the Session object.)

4.2.1.1 Change into the Session space and examine its properties. Assign Posn.

§§ 4.2.2 Examining Session Menus and Buttons

The quickest way to gain an overview of the contents of ⎕SE is by way of the Workspace Explorer, to be found in [Tools][Explorer…] or via the Explorer button tipped on the session in §§ 4.1.1



4.2.2.1 Use the Workspace Explorer to investigate the object hierarchy in the Session object.

4.2.2.2 In the explorer, use right click [Set Session Space] to change into the space of the Explorer ToolButton Bitmap, ⎕SE.cbtop.bandtb3.tb.explorer.bm, and display the value of the Bits and CMap properties.

4.2.2.3 Look at the value of the Event property of the MenuItems in Menu space ⎕SE.mb.file. The Select event of most MenuItems in the Session has a bracketed keyword that performs some specific function outside of APL. These calls are specially crafted for the Session object although you can make use of some of them in an aplication if they are appropriate.

4.2.2.4 What happens if you change the Event property of the ⎕SE.mb.file.load MenuItem from

```
Event ↳ ,⊂'onSelect' '[ChooseColors]'
```

to

```
Event ← 0 0
```

4.2.2.5 Replace the Tip on the wsload ToolButton with the word 'Open', and the Hint on the load MenuItem to 'Opens workspace file'.

§§ 4.2.3 Miscellaneous Properties and Methods

Some properties have an obvious meaning.

4.2.3.1 Try changing `⎕SE.Posn` or `⎕SE.Size`.

Other properties have a more session-specific rôle. For example, the `CurObj` property can be useful in session-related functionality such as the `ToolButton` version of **varChar**. `⎕SE.CurObj` reports the name currently under the input cursor. Its value is reported in the session `StatusBar` and its significance can be understood by clicking on various names in the session window and viewing the message reported in the `StatusBar`.

The effect of the FontObj property is fun although the session is not geared to non-fixed width or large true-type fonts. Version 11 includes a format bar in the session.

4.2.3.2 Write a function, such as that below, which loops round changing the `⎕SE.FontObj` property to each of the fonts in your `FontList` in turn. Trace `∇TestFonts∇`. Notice the effect in the session. Be prepared to close APL if the session becomes unusable (or hit Ctrl+Shift+Enter to get to line [9]).

```
      ∇ TestFonts;av;a;c
[1]     16 16⍴av←⎕AV[1+⎕NXLATE 0]
[2]     a←⊃¨#.FontList
[3]     a←a[⍋↑a]
[4]     c←1
[5]    Loop:→(c>⊃⍴a)⍴End
[6]     ⎕SE.FontObj←c⊃a
[7]     c+←1
[8]    →Loop
[9]   End:⎕SE.FontObj←'dyalog std'∇
```

```
      ##                                  ⍝ Parent of current space
```

`##` is analogous to the use of .. in DOS. It returns a ref as can be verified from

```
      ⎕P←##◊⎕NC'P'↳9
```

4.2.3.3 Consider a use for method `⎕SE.Create`, bearing in mind that runtime systems do have a session object, albeit initially empty.

4.2.3.4 Copy the function `∇DISPLAY∇` from workspace ..\ws\util.dws into the session object and set `⎕PATH` to `'⎕SE'`.

§ 4.3 Building the Session Object

§§ 4.3.1 Tracing `∇BUILD_SESSION∇`

4.3.1.1 Load the supplied workspace ..\ws\buildse.dws . Choose a country from the list

```
      Trans.CODES↳'UK' 'FR' 'IT' 'FI' 'US' 'GR'
```

and type and trace `∇BUILD_SESSION∇` with a Rarg of the language key of your choice; English, French, Italian, Finnish, American or German. (This choice would normally match your innate keyboard language and the language set in [Control Panel][Regional and Language Options][Languages][Details].) Watch the component objects being built from the contents of the current (.DSE) session file (registry parameter Session_File or as set in [Options][Configure][Session]).

4.3.1.2 Practice tracing parts of the code and running others. Find the quickest keyboard tracing route to some particular point in the code. (Set a `⎕STOP` at the target line using the mouse at the second column on the left side of the (`⎕ED`) edit window, first checking the editor menu items in [View].)

§§ 4.3.2 Foreign Language Support

4.3.2.1 Build Sessions in other languages. Marvel at the changed `Menu`s, `Hint`s and `Tip`s. Notice how standard menus and menu items make (almost blind) navigation possible. Any APL application may be made multi-lingual by the techniques employed in this workspace (see variable `Trans.STRINGS`). Note the [Session][Open] `MenuItem`, and also the [Log][Open] `MenuItem` which opens a DLF log file that is maintained separately from the DSE session file. The session file is, however, entangled with the contents of registry parameters which are described in the all-singing Dyalog APL *User Guide*.

§§ 4.3.3 Adding useful Extensions

4.3.3.1 Add a new [File][Open] `MenuItem` whose purpose is to tie an APL component file. Assign the `onSelect` property of a `FileBox` object (with DCF file filter) to the name of a function such as that below.

```
      onSelect←'selectOpen'

      ∇ selectOpen
[1]   '##.FB'⎕WC'Filebox' 'Open'
[2]   r←##.⎕DQ'FB'
[3]   ⍝ Tie file in r ∇
```

4.3.3.2 Add a `ToolButton` to the session that `DISPLAY`s the `CurObj`-*Value* pair in the session log.

Sales message: There is a version of ***varChar*** *for the APL development environment which makes use of* `⎕SE.CurObj`.

4.3.3.3 Please ask for the next module on **Control Structures** ☺.