

PLT Tools: DrScheme Extension Manual

Robert Bruce Findler (robby@plt-scheme.org)

301

Released December 2006

Copyright notice

Copyright ©1996-2005 PLT

Permission to make digital/hard copies and/or distribute this documentation for any purpose is hereby granted without fee, provided that the above copyright notice, author, and this permission notice appear in all copies of this documentation.

Send us your Web links

If you use any parts or all of the PLT Scheme package (software, lecture notes) for one of your courses, for your research, or for your work, we would like to know about it. Furthermore, if you use it and publicize the fact on some Web page, we would like to link to that page. Please drop us a line at *scheme@plt-scheme.org*. Evidence of interest helps the DrScheme Project to maintain the necessary intellectual and financial support. We appreciate your help.

Contents

1 This Manual	1
1.1 Thanks	1
2 Implementing DrScheme Tools	2
2.1 Adding Languages to DrScheme	4
2.1.1 Adding module-based Languages to DrScheme	4
2.1.2 Adding Arbitrary Languages to DrScheme	5
2.1.3 Language Extensions	5
2.2 Creating New Kinds of DrScheme Frames	5
2.3 Extending the Existing DrScheme Classes	5
2.4 Expanding the User's Program Text and Breaking	6
2.5 Editor Modes	6
3 Tools Reference	7
3.1 <code>drscheme:debug:profile-definitions-text-mixin</code>	7
3.2 <code>drscheme:debug:profile-interactions-text-mixin</code>	7
3.3 <code>drscheme:debug:profile-unit-frame-mixin</code>	7
3.4 <code>drscheme:frame:<%></code>	8
3.5 <code>drscheme:frame:basics<%></code>	9
3.6 <code>drscheme:frame:basics-mixin</code>	9
3.7 <code>drscheme:frame:mixin</code>	11
3.8 <code>drscheme:frame:name-message%</code>	12
3.9 <code>drscheme:get/extend:base-definitions-canvas%= (canvas:info-mixin (canvas:delegate-mixin canvas:color%))</code>	12
3.10 <code>drscheme:get/extend:base-definitions-text%= (drscheme:debug:profile-definitions-text-mixin drscheme:unit:definitions-text%)</code>	14

3.11 drscheme:get/extend:base-interactions-canvas% = (canvas:info-mixin (canvas:delegate-mixin canvas:color%))	14
3.12 drscheme:get/extend:base-interactions-text% = (drscheme:debug:profile-interactions-text drscheme:rep:text%)	15
3.13 drscheme:get/extend:base-tab% = drscheme:unit:tab%	15
3.14 drscheme:get/extend:base-unit-frame% = (drscheme:debug:profile-unit-frame-mixin drscheme:unit:frame%)	16
3.15 drscheme:language:language<%>	16
3.16 drscheme:language:module-based-language<%>	22
3.17 drscheme:language:module-based-language->language-mixin	25
3.18 drscheme:language:simple-module-based-language<%>	26
3.19 drscheme:language:simple-module-based-language%	27
3.20 drscheme:language:simple-module-based-language->module-based-language-mixin	28
3.21 drscheme:rep:context<%>	31
3.22 drscheme:rep:drs-bindings-keymap-mixin	33
3.23 drscheme:rep:text<%>	34
3.24 drscheme:rep:text%	34
3.25 drscheme:unit:definitions-canvas%	38
3.26 drscheme:unit:definitions-text<%>	40
3.27 drscheme:unit:definitions-text% = (drscheme:unit:program-editor-mixin (scheme:text-mixin (drscheme:rep:drs-bindings-keymap-mixin text:info%)))	40
3.28 drscheme:unit:frame<%>	41
3.29 drscheme:unit:frame% = (drscheme:frame:basics-mixin (drscheme:frame:mixin frame:searchable%))	42
3.30 drscheme:unit:interactions-canvas%	48
3.31 drscheme:unit:program-editor-mixin	49
3.32 drscheme:unit:tab<%>	50
3.33 drscheme:unit:tab%	52
3.34 DrScheme Tools Functions	53
3.35 Contract Helpers	68
Index	70

1. This Manual

This manual describes DrScheme's tools interface. It assumes familiarity with DrScheme, as described in *PLT DrScheme: Development Environment Manual*, the Framework, as described in *PLT Framework: GUI Application Framework*, MrEd as described in *PLT MrEd: Graphical Toolbox Manual*, and MzScheme as described in *PLT MzScheme: Language Manual*.

[build date: January 12, 2006]

1.1 Thanks

Thanks to Eli Barzilay, John Clements, Matthias Felleisen, Cormac Flanagan, Matthew Flatt, Max Halipern, Philippe Meunier, and Christian Queinnec, PLT at large, and many others for their feedback and help.

This manual was typeset using \LaTeX , \SIATeX , and \tex2page . Some typesetting macros were originally taken from Julian Smart's *Reference Manual for wxWindows 1.60: a portable C++ GUI toolkit*.

This manual was typeset on January 12, 2006.

2. Implementing DrScheme Tools

Tools are designed for major extensions in DrScheme's functionality. To extend the appearance or the functionality the DrScheme window (say, to annotate programs in certain ways, to add buttons to the DrScheme frame or to add additional languages to DrScheme) use a tool. The Static Debugger, the Syntax Checker, the Stepper, and the teaching languages are all implemented as tools.

Libraries are for extensions of DrScheme that only want to add new functions and other values bound in the users namespace. See the DrScheme manual for more information on constructing libraries.

Tools rely heavily on MzScheme's units. See units, §51 in *PLT MzScheme: Language Manual* for information on how to construct units. They also require understanding of libraries and collections, §16 in *PLT MzScheme: Language Manual*.

When DrScheme starts up, it looks for tools by reading fields in the **info.ss** file of each collection (Technically, DrScheme looks in a cache of the info.ss files contents created by setup-plt. Be sure to re-run setup-plt if you change the contents of the **info.ss** files). DrScheme checks for these fields:

```
tools (listof (listof string[subcollection-name]))
tool-names (listof (union #f string))
tool-icons (listof (union #f string[relative-pathname] (cons string[filename] (listof
  string[collection-name]))))
tool-urls (listof (union #f string[url]))
```

The *tools* field names a list of tools in this collection. Each tool is specified as a collection path, relative to the collection where the **info.ss** file resides. As an example, if there is only one tool named **tool.ss**, this suffices:

```
(define tools (list (list "tool.ss")))
```

If the *tool-icons* or *tool-names* fields are present, they must be the same length as *tools*. The *tool-icons* specifies the path to an icon for each tool and the name of each tool. If it is #f, no tool is shown. If it is a relative pathname, it must refer to a bitmap and if it is a list of strings, it is treated the same as the arguments to *lib*, inside *require*.

This bitmap and the name show up in the about box, Help Desk's bug report form, and the splash screen as the tool is loaded at DrScheme's startup.

Each of *tools* files must contain a module that provides *tool@*, which must be bound to a unit/sig, §51 in *PLT MzLib: Libraries Manual* The unit must import the `drscheme:tool^` signature, which is provided by the **tool.ss** library in the *drscheme* collection. The `drscheme:tool^` signature contains all of the names listed in this manual. The unit must export the `drscheme:tool-exports^` signature.

The `drscheme:tool-exports^` signature contains two names: *phase1* and *phase2*. These names must be bound to thunks. After all of the tools are loaded, all of the *phase1* functions are called and then all of the *phase2* functions are called. Certain primitives can only be called during the dynamic extent of those calls.

This mechanism is designed to support DrScheme's `drscheme:language:language<%>` extension capabilities. That is, this mechanism enables two tools to cooperate via new capabilities of languages. The first phase is used for adding functionality that each language must support and the second is used for creating instances of languages. As an example, a tool may require certain specialized language-specific information. It uses phase1 to extend the `drscheme:language:language<%>` interface and supply a default implementation of the interface extension. Then, other languages that are aware of the extension can supply non-default implementations of the additional functionality.

Phase 1 functions:

- `drscheme:language:extend-language-interface`
- `drscheme:unit:add-to-program-editor-mixin`

Phase 2 functions:

- `drscheme:language-configuration:add-language`
- `drscheme:language:get-default-mixin`
- `drscheme:language:get-language-extensions`

If the tools raises an error as it is loaded, invoked, or as the *phase1* or *phase2* thunks are called, DrScheme catches the error and displays a message box. Then, DrScheme continues to start up, without the tool.

For example, if the **info.ss** file in a collection contains:

```
(module info (lib "infotab.ss" "setup")
  (define name "Tool Name")
  (define tools (list (list "tool.ss"))))
```

then the same collection would be expected to contain a **tool.ss** file. It might contain something like this:

```
(module tool mzscheme
  (require (lib "tool.ss" "drscheme")
           (lib "mred.ss" "mred")
           (lib "unitsig.ss"))

  (provide tool@)

  (define tool@
    (unit/sig drscheme:tool-exports^
      (import drscheme:tool^)
      (define (phase1)
        (message-box "tool example" "phase1"))
      (define (phase2)
        (message-box "tool example" "phase2")))))
```

This tool just opens a window to indicate that it has been loaded.

2.1 Adding Languages to DrScheme

2.1.1 Adding module-based Languages to DrScheme

If a language can be implemented as a module (see module for details) and the standard language settings are sufficient, simply create an **info.ss** file in the collection where the module is saved. Include these definitions:

drscheme-language-modules This must be bound to a list of collection path specifications, one for each language in the collection. Each collection path specification is the quoted form of what might appear as an argument to `require`, using the `lib` argument.

drscheme-language-positions This must be bound to a list of language positions. Each language position corresponds to the position of the language in language dialog. Each language position is a list of strings whose length must be at least two.

drscheme-language-numbers This is optional. If present, it must be a list of a list of numbers. Each list corresponds to a single language from this collection. Each number indicates a sorting order in the language dialog for the corresponding string in **drscheme-language-positions**. If absent, it defaults to a list of zeros that has the same length as **drscheme-language-positions**. This will rarely be correct.

drscheme-language-one-line-summaries This is optional. If present, it must be a list of strings. Each string is displayed at the bottom of the language dialog when the corresponding language is selected.

drscheme-language-urls This is optional. If present, it must be a list whose elements are either strings or `#f`. Clicking the corresponding language's name in the interactions window opens a web browser to the url.

drscheme-language-readers This is optional. If present, it must be bound to a quoted list of module specifications (that is, a quoted version of the argument to `require`, except not plain strings). Each specification must be a module that exports a function named `read-syntax`. Each of these `read-syntax` functions must match MzScheme's `read-syntax` primitive's contract, but may read different concrete syntax.

The lists must have the same length.

As an example, the *Essentials of Programming Languages* language specification's **info.ss** looks like this:

```
(module info (lib "infotab.ss" "setup")
  (require (lib "string-constant.ss" "string-constants")))
(define name "EoPL Support")
(define drscheme-language-modules
  (list '("eopl-lang.ss" "eopl")))
(define drscheme-language-positions
  (list (list (string-constant teaching-languages)
             "Essentials of Programming Languages"))))
```

This **info.ss** file indicates that there is a single language in this collection. The module that implements the language is the **eopl-lang.ss** file in the **eopl** collection. Additionally, the language dialog will contain *Essentials of Programming Languages* as a potential language. The use of the string constant *teaching-languages* ensures that EoPL's language is placed properly in foreign language versions of DrScheme.

For collections that define multiple (related) languages, if the language-positions contain multiple strings, the languages whose leading strings match are grouped together. That is, if two languages have strings:

```
'("My Text" "First Language")
```

and

```
'("My Text" "Second Language")
```

the two languages will be grouped together in the language dialog.

2.1.2 Adding Arbitrary Languages to DrScheme

With some additional work, any language that can be compiled to MzScheme's language is supported by the tools interface, not just those that use standard configurations and `module`.

Each language is a class that implement the `drscheme:language:language<%>` interface. DrScheme also provides two simpler interfaces: `drscheme:language:module-based-language<%>` and `drscheme:language:simple-module-based-language<%>` and mixins, §3.2 in *PLT Framework: GUI Application Framework* `drscheme:language:simple-module-based-language->language-mixin` and `drscheme:language:module-based-language->language-mixin` that build implementations of `language^s` from these simpler interfaces.

Once you have an implementation of the `drscheme:language:language<%>` interface, call `drscheme:language-configure` to add the language to DrScheme.

Each language comes with its own type, called `settings`. This can be any type the language designer chooses, but to aid documentation, we call it `settings` here. The settings type is expected to contain parameters of the language, such as case sensitivity, etc. The implementor of the language provides a GUI so the user can configure the settings and all of the language's operations accept a setting. DrScheme maintains the current settings for each language.

2.1.3 Language Extensions

Some tools may require additional functionality from the `drscheme:language:language<%>` interface. The `drscheme:language:extend-language-interface` function and the `drscheme:language:get-default-mixin` mixin make this possible.

For example, the MrFlow tool expands programs, analyzes it and then displays sets of values for each program point. These sets of values should be rendered in the syntax of the language that MrFlow analyzes. Since MrFlow doesn't apriori know which languages are available, it can call `drscheme:language:extend-language-interface` to extend the `drscheme:language:language<%>` interface with a method for rendering sets of values and provide a default implementation of that method. Tools that know about MrFlow can then override the value rendering method to provide a language-specific implementation of value rendering. Additionally, since the `drscheme:language:get-default-mixin` adds the default implementation for the value-set rendering method, all languages at least have some form of value-set rendering.

2.2 Creating New Kinds of DrScheme Frames

Each frame in DrScheme has certain menus and functionality, most of which is achieved by using the framework. Additionally, there is one mixin that DrScheme provides to augment that. It is `drscheme:frame:basics-mixin`. Be sure to mix it into any new frame class that you add to DrScheme.

2.3 Extending the Existing DrScheme Classes

Each of the names:

- `drscheme:get/extend:extend-interactions-text`
- `drscheme:get/extend:extend-definitions-text`
- `drscheme:get/extend:extend-interactions-canvas`
- `drscheme:get/extend:extend-definitions-canvas`
- `drscheme:get/extend:extend-unit-frame`

- `drscheme:get/extend:extend-tab`

is bound to an extender function. In order to change the behavior of `drscheme`, you can derive new classes from the standard classes for the frame, texts, canvases. Each extender accepts a function as input. The function it accepts must take a class as its argument and return a class derived from that class as its result. For example:

```
(drscheme:get/extend:extend-interactions-text
  (lambda (super%
    (class super%
      (public method1)
      (define (method1 x) ...)
      ...)))
```

extends the `interactions text` class with a method named `rawscmmethod1`.

2.4 Expanding the User's Program Text and Breaking

Macro-expanding a program may involve arbitrary computation and requires the setup of the correct language. To aid this, DrScheme's tool interface provides `drscheme:eval:expand-program` to help. Use this method to extract the fully expanded program text in a particular language.

Because expanding the user's program may require DrScheme to evaluate arbitrary code that the user wrote, tools that expand the user's program should also allow the user to break the expansion. To help with this, the tools interfaces provides these methods: `enable-evaluation` and `disable-evaluation`. Since your tool will be expanding the program text, you should be both overriding `enable-evaluation` and `disable-evaluation` to disable your tool and calling them to ensure that only one expansion is happening at a time.

Finally, DrScheme provides the `set-breakables`, method. This method controls what behavior the Break button has.

2.5 Editor Modes

DrScheme provides support for multiple editor modes. Tools register modes via `drscheme:modes:add-mode`. Each mode is visible in the Modes submenu of the Edit menu. Initially, DrScheme only supports two modes: `scheme mode` and `text mode`.

DrScheme automatically selects a mode for each open file based on the file's extension. If the file ends with `.txt`, DrScheme uses `text mode`. Otherwise, DrScheme uses `Scheme mode`.

3. Tools Reference

3.1 drscheme:debug:profile-definitions-text-mixin

Domain: `drscheme:unit:definitions-text<%>`

Domain: (class->interface `text%`)

Implements: `drscheme:unit:definitions-text<%>`

```
- init args: [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]  
  line-spacing = 1.0 : non-negative real number  
  tab-stops = null : list of real numbers  
  auto-wrap = #f : boolean
```

The `line-spacing` argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about `tabstops`.

If `auto-wrap` is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

3.2 drscheme:debug:profile-interactions-text-mixin

Domain: `drscheme:rep:text<%>`

Implements: `drscheme:rep:text<%>`

3.3 drscheme:debug:profile-unit-frame-mixin

Domain: `drscheme:frame:<%>`

Domain: `drscheme:unit:frame<%>`

Implements: `drscheme:frame:<%>`

Implements: `drscheme:unit:frame<%>`

3.4 drscheme:frame:<%>

Extends: `frame:text-info<%>`

Extends: `frame:editor<%>`

Extends: `drscheme:frame:basics<%>`

`add-show-menu-items`

This method is called during the construction of the view menu. This method is intended to be overridden. It is expected to add other Show/Hide menu items to the show menu.

See also `get-show-menu`.

```
- (send a-drscheme:frame: add-show-menu-items show-menu) ⇒ void
  show-menu : (is-a?/c menu%)
```

Does nothing.

`get-show-menu`

returns the view menu, for use by the `update-shown` method.

See also `add-show-menu-items`.

The method (and others) uses the word `show` to preserve backwards compatibility from when the menu itself was named the Show menu.

```
- (send a-drscheme:frame: get-show-menu) ⇒ (instanceof menu%)
```

`not-running`

updates the status pane at the bottom of the window to show that evaluation is not taking place in the user's program.

```
- (send a-drscheme:frame: not-running) ⇒ void
```

`running`

updates the status pane at the bottom of the window to show that evaluation is taking place in the user's program.

```
- (send a-drscheme:frame: running) ⇒ void
```

update-shown

This method is intended to be overridden. It's job is to update the "View" menu to match the state of the visible windows. In the case of the standard DrScheme window, it change the menu items to reflect the visibility of the definitions and interaction `editor-canvas%`.

Call this method whenever the state of the show menu might need to change.

See also `get-show-menu`.

```
- (send a-drscheme:frame: update-shown) => void
  Does nothing.
```

3.5 drscheme:frame:basics<%>

Extends: `frame:standard-menus<%>`

This interface is the result of the `drscheme:frame:basics-mixin`

3.6 drscheme:frame:basics-mixin

Domain: `frame:standard-menus<%>`

Implements: `drscheme:frame:basics<%>`

Implements: `frame:standard-menus<%>`

Use this mixin to establish some common menu items across various DrScheme windows.

edit-menu:between-find-and-preferences

This method is called between the addition of the find menu-item and before the addition of the preferences menu-item to the edit-menu menu. Override it to add additional menus at that point.

```
- (send a-drscheme:frame:basics-mixin edit-menu:between-find-and-preferences void
  Adds a separator-menu-item%. Next, adds the "Keybindings" menu item to the edit menu. Finally, if the current-eventspace-has-standard-menus? procedure returns #f, creates another separator-menu-item%.
```

file-menu:between-open-and-revert

This method is called between the addition of the open menu-item and before the addition of the revert menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-drscheme:frame:basics-mixin file-menu:between-open-and-revert file-menu
  void
  file-menu : (is-a?/c menu%)
```

Adds an *Install .plt File...* menu item, which downloads and installs .plt files from the web, or installs them from the local disk. After that, calls the super method.

`file-menu:between-print-and-close`

This method is called between the addition of the print menu-item and before the addition of the close menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-drscheme:frame:basics-mixin file-menu:between-print-and-close file-menu
void
  file-menu : (is-a?/c menu%)
```

Calls the super method. Then, creates a menu item for multi-file searching. Finally, adds a `separator-menu-item%`.

`file-menu:new-callback`

This method is called when the new menu-item of the file-menu menu is selected.

```
- (send a-drscheme:frame:basics-mixin file-menu:new-callback item evt void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)
```

Opens a new, empty DrScheme window.

`file-menu:new-string`

The result of this method is the name of this menu.

```
- (send a-drscheme:frame:basics-mixin file-menu:new-string string
  Returns the empty string.
```

`file-menu:open-callback`

This method is called when the open menu-item of the file-menu menu is selected.

```
- (send a-drscheme:frame:basics-mixin file-menu:open-callback item evt void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%)
```

Calls `handler:edit-file`.

`file-menu:open-string`

The result of this method is the name of this menu.

```
- (send a-drscheme:frame:basics-mixin file-menu:open-string string
  Returns the empty string.
```

`get-additional-important-urls`

Each string in the result of this method is added as a menu item to DrScheme's "Related Web Sites" menu item. The first string is the name of the menu item and the second string is a url that, when the menu item is chosen, is sent to the user's browser.

```
- (send a-drscheme:frame:basics-mixin get-additional-important-urls (listof (list
  string string)))
```

Defaultly returns the empty list.

`help-menu:about-callback`

This method is called when the about menu-item of the help-menu menu is selected.

```
- (send a-drscheme:frame:basics-mixin help-menu:about-callback item evt void
  item : (instance (derived-from menu-item%))
  evt : (instance control-event%))
```

Opens an about box for DrScheme.

`help-menu:about-string`

The result of this method is the name of this menu.

```
- (send a-drscheme:frame:basics-mixin help-menu:about-string string
  Returns the string "DrScheme".
```

`help-menu:before-about`

This method is called before the addition of the about menu-item to the help-menu menu. Override it to add additional menus at that point.

```
- (send a-drscheme:frame:basics-mixin help-menu:before-about help-menu void
  help-menu : (instance menu%))
```

Adds the Help Desk menu item and the Welcome to DrScheme menu item.

`help-menu:create-about?`

The result of this method determines if the corresponding menu-item is created. Override this to control the creation of the menu-item.

```
- (send a-drscheme:frame:basics-mixin help-menu:create-about? boolean
  Returns #t.
```

3.7 drscheme:frame:mixin

Domain: `frame:text-info<%>`

Domain: `drscheme:frame:basics<%>`

Domain: `frame:editor<%>`

Implements: `frame:text-info<%>`

Implements: `drscheme:frame:basics<%>`

Implements: `drscheme:frame:<%>`

Implements: `frame:editor<%>`

Provides an implementation of `drscheme:frame:<%>`

3.8 drscheme:frame:name-message%

Superclass: `canvas%`

This class implements the little filename button in the top-right hand side of drscheme's frame.

```
- (make-object drscheme:frame:name-message% parent) => drscheme:frame:name-message%
object
  parent : (instance (implements area-container<%>))
```

set-message

Sets the names that the button shows.

```
- (send a-drscheme:frame:name-message set-message name short-name) => void
  name : (union string #f)
  short-name : string
```

The string `short-name` is the name that is shown on the button and `name` is shown when the button is clicked on, in a separate window. If `name` is `#f`, a message indicating that the file hasn't been saved is shown.

3.9 drscheme:get/extend:base-definitions-canvas% = (canvas:info-mixin (canvas:delegate-mixin canvas:color%))

```
drscheme:get/extend:base-definitions-canvas% = (canvas:info-mixin (canvas:delegate-mixin
canvas:color%))
```

```
- (new drscheme:get/extend:base-definitions-canvas% (parent _) [(editor _)] [(style
_) [(scrolls-per-page _) [(label _) [(wheel-step _) [(line-count _) [(horizontal-in
_) [(vertical-inset _) [(enabled _) [(vert-margin _) [(horiz-margin _) [(min-width
_) [(min-height _) [(stretchable-width _) [(stretchable-height _)]) => drscheme:get/ext
```

object

parent : *frame%*, *dialog%*, *panel%*, or *pane%* object
editor = #f : *text%* or *pasteboard%* object or #f
style = null : list of symbols in '(no-border control-border combo no-hscroll
no-vscroll hide-hscroll hide-vscroll auto-vscroll
auto-hscroll resize-corner deleted transparent)
scrolls-per-page = 100 : exact integer in [1, 10000]
label = #f : string (up to 200 characters) or #f
wheel-step = 3 : exact integer in [1, 10000] or #f
line-count = #f : exact integer in [1, 1000] or #f
horizontal-inset = 5 : exact integer in [0, 1000]
vertical-inset = 5 : exact integer in [0, 1000]
enabled = #t : boolean
vert-margin = 0 : exact integer in [0, 1000]
horiz-margin = 0 : exact integer in [0, 1000]
min-width = 0 : exact integer in [0, 10000]
min-height = 0 : exact integer in [0, 10000]
stretchable-width = #t : boolean
stretchable-height = #t : boolean

Passes all arguments to *super-init*.

- (make-object drscheme:get/extend:base-definitions-canvas% *parent editor style scrolls-per-page label wheel-step line-count horizontal-inset vertical-inset enabled vert-margin horiz-margin min-width min-height stretchable-width stretchable-height*) ⇒ *drscheme:get/extend:base-definitions-canvas%* object

parent : *frame%*, *dialog%*, *panel%*, or *pane%* object
editor = #f : *text%* or *pasteboard%* object or #f
style = null : list of symbols in '(no-border control-border combo no-hscroll
no-vscroll hide-hscroll hide-vscroll auto-vscroll
auto-hscroll resize-corner deleted transparent)
scrolls-per-page = 100 : exact integer in [1, 10000]
label = #f : string (up to 200 characters) or #f
wheel-step = 3 : exact integer in [1, 10000] or #f
line-count = #f : exact integer in [1, 1000] or #f
horizontal-inset = 5 : exact integer in [0, 1000]
vertical-inset = 5 : exact integer in [0, 1000]
enabled = #t : boolean
vert-margin = 0 : exact integer in [0, 1000]
horiz-margin = 0 : exact integer in [0, 1000]
min-width = 0 : exact integer in [0, 10000]
min-height = 0 : exact integer in [0, 10000]
stretchable-width = #t : boolean
stretchable-height = #t : boolean

Calls *super-new*, adding 'hide-hscroll to the style argument.

on-focus

Called when a window receives or loses the keyboard focus. If the argument is #t, the keyboard focus was received, otherwise it was lost.

Note that under X, keyboard focus can move to the menu bar when the user is selecting a menu item.

- (send *a-drscheme:get/extend:base-definitions-canvas* on-focus *on?*) ⇒ void

3.10. `drscheme:get/extend:base-definitions-text%` =
(`drscheme:debug:profile-definitions-text-mixin`
`drscheme:unit:definitions-text%`)

`on?` : boolean

When the focus is on, calls `make-searchable` with this.

3.10 `drscheme:get/extend:base-definitions-text%` = (`drscheme:debug:profile-def`
`drscheme:unit:definitions-text%`)

`drscheme:get/extend:base-definitions-text%` = (`drscheme:debug:profile-definitions-text-mixin`
`drscheme:unit:definitions-text%`)

- (new `drscheme:get/extend:base-definitions-text%` [(`line-spacing` _)] [(`tab-stops`
_)] [(`auto-wrap` _)]) ⇒ `drscheme:get/extend:base-definitions-text%` object
`line-spacing` = 1.0 : non-negative real number
`tab-stops` = null : list of real numbers
`auto-wrap` = #f : boolean

Passes all arguments to `super-init`.

3.11 `drscheme:get/extend:base-interactions-canvas%` = (`canvas:info-mixin`
(`canvas:delegate-mixin` `canvas:color%`))

`drscheme:get/extend:base-interactions-canvas%` = (`canvas:info-mixin` (`canvas:delegate-mixin`
`canvas:color%`))

- (new `drscheme:get/extend:base-interactions-canvas%` (`parent` _) [(`editor` _)] [(`style`
_)] [(`scrolls-per-page` _)] [(`label` _)] [(`wheel-step` _)] [(`line-count` _)] [(`horizontal-in`
_)] [(`vertical-inset` _)] [(`enabled` _)] [(`vert-margin` _)] [(`horiz-margin` _)] [(`min-width`
_)] [(`min-height` _)] [(`stretchable-width` _)] [(`stretchable-height` _)]) ⇒ `drscheme:get/ext`
object
`parent` : `frame%`, `dialog%`, `panel%`, or `pane%` object
`editor` = #f : `text%` or `pasteboard%` object or #f
`style` = null : list of symbols in '(no-border control-border combo no-hscroll
no-vscroll hide-hscroll hide-vscroll auto-vscroll
auto-hscroll resize-corner deleted transparent)
`scrolls-per-page` = 100 : exact integer in [1, 10000]
`label` = #f : string (up to 200 characters) or #f
`wheel-step` = 3 : exact integer in [1, 10000] or #f
`line-count` = #f : exact integer in [1, 1000] or #f
`horizontal-inset` = 5 : exact integer in [0, 1000]
`vertical-inset` = 5 : exact integer in [0, 1000]
`enabled` = #t : boolean
`vert-margin` = 0 : exact integer in [0, 1000]
`horiz-margin` = 0 : exact integer in [0, 1000]
`min-width` = 0 : exact integer in [0, 10000]
`min-height` = 0 : exact integer in [0, 10000]
`stretchable-width` = #t : boolean
`stretchable-height` = #t : boolean

Passes all arguments to `super-init`.

- (make-object `drscheme:get/extend:base-interactions-canvas%` `parent` `editor` `style`
`scrolls-per-page` `label` `wheel-step` `line-count` `horizontal-inset` `vertical-inset`

3. Tools Reference (drscheme:debug:profile-interactions-text-mixin drscheme:rep:text%)

```

enabled vert-margin horiz-margin min-width min-height stretchable-width stretchable-height
⇒ drscheme:get/extend:base-interactions-canvas% object
  parent : frame%, dialog%, panel%, or pane% object
  editor = #f : text% or pasteboard% object or #f
  style = null : list of symbols in '(no-border control-border combo no-hscroll
    no-vscroll hide-hscroll hide-vscroll auto-vscroll
    auto-hscroll resize-corner deleted transparent)
  scrolls-per-page = 100 : exact integer in [1, 10000]
  label = #f : string (up to 200 characters) or #f
  wheel-step = 3 : exact integer in [1, 10000] or #f
  line-count = #f : exact integer in [1, 1000] or #f
  horizontal-inset = 5 : exact integer in [0, 1000]
  vertical-inset = 5 : exact integer in [0, 1000]
  enabled = #t : boolean
  vert-margin = 0 : exact integer in [0, 1000]
  horiz-margin = 0 : exact integer in [0, 1000]
  min-width = 0 : exact integer in [0, 10000]
  min-height = 0 : exact integer in [0, 10000]
  stretchable-width = #t : boolean
  stretchable-height = #t : boolean
Calls super-new, adding 'hide-hscroll to the style argument.

```

on-focus

Called when a window receives or loses the keyboard focus. If the argument is #t, the keyboard focus was received, otherwise it was lost.

Note that under X, keyboard focus can move to the menu bar when the user is selecting a menu item.

```

- (send a-drscheme:get/extend:base-interactions-canvas on-focus on?) ⇒ void
  on? : boolean

```

When the focus is on, calls `make-searchable` with this.

3.12 drscheme:get/extend:base-interactions-text% = (drscheme:debug:profile-interactions-text-mixin drscheme:rep:text%)

```

drscheme:get/extend:base-interactions-text% = (drscheme:debug:profile-interactions-text-mixin
  drscheme:rep:text%)

```

```

- (new drscheme:get/extend:base-interactions-text% (context _)) ⇒ drscheme:get/extend:base-interactions-text%
  object
  context : (implements drscheme:rep:context<%>)

```

Passes all arguments to super-init.

3.13 drscheme:get/extend:base-tab% = drscheme:unit:tab%

```

drscheme:get/extend:base-tab% = drscheme:unit:tab%

```

```

- (new drscheme:get/extend:base-tab% ) ⇒ drscheme:get/extend:base-tab% object

```

Passes all arguments to super-init.

3.14 `drscheme:get/extend:base-unit-frame%` = (`drscheme:debug:profile-unit-frame-mixin` `drscheme:unit:frame%`)

`drscheme:get/extend:base-unit-frame%` = (`drscheme:debug:profile-unit-frame-mixin` `drscheme:unit:frame%`)

```
- (new drscheme:get/extend:base-unit-frame% [(parent _)] [(width _)] [(height
_) [(x _) [(y _) [(style _) [(enabled _) [(border _) [(spacing _) [(alignment
_) [(min-width _) [(min-height _) [(stretchable-width _) [(stretchable-height
_) [(filename _)] => drscheme:get/extend:base-unit-frame% object
  parent = #f : frame% object or #f
  width = #f : exact integer in [0, 10000] or #f
  height = #f : exact integer in [0, 10000] or #f
  x = #f : exact integer in [-10000, 10000] or #f
  y = #f : exact integer in [-10000, 10000] or #f
  style = null : list of symbols in '(no-resize-border no-caption no-system-menu
      mdi-parent mdi-child toolbar-button hide-menu-bar float
      metal)
  enabled = #t : boolean
  border = 0 : exact integer in [0, 1000]
  spacing = 0 : exact integer in [0, 1000]
  alignment = '(center top) : two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0 : exact integer in [0, 10000]
  min-height = 0 : exact integer in [0, 10000]
  stretchable-width = #t : boolean
  stretchable-height = #t : boolean
  filename : string?
```

Passes all arguments to `super-init`.

3.15 `drscheme:language:language<%>`

Implementations of this interface are languages that DrScheme supports.

See §2.1 for an overview of adding languages to DrScheme.

`config-panel`

This method used by the language configuration dialog to construct the "details" panel for this language. It accepts a parent panel and returns a `get/set` function that either updates the GUI to the argument or returns the settings for the current GUI.

```
- (send a-drscheme:language:language config-panel parent) => (case-i (-i settings) (set-
tings -i void))
  parent : (instanceof panel%)
```

`create-executable`

This method creates an executable in the given language. The `program-filename` is the name of the program to store in the executable and `executable-filename` is the name of a file where the executable goes.

See also `drscheme:language:create-module-based-stand-alone-executable` and `drscheme:language:crea`

```
- (send a-drscheme:language:language create-executable settings parent program-filename
  teachpack-cache) ⇒ void
  settings: settings
  parent: (union (instanceof dialog%) (instanceof frame%))
  program-filename: string
  teachpack-cache: drscheme:teachpack:teachpack-cache
```

default-settings

Specifies the default settings for this language.

```
- (send a-drscheme:language:language default-settings) ⇒ settings
```

default-settings?

Return #t if the input settings matches the default settings obtained via `default-settings`.

```
- (send a-drscheme:language:language default-settings? settings) ⇒ boolean
  settings: settings
```

first-opened

```
- (send a-drscheme:language:language first-opened) ⇒ void
```

This method is called when the language is initialized, but no program is run. It is called from the user's eventspace's main thread.

See also `initialize-console`.

front-end/complete-program

`front-end/complete-program` method reads, parses, and optionally compiles a program in the language. The first argument contains all of the data to be read (until eof) and the second argument is a value representing the source of the program (typically an editor, but may also be a string naming a file or some other value).

The third argument is the current settings for the language. The `front-end/complete-program` method is expected to return a thunk that is called repeatedly to get all of the expressions in the program. When all expressions have been read, the thunk is expected to return `eof`.

This method is only called for programs in the definitions window. Notably, it is not called for programs that are loaded or `eval`d. See `current-load` and `current-eval` for those.

This method is expected to raise an appropriate exception if the program is malformed, eg an `exn:syntax` or `exn:read`.

This is called on the user's thread, as is the thunk it returns.

Implementations of this method should not return fully expanded expressions, since there are two forms of expansion, using either `expand`, §12.6.1 in *PLT MzScheme: Language Manual* or `expand-top-level-with-compile-time-evals` and the use of the expanded code dictates which applies.

See also `front-end/interaction`.

```
- (send a-drscheme:language:language front-end/complete-program port line-col-offset
  settings teachpack-cache) => (-i (union sexp syntax eof))
  port: port
  line-col-offset: (union false? (list/p (union false? number?) (union false? number?) (union false? number?)))
  settings: settings
  teachpack-cache: drscheme:teachpack:teachpack-cache
```

The teachpacks in *teachpack-cache* have already been loaded and require'd in the toplevel namespace when this method is called.

The *line-col-offset* argument specifies the initial line, column, and offset for the text to be read from the port. If it is #f, a default value, (list 1 0 0) is used. See *read-syntax* for details on this argument. It should match *read-syntax*'s last argument.

front-end/interaction

This method is just like *front-end/complete-program* except that it is called with program fragments, for example the expressions entered in the interactions window. It is also used in other contexts by tools to expand single expressions.

```
- (send a-drscheme:language:language front-end/interaction port line-col-offset
  settings teachpack-cache) => (-i (union sexp syntax eof))
  port: port
  line-col-offset: (union false? (list/p (union false? number?) (union false? number?) (union false? number?)))
  settings: settings
  teachpack-cache: drscheme:teachpack:teachpack-cache
```

The *line-col-offset* argument specifies the initial line, column, and offset for the text to be read from the port. If it is #f, a default value, (list 1 0 0) is used. See *read-syntax* for details on this argument. It should match *read-syntax*'s last argument.

get-comment-character

Returns text to be used for the “Insert Large Letters” menu item in DrScheme. The first result is a prefix to be placed at the beginning of each line and the second result is a character to be used for each pixel in the letters.

```
- (send a-drscheme:language:language get-comment-character) => (values string? char?)
```

get-language-name

Returns the name of the language as shown in the REPL when executing programs in the language.

```
- (send a-drscheme:language:language get-language-name) => string
```

get-language-numbers

This method is used in a manner analogous to *get-language-position*.

Each element in the list indicates how the names at that point in dialog will be sorted. Names with lower numbers appear first. If two languages are added to DrScheme with the same strings (as given by the *get-language-position* method) the corresponding numbers returned by this method must be the same.

```
- (send a-drscheme:language:language get-language-numbers) ⇒ (cons number (listof
  number))
```

get-language-position

This method returns a list of strings that is used to organize this language with the other languages. Each entry in that list is a category or subcategory of the language and the last entry in the list is the name of the language itself. In the language dialog, each element in the list except for the last will be a nested turn down triangle on the left of the dialog. The final entry will be the name that the user can choose to select this language. Names that are the same will be combined into the same turndown entry.

For example, if one language's position is:

```
(list "General Category" "Specific Category" "My Language")
```

and another's is:

```
(list "General Category" "Specific Category" "My Other Language")
```

The language dialog will collapse the first two elements in the list, resulting in only a pair of nested turn-down triangles, not parallel pairs of nested turn-down triangles.

```
- (send a-drscheme:language:language get-language-position) ⇒ (cons string (listof
  string))
```

get-language-url

Returns a url for the language.

```
- (send a-drscheme:language:language get-language-url) ⇒ (union #f string)
```

If the result isn't #f, the name of the language is clickable in the interactions window and clicking takes you to this url.

get-one-line-summary

The result of this method is shown in the language dialog when the user selects this language.

```
- (send a-drscheme:language:language get-one-line-summary) ⇒ string
```

get-style-delta

The style delta that this method returns is used in the language dialog and the DrScheme REPL when the language's name is printed.

When it is \#f, no styling is used.

If the result is a list, each element is expected to be a list of three items, a style-delta, and two numbers. The style delta will be applied to the corresponding portion of the name.

```
- (send a-drscheme:language:language get-style-delta) ⇒ (union #f
  (instanceof style-delta%) (listof (list (instanceof style-delta%) number number)))
```

marshall-settings

Translates an instance of the settings type into a scheme object that can be written out to disk.

```
- (send a-drscheme : language : language marshall-settings settings) ⇒ writable
  settings : settings
```

on-execute

The `on-execute` method is called on DrScheme's eventspace's main thread before any evaluation happens during execution. Use this method to initialize MzScheme's parameters, §7.9 in *PLT MzScheme: Language Manual* for the user. When this function is called, the user's thread has already been created, as has its custodian. These parameters have been changed from the defaults in MzScheme:

- `current-custodian` is set to a new custodian.
- `current-namespace` has been set to a newly created empty namespace. This namespace has the following modules copied (with `namespace-attach-module`) from DrScheme's original namespace:
 - 'mzscheme
 - '(lib "mred.ss" "mred")
- `read-curly-brace-as-paren` is #t,
- `read-square-bracket-as-paren` is #t,
- The `port-write-handler` and `port-display-handler` have been set to procedures that call `pretty-print` and `pretty-display` instead of `write` and `display`. When `pretty-print` and `pretty-display` are called by these parameters, the `pretty-print-columns` parameter is set to 'infinity, so the output looks just like `write` and `display`. This is done so that special scheme values can be displayed as snips.
- The `current-print-covert-hook` is to a procedure so that `snip%`s are just returned directly to be inserted into the interactions `text%` object.
- The output and input ports are set to point to the interactions window with these parameters: `current-input-port`, `current-output-port`, and `current-error-port`.
- The `event-dispatch-handler` is set so that DrScheme can perform some initial setup and close down around the user's code.
- The `current-directory` and `current-load-relative-directory` are set to the directory where the definitions file is saved, or if it isn't saved, to the initial directory where DrScheme started up.
- The `snip-class-list`, returned by `get-the-snip-class-list` is initialized with all of the snipclasses in DrScheme's eventspace's snip-class-list.
- The `error-print-source-location` parameter is set to #f and the `error-display-handler` is set to a handler that creates an error message from the exception record, with font and color information and inserts that error message into the definitions window.

The `run-in-user-thread` arguments accepts thunks and runs them on the user's eventspace's main thread. These thunks must not raise an exceptions (or drscheme itself will get stuck). In addition, the output ports are not yet functioning, so print outs should be directed to the original drscheme output port, if necessary.

```
- (send a-drscheme:language:language on-execute settings run-in-user-thread) ⇒
void
  settings: settings
  run-in-user-thread: ((-¿ void) -¿ void)
```

order-manuals

Returns a sublist of its input, that specifies the manuals (and their order) to search in. The boolean result indicates if doc.txt files should be searched.

```
- (send a-drscheme:language:language order-manuals manuals) ⇒ (values (listof bytes?)
boolean?)
  manuals: (listof bytes?)
```

render-value

This method is just like [render-value/format](#) except that it is expected to put the entire value on a single line with no newline after the value.

```
- (send a-drscheme:language:language render-value value settings port) ⇒ void
  value: TST
  settings: settings
  port: port
```

render-value/format

This method is used to print values into a port, for display to a user. The final argument is a maximum width to use (in characters) when formatting the value.

This method is expected to format the value by inserting newlines in appropriate places and is expected to render a newline after the value.

See also [render-value](#).

```
- (send a-drscheme:language:language render-value/format value settings port width)
⇒ void
  value: TST
  settings: settings
  port: port
  width: (union #f number)
```

unmarshall-settings

Translates a Scheme value into a settings, returning #f if that is not possible.

```
- (send a-drscheme:language:language unmarshall-settings input) ⇒ (union settings
#f)
  input: writable
```

3.16 drscheme:language:module-based-language<%>

This interface is for languages that can be implemented with MzScheme modules.

Use the `drscheme:language:module-based-language->language-mixin` mixin to construct an implementation of `drscheme:language:language<%>` from an implementation of this interface.

:

- (send *a-drscheme:language:module-based-language* :) ⇒ procedure with the same type as `read-syntax`

This method must return a procedure that is used to read syntax from a port in the same manner as `read-syntax`. It is used as the reader for this language.

`config-panel`

This method is the same as `config-panel`.

- (send *a-drscheme:language:module-based-language* `config-panel` *parent*) ⇒ (case-
 λ (- λ *settings*) (*settings* - λ void))
parent : (instanceof `panel%`)

`default-settings`

This method is the same as `default-settings`.

- (send *a-drscheme:language:module-based-language* `default-settings`) ⇒ `settings`

`default-settings?`

This method is the same as `default-settings?`.

- (send *a-drscheme:language:module-based-language* `default-settings?` *settings*)
⇒ `boolean`
settings : `settings`

`get-init-code`

Returns a module in `sexpression` form that is used for creating executables. The module must provide a thunk, called `init-code`.

When either a stand-alone executable or a launcher is created, the module is required, and `init-code` is invoked. This procedure is expected to set up the environment, based on the settings.

- (send *a-drscheme:language:module-based-language* `get-init-code` *settings* *teachpack-cache*)
⇒ `sexp`
settings : `settings`
teachpack-cache : `drscheme:teachpack:teachpack-cache`

get-language-numbers

This method is the same as `get-language-numbers`.

```
- (send a-drscheme:language:module-based-language get-language-numbers) ⇒ (cons
  number (listof number))
```

get-language-position

This method is the same as `get-language-position`.

```
- (send a-drscheme:language:module-based-language get-language-position) ⇒ (cons
  string (listof string))
```

get-module

This method specifies the module that defines the language. It is used to initialize the user's namespace.

The result is expected to be the specification of a module except as value, ie quoted.

See also `get-transformer-module`.

```
- (send a-drscheme:language:module-based-language get-module) ⇒ s-expression
```

get-one-line-summary

The result of this method is shown in the language dialog when the user selects this language.

```
- (send a-drscheme:language:module-based-language get-one-line-summary) ⇒ string
```

get-transformer-module

This method specifies the module that defines the transformation language. It is used to initialize the transformer portion of the user's namespace.

The result is expected to be the specification of a module except as value, ie quoted.

See also `get-module`.

```
- (send a-drscheme:language:module-based-language get-transformer-module) ⇒ s-
  expression
```

marshall-settings

This method is the same as `marshall-settings`.

```
- (send a-drscheme:language:module-based-language marshall-settings settings)
  ⇒ writable
  settings: settings
```

on-execute

This method is the same as `on-execute`.

```
- (send a-drscheme:language:module-based-language on-execute settings run-in-user-thread
  => void
  settings: settings
  run-in-user-thread: ((-i void) -i void))
```

render-value

This method is the same as `render-value`.

```
- (send a-drscheme:language:module-based-language render-value value settings
  port port-write) => void
  value: TST
  settings: settings
  port: port
  port-write: (union #f ((instanceof snip%) -i void))
```

render-value/format

This method is the same as `render-value/format`.

```
- (send a-drscheme:language:module-based-language render-value/format value settings
  port port-write) => void
  value: TST
  settings: settings
  port: port
  port-write: (union #f ((instanceof snip%) -i void))
```

unmarshall-settings

This method is the same as `unmarshall-settings`.

```
- (send a-drscheme:language:module-based-language unmarshall-settings input) =>
  (union settings #f)
  input: writable
```

use-mred-launcher

This method is called when an executable is created to determine if the executable should use the mred or the mzscheme binary.

```
- (send a-drscheme:language:module-based-language use-mred-launcher) => boolean
```

use-namespace-require/copy?

The result of this method controls how the module is attached to the user's namespace. If the method returns `#t`, the mzscheme primitive `namespace-require/copy` is used and if it returns `#f`, `namespace-require` is used.

```
- (send a-drscheme:language:module-based-language use-namespace-require/copy?)
  => boolean
```

Defaultly returns #f.

3.17 `drscheme:language:module-based-language->language-mixin`

Domain: `drscheme:language:module-based-language<%>`

Implements: `drscheme:language:language<%>`

Implements: `drscheme:language:module-based-language<%>`

`front-end/complete-program`

`front-end/complete-program` method reads, parses, and optionally compiles a program in the language. The first argument contains all of the data to be read (until eof) and the second argument is a value representing the source of the program (typically an editor, but may also be a string naming a file or some other value).

The third argument is the current settings for the language. The `front-end/complete-program` method is expected to return a thunk that is called repeatedly to get all of the expressions in the program. When all expressions have been read, the thunk is expected to return `eof`.

This method is only called for programs in the definitions window. Notably, it is not called for programs that are loaded or `eval`d. See `current-load` and `current-eval` for those.

This method is expected to raise an appropriate exception if the program is malformed, eg an `exn:syntax` or `exn:read`.

This is called on the user's thread, as is the thunk it returns.

Implementations of this method should not return fully expanded expressions, since there are two forms of expansion, using either `expand`, §12.6.1 in *PLT MzScheme: Language Manual* or `expand-top-level-with-compile-time-evals` and the use of the expanded code dictates which applies.

See also `front-end/interaction`.

```
- (send a-drscheme:language:module-based-language->language-mixin front-end/complete-program
  port line-col-offset settings teachpack-cache (-i (union sexp syntax eof))
  port : port
  line-col-offset : (union false? (list/p (union false? number?) (union false? number?) (union false? number?)))
  settings : settings
  teachpack-cache : drscheme:teachpack:teachpack-cache
```

Reads a syntax object, from `input`. Does not use `settings`.

For languages that use these mixins, there is no difference between this method and `front-end/interaction`.

`front-end/interaction`

This method is just like `front-end/complete-program` except that it is called with program fragments, for example the expressions entered in the interactions window. It is also used in other contexts by tools to expand single expressions.

```
- (send a-drscheme:language:module-based-language->language-mixin front-end/interaction
    port line-col-offset settings teachpack-cache (-i (union sexp syntax eof))
    port : port
    line-col-offset : (union false? (list/p (union false? number?) (union false? number?) (union false? number?)))
    settings : settings
    teachpack-cache : drscheme:teachpack:teachpack-cache
```

Reads a syntax object, from `input`. Does not use `settings`.

For languages that use these mixins, there is no difference between this method and `front-end/complete-program`.

`get-language-name`

Returns the name of the language as shown in the REPL when executing programs in the language.

```
- (send a-drscheme:language:module-based-language->language-mixin get-language-name
    string
```

Returns the last element of the list returned by `get-language-position`.

`on-execute`

This method is the same as `on-execute`.

```
- (send a-drscheme:language:module-based-language->language-mixin on-execute settings
    run-in-user-thread void
    settings : settings
    run-in-user-thread : ((-i void) -i void)
```

Calls the super method.

Uses `namespace-require` to install the the result of `get-module` and Uses `namespace-transformer-require` to install the result of `get-transformer-module` into the user's namespace.

3.18 `drscheme:language:simple-module-based-language<%>`

This interface represents the bare essentials when defining a module-based language. Use the `drscheme:language:simple-module-based-language-mixin` to construct an implementation of `drscheme:language:module-based-language<%>` from an implementation of this interface.

The class `drscheme:language:simple-module-based-language%` provides an implementation of this interface.

`get-language-numbers`

Returns a list of numbers, whose length must be the same as the result of `get-language-position`. Each number indicates the sorted order of the language positions in the language dialog.

```
- (send a-drscheme:language:simple-module-based-language get-language-numbers)
⇒ (cons number (listof number))
```

get-language-position

This method is the same as `get-language-position`.

```
- (send a-drscheme:language:simple-module-based-language get-language-position)
⇒ (cons string (listof string))
```

get-module

This method specifies the module that defines the language.

This method replaces `front-end/complete-program` and `front-end/interaction`.

The result is expected to be the specification of a module except as value, ie quoted.

```
- (send a-drscheme:language:simple-module-based-language get-module) ⇒ s-expression
```

get-one-line-summary

The result of this method is shown in the language dialog when the user selects this language.

```
- (send a-drscheme:language:simple-module-based-language get-one-line-summary)
⇒ string
```

3.19 drscheme:language:simple-module-based-language%

Implements: `drscheme:language:simple-module-based-language<%>`

```
- (make-object drscheme:language:simple-module-based-language% module language-position
language-numbers one-line-summary documentation-reference) ⇒ drscheme:language:simple-m
object
  module : s-expression
  language-position : (cons string (listof string))
  language-numbers = (map (lambda (x) 0) language-position) : (cons number (listof number))
  one-line-summary = " " : string
  documentation-reference = #f : (union #f something-else)
```

The init args are used as the results of the `get-module` and `get-language-position` methods

get-language-numbers

Returns a list of numbers, whose length must be the same as the result of `get-language-position`. Each number indicates the sorted order of the language positions in the language dialog.

```
- (send a-drscheme:language:simple-module-based-language get-language-numbers)
⇒ (cons number (listof number))
```

returns the corresponding init arg.

get-language-position

This method is the same as [get-language-position](#).

```
- (send a-drscheme:language:simple-module-based-language get-language-position)
  => s-expression
  returns the corresponding init arg.
```

get-module

This method specifies the module that defines the language.

This method replaces [front-end/complete-program](#) and [front-end/interaction](#).

The result is expected to be the specification of a module except as value, ie quoted.

```
- (send a-drscheme:language:simple-module-based-language get-module) => (cons
  string (listof string))
  returns the corresponding init arg.
```

get-one-line-summary

The result of this method is shown in the language dialog when the user selects this language.

```
- (send a-drscheme:language:simple-module-based-language get-one-line-summary)
  => string
  returns the corresponding initialization argument.
```

3.20 drscheme:language:simple-module-based-language->module-based-language-mixin

Domain: [drscheme:language:simple-module-based-language<%>](#)

Implements: [drscheme:language:module-based-language<%>](#)

Implements: [drscheme:language:simple-module-based-language<%>](#)

This mixin uses a struct definition for its settings:

```
(define-struct drscheme:language:simple-settings (case-sensitive printing-style fraction-style
;; case-sensitive : boolean
;; printing-style : (union 'constructor 'quasiquote 'write)
;; fraction-style : (union 'mixed-fraction 'mixed-fraction-e 'repeating-decimal 'repeating-decimal)
;; show-sharing : boolean
;; insert-newlines : boolean
;; annotations : boolean
```

3. Tools Reference `language:simple-module-based-language->module-based-language-mixin`

The settings in this structure reflect the settings show in the language configuration dialog for languages constructed with this mixin. The first controls the input for the language. The rest specify printing controls for the language. The style `'write` is the default style, used in the MzScheme REPL. The sharing field determines if cycles and sharing in values are displayed when the value is rendered. The insert newlines field determines if values in the repl are formatted with `write` style-line printouts, or with `pretty-print` multi-line printouts.

`config-panel`

This method is the same as `config-panel`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  config-panel parent (case-i (-i settings) (settings -i void))
  parent : (instanceof panel%))
```

Constructs a configuration panel that lets the user configure all of the settings for this language.

See also §3.20 for details of the simple-settings structure, this mixins `settings` type.

`default-settings`

This method is the same as `default-settings`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  default-settings settings)
```

The defaults for the settings are

- `case-sensitive` is `#f`
- `printing-style` is `'write`
- `show-sharing` is `#f`
- `insert-newlines` is `#t`

See also §3.20 for details of the simple-settings structure, this mixins `settings` type.

`default-settings?`

This method is the same as `default-settings?`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  default-settings? settings boolean
  settings: settings)
```

`get-init-code`

Returns a module in sexpression form that is used for creating executables. The module must provide a thunk, called `init-code`.

When either a stand-alone executable or a launcher is created, the module is required, and `init-code` is invoked. This procedure is expected to set up the environment, based on the settings.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  get-init-code settings teachpack-cache sexpression
  settings: settings
  teachpack-cache: drscheme:teachpack:teachpack-cache)
```

Creates an s-expression of a module that sets the `current-inspector`, `read-case-sensitive`, and `error-value->string` parameters. Additionally, it may load `errortrace`, if debugging is enabled.

`get-transformer-module`

This method specifies the module that defines the transformation language. It is used to initialize the transformer portion of the user's namespace.

The result is expected to be the specification of a module except as value, ie quoted.

See also `get-module`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  get-transformer-module s-expression
```

Returns 'mzscheme.

`marshall-settings`

This method is the same as `marshall-settings`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  marshall-settings settings writable
  settings: settings
```

Constructs a vector from the structure.

See also §3.20 for details of the simple-settings structure, this mixins `settings` type.

`on-execute`

This method is the same as `on-execute`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  on-execute settings run-in-user-thread void
  settings: settings
  run-in-user-thread: ((-i void) -i void)
```

Sets the case sensitivity of the language.

Sets the structure inspector to a new inspector, saving the original inspector for use during printing.

Sets the `global-port-print-handler` to print based on the settings structure, but without any newlines.

If debugging is enabled, it sets the `current-eval handler` to one that annotates each evaluated program with debugging annotations. Additionally, it sets the `error-display-handler` to show the debugging annotations when an error is raised.

See also §3.20 for details of the simple-settings structure, this mixin's `settings` type.

`render-value`

This method is the same as `render-value`.

```
- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  render-value value settings port port-write void
```

```

value : TST
settings : settings
port : port
port-write : (union #f ((instanceof snip%) -¿ void))

```

Translates the value to a string, based on the settings.

Restores a super struct inspector to render structs properly. (See also `on-execute`)

See also §3.20 for details of the simple-settings structure, this mixins `settings` type.

`render-value/format`

This method is the same as `render-value/format`.

```

- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  render-value/format value settings port port-write void
  value : TST
  settings : settings
  port : port
  port-write : (union #f ((instanceof snip%) -¿ void))

```

Translates the value to a string, based on the settings.

Restores a super struct inspector to render structs properly. (See also `on-execute`)

See also §3.20 for details of the simple-settings structure, this mixins `settings` type.

`unmarshall-settings`

This method is the same as `unmarshall-settings`.

```

- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  unmarshall-settings input (union #f settings)
  input : writable

```

Builds a settings structure from the vector, or `#f` if the vector doesn't match the types of the structure.

See also §3.20 for details of the simple-settings structure, this mixins `settings` type.

`use-mred-launcher`

This method is called when an executable is created to determine if the executable should use the mred or the mzscheme binary.

```

- (send a-drscheme:language:simple-module-based-language->module-based-language-mixin
  use-mred-launcher boolean

```

Returns `#t`.

3.21 drscheme:rep:context<%>

Objects that match this interface provide all of the services that the `drscheme:rep:text%` class needs to connect with it's context.

`clear-annotations`

Call this method to clear any annotations in the text before executing or analyzing or other such activities that should process the program.

Tools that annotate the program text should override this method to clear annotations.

DrScheme calls this method before a program is run (via the Run button).

```
- (send a-drscheme:rep:context clear-annotations) ⇒ void
```

Clears any error highlighting in the definitions window.

`disable-evaluation`

Call this method to disable evaluation GUI evaluation while some evaluation (or expansion) is taking place on another thread.

Override this method if you add a GUI-based mechanism for initiating evaluation in the frame.

This method is also called when the user switches tabs.

See also [enable-evaluation](#).

```
- (send a-drscheme:rep:context disable-evaluation) ⇒ void
```

`enable-evaluation`

This method must disable the GUI controls that start user-sponsored evaluation. It is called once the user starts some evaluation to ensure that only one evaluation proceeds at a time.

It is also called when the user switches tabs.

See also [disable-evaluation](#).

```
- (send a-drscheme:rep:context enable-evaluation) ⇒ void
```

`ensure-rep-shown`

```
- (send a-drscheme:rep:context ensure-rep-shown rep) ⇒ void
  rep: (is-a?/c drscheme:rep:text<%>)
```

This method is called to force the rep window to be visible when, for example, an error message is put into the rep. Also ensures that the appropriate tab is visible, if necessary.

`get-breakables`

Returns the last values passed to [set-breakables](#).

```
- (send a-drscheme:rep:context get-breakables) ⇒ (values (union thread #f) (union custodian #f))
```

`get-directory`

The result of this method is used as the initial directory for the user's program to be evaluated in.

```
- (send a-drscheme:rep:context get-directory) ⇒ (union string #f)
```

`needs-execution`

This method should return an explanatory string when the state of the program that the repl reflects has changed. It should return `#f` otherwise.

```
- (send a-drscheme:rep:context needs-execution) ⇒ (union string? false/c)
```

`reset-offer-kill`

The break button typically offers to kill if it has been pushed twice in a row. If this method is called, however, it ignores any prior clicks.

```
- (send a-drscheme:rep:context reset-offer-kill) ⇒ void
```

`set-breakables`

Calling this method with a thread and a custodian means that the next time the break button is clicked, it will either break the thread or shutdown the custodian.

See also [get-breakables](#).

```
- (send a-drscheme:rep:context set-breakables thread custodian) ⇒ void
  thread : (union thread #f)
  custodian : (union custodian #f)
```

`update-running`

```
- (send a-drscheme:rep:context update-running running?) ⇒ void
  running? : boolean
```

This method should update some display in the gui that indicates whether or not evaluation is currently proceeding in the user's world.

3.22 drscheme:rep:drs-bindings-keymap-mixin

Domain: `editor:keymap<%>`

Implements: `editor:keymap<%>`

This mixin adds some drscheme-specific keybindings to the editor it is mixed onto.

`get-keymaps`

The keymaps returned from this method are chained to this `editor<%>`'s keymap.

- (send *a-drscheme:rep:drs-bindings-keymap-mixin* get-keymaps (listof (instanceof `keymap%`)))
Calls the super method and adds in a keymap with the drscheme-specific keybindings:
 - `f5` - Run
 - `c:x;o` - toggles the focus between the definition and interactions windows.

3.23 `drscheme:rep:text<%>`

3.24 `drscheme:rep:text%`

Implements: `drscheme:rep:text<%>`

This class implements a read-eval-print loop for DrScheme. User submitted evaluations in DrScheme are evaluated asynchronously, in an eventspace created for the user. No evaluations carried out by this class affect the implementation that uses it.

- (make-object `drscheme:rep:text%` *context*) ⇒ `drscheme:rep:text%` object
context : (implements `drscheme:rep:context<%>`)

`after-delete`

Called after a given range is deleted from the editor (and after the `display` is refreshed; use `on-delete` and `begin-edit-sequence` to avoid extra refreshes when `after-delete` modifies the editor).

See also `can-delete?` and `on-edit-sequence`.

No internals locks are set when this method is called.

- (send *a-drscheme:rep:text* after-delete *orig super-call start len*) ⇒ void
orig : ???
super-call : (??? -i ???)
start : ??
len : ??

Resets any error highlighting in this editor.

`after-insert`

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

- (send *a-drscheme:rep:text* after-insert *orig super-call start len*) ⇒ void
orig : ???

```
super-call : (??? -i ???)
start : ??
len : ??
```

Resets any error highlighting in this editor.

display-results

```
- (send a-drscheme:rep:text display-results results) ⇒ void
  results : (list-of TST)
```

This displays each of the elements of *results* in the interactions window, except those elements of *results* that are void. Those are just ignored.

do-many-evals

Use this function to evaluate code or run actions that should mimic the user's interactions. For example, DrScheme uses this function to evaluate expressions in the definitions window and expressions submitted at the prompt.

```
- (send a-drscheme:rep:text do-many-evals run-loop) ⇒ void
  run-loop : (((-i void) -i void) -i void)
```

The function *run-loop* is called. It is expected to loop, calling it's argument with a thunk that corresponds to the user's evaluation. It should call it's argument once for each expression the user is evaluating. It should pass a thunk to it's argument that actually does the users's evaluation.

do-many-text-evals

This function evaluates all of the expressions in a text.

```
- (send a-drscheme:rep:text do-many-text-evals text start end complete-program?)
  ⇒ void
  text : a text% object
  start : int
  end : int
  complete-program? : boolean
```

It evaluates all of the expressions in *text* starting at *start* and ending at *end*, calling *do-many-evals* to handle the evaluation.

The *complete-program?* argument determines if the *front-end/complete-program* method or the *front-end/interaction* method is called.

get-error-range

Indicates the highlighted error range. The state for the error range is shared across all instances of this class, so there can only be one highlighted error region at a time.

```
- (send a-drscheme:rep:text get-error-range) ⇒ (union #f (list (instanceof text:basic%
  number number)))
```

If *#f*, no region is highlighted. If a list, the first element is the editor where the range is highlighted and the second and third are the beginning and ending regions, respectively.

`get-user-custodian`

This is the custodian controlling the user's program.

```
- (send a-drscheme:rep:text get-user-custodian) ⇒ (union #f custodian)
```

`get-user-eventspace`

This is the user's eventspace. The result of `get-user-thread` is the main thread of this eventspace.

```
- (send a-drscheme:rep:text get-user-eventspace) ⇒ (union #f eventspace)
```

`get-user-language-settings`

Returns the user's language-settings for the most recently run program. Consider using `get-next-settings` instead, since the user may have selected a new language since the program was last run.

```
- (send a-drscheme:rep:text get-user-language-settings) ⇒ language-settings
```

`get-user-namespace`

Returns the user's namespace. This method returns a new namespace each time Run is clicked.

```
- (send a-drscheme:rep:text get-user-namespace) ⇒ (union #f namespace)
```

`get-user-thread`

This method returns the thread that the users code runs in. It returns a different result, each time the user runs the program.

It is `#f` before the first time the user click on the Run button or the evaluation has been killed.

This thread has all of its parameters initialized according to the settings of the current execution. See parameters, §7.9 in *PLT MzScheme: Language Manual* for more information about parameters.

```
- (send a-drscheme:rep:text get-user-thread) ⇒ (union #f thread)
```

`highlight-errors`

Call this method to highlight errors associated with this repl. See also `reset-highlighting`, and `highlight-errors/exn`.

This method highlights a series of dis-contiguous ranges in the editor.

It puts the caret at the location of the first error.

```
- (send a-drscheme:rep:text highlight-errors locs) ⇒ void
  locs : (listof (list (instance (implements text:basic<*>)) small-integer small-integer))
```

`highlight-errors/exn`

```
- (send a-drscheme:rep:text highlight-errors/exn exn) ⇒ void
  exn: exn
```

Highlights the errors associated with the `exn` (only syntax and read errors – does not extract any information from the continuation marks)

See also `highlight-errors`.

`initialize-console`

```
- (send a-drscheme:rep:text initialize-console) ⇒ void
```

This inserts the “Welcome to DrScheme” message into the interactions buffer, calls `reset-console`, `insert-prompt`, and `clear-undos`.

Once the console is initialized, this method calls `first-opened`. Accordingly, this method should not be called to initialize a REPL when the user’s evaluation is imminent. That is, this method should be called when new tabs or new windows are created, but not when the Run button is clicked.

`insert-prompt`

```
- (send a-drscheme:rep:text insert-prompt) ⇒ void
```

Inserts a new prompt at the end of the text.

`kill-evaluation`

This method is called when the user chooses the kill menu item.

```
- (send a-drscheme:rep:text kill-evaluation) ⇒ void
```

`on-close`

This method is called when an editor is closed. See also `can-close?` and `close`.

```
- (send a-drscheme:rep:text on-close) ⇒ void
```

Calls `shutdown`.

Calls the super method.

`queue-output`

This method queues `thunks` for `drscheme`’s eventspace in a special output-related queue.

```
- (send a-drscheme:rep:text queue-output thunk) ⇒ void
  thunk: (-? void?)
```

`reset-console`

```
- (send a-drscheme:rep:text reset-console) ⇒ void
```

Kills the old eventspace, and creates a new parameterization for it.

reset-highlighting

This method resets the highlighting being displayed for this repl. See also: [highlight-errors](#), and [highlight-errors/exn](#).

```
- (send a-drscheme:rep:text reset-highlighting) ⇒ void
```

run-in-evaluation-thread

This function runs it's arguments in the user evaluation thread. This thread is the same as the user's eventspace main thread.

See also [do-many-evals](#).

```
- (send a-drscheme:rep:text run-in-evaluation-thread f) ⇒ void
  f : (-> void)
```

Calls *f*, after switching to the user's thread.

shutdown

Shuts down the user's program and all windows. Reclaims any resources the program allocated. It is expected to be called from DrScheme's main eventspace thread.

```
- (send a-drscheme:rep:text shutdown) ⇒ void
```

wait-for-io-to-complete

This waits for all pending IO in the rep to finish and then returns.

This method must only be called from the main thread in DrScheme's eventspace

```
- (send a-drscheme:rep:text wait-for-io-to-complete) ⇒ void
```

wait-for-io-to-complete/user

This waits for all pending IO in the rep to finish and then returns.

This method must only be called from the main thread in the user's eventspace

```
- (send a-drscheme:rep:text wait-for-io-to-complete/user) ⇒ void
```

3.25 drscheme:unit:definitions-canvas%

Superclass: [editor-canvas%](#)

Initializes the visibility of the save button.

```
- (new drscheme:unit:definitions-canvas% (parent _) [(editor _)] [(style _)] [(scrolls-per
  _)] [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
```

```

_)] [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
_)] [(stretchable-width _)] [(stretchable-height _)] => drscheme:unit:definitions-canvas%
object
  parent = frame%, dialog%, panel%, or pane% object
  editor = #f : text% or pasteboard% object or #f
  style = null : list of symbols in '(no-border control-border combo no-hscroll
    no-vscroll hide-hscroll hide-vscroll auto-vscroll
    auto-hscroll resize-corner deleted transparent)
  scrolls-per-page = 100 : exact integer in [1, 10000]
  label = #f : string (up to 200 characters) or #f
  wheel-step = 3 : exact integer in [1, 10000] or #f
  line-count = #f : exact integer in [1, 1000] or #f
  horizontal-inset = 5 : exact integer in [0, 1000]
  vertical-inset = 5 : exact integer in [0, 1000]
  enabled = #t : boolean
  vert-margin = 0 : exact integer in [0, 1000]
  horiz-margin = 0 : exact integer in [0, 1000]
  min-width = 0 : exact integer in [0, 10000]
  min-height = 0 : exact integer in [0, 10000]
  stretchable-width = #t : boolean
  stretchable-height = #t : boolean

```

If a canvas is initialized with #f for *editor*, install an editor later with `set-editor`.

The *style* list can contain the following flags:

- 'no-border — omits a border around the canvas
- 'control-border — gives the canvas a border that is like a `text-field%` control
- 'combo — gives the canvas a combo button that is like a `combo-field%` control; this style is intended for use with 'control-border, 'hide-hscroll, and 'hide-vscroll
- 'no-hscroll — disallows horizontal scrolling and hides the horizontal scrollbar
- 'no-vscroll — disallows vertical scrolling and hides the vertical scrollbar
- 'hide-hscroll — allows horizontal scrolling, but hides the horizontal scrollbar
- 'hide-vscroll — allows vertical scrolling, but hides the vertical scrollbar
- 'auto-hscroll — automatically hides the horizontal scrollbar when unneeded (unless 'no-hscroll or 'hide-hscroll is specified)
- 'auto-vscroll — automatically hides the vertical scrollbar when unneeded (unless 'no-vscroll or 'hide-vscroll is specified)
- 'resize-corner — leaves room for a resize control at the canvas's bottom right when only one scrollbar is visible
- 'deleted — creates the canvas as initially hidden and without affecting *parent*'s geometry; the canvas can be made active later by calling *parent*'s `add-child` method
- 'transparent — the canvas is “erased” before an update using it's parent window's background

While vertical scrolling of text editors is based on lines, horizontal scrolling and pasteboard vertical scrolling is based on a fixed number of steps per horizontal page. The *scrolls-per-page* argument sets this value.

If provided, the *wheel-step* argument is passed on to the `wheel-step` method. The default wheel step can be overridden globally though the '|MrEd:wheelStep| preference; see “Preferences” (§12 in *PLT MrEd: Graphical Toolbox Manual*).

If *line-count* is not #f, it is passed on to the `set-line-count` method.

If *horizontal-inset* is not 5, it is passed on to the `horizontal-inset` method. Similarly, if *vertical-inset* is not 5, it is passed on to the `vertical-inset` method.

For information about the *enabled* argument, see `window<%>`. For information about the *horiz-margin* and *vert-margin* arguments, see `subarea<%>`. For information about the *min-width*, *min-height*, *stretchable-width*, and *stretchable-height* arguments, see `area<%>`.

3.26 drscheme:unit:definitions-text<%>

This interface is implemented by the definitions text.

`after-set-next-settings` (*augmentable only*)

Called when the next settings changes. See also `get-next-settings`.

```
- (send a-drscheme:unit:definitions-text after-set-next-settings language-settings)
  => void
   language-settings: language-settings
```

`get-next-settings`

This method returns the language-settings that will be used when the user next clicks Run in this DrScheme window.

```
- (send a-drscheme:unit:definitions-text get-next-settings) => language-settings
```

`get-tab`

Returns the editor's enclosing tab.

```
- (send a-drscheme:unit:definitions-text get-tab) => (instanceof drscheme:unit:tab%)
```

3.27 drscheme:unit:definitions-text% = (drscheme:unit:program-editor-mixin (scheme:text-mixin (drscheme:rep:drs-bindings-keymap-mixin text:info%)))

```
drscheme:unit:definitions-text% = (drscheme:unit:program-editor-mixin (scheme:text-mixin
  (drscheme:rep:drs-bindings-keymap-mixin text:info%)))
```

Extends: `drscheme:unit:definitions-text<%>`

```
- (new drscheme:unit:definitions-text% [(line-spacing _)] [(tab-stops _)] [(auto-wrap
  _)]) => drscheme:unit:definitions-text% object
   line-spacing = 1.0: non-negative real number
   tab-stops = null: list of real numbers
   auto-wrap = #f: boolean
```

Passes all arguments to `super-init`.

`set-filename`

Set the path name for the file to be saved from or reloaded into this editor. This method is also called when the filename changes through any method (such as `load-file`).

```
- (send a-drscheme:unit:definitions-text set-filename filename temporary?) =>
  void
```

filename : path or #f
temporary? = #f : boolean

Calls `update-save-message`.

`set-modified`

Sets the modified state of the editor. Usually, the state is changed automatically after an insertion, deletion, or style change by calling this method. (This method is also called when the modification state changes through *any* method.) This method is usually not called when the state of the flag is not changing.

See also `is-modified?` and `on-snip-modified`.

```
- (send a-drscheme:unit:definitions-text set-modified modified?) => void
   modified? : boolean
```

Calls `update-save-button`.

3.28 drscheme:unit:frame<%>

`get-current-tab`

Returns the currently active tab.

```
- (send a-drscheme:unit:frame get-current-tab) => (is-a?/c drscheme:unit:tab<%>)
```

`get-definitions-canvas`

```
- (send a-drscheme:unit:frame get-definitions-canvas) => (instanceof (derivedfrom drscheme:unit:definitions-canvas))
```

This canvas is the canvas containing the `get-definitions-text`. It is initially the top half of the drscheme window.

This canvas defaults to a `drscheme:unit:definitions-canvas%` object, but if you change the `drscheme:get/extend:extend-definitions-canvas` procedure, it will use the class in the parameter to create the canvas.

`get-definitions-text`

```
- (send a-drscheme:unit:frame get-definitions-text) => (instanceof (derivedfrom drscheme:unit:definitions-text))
```

Calls result of `get-current-tab`'s `get-ints` method.

`get-interactions-canvas`

```
- (send a-drscheme:unit:frame get-interactions-canvas) => (instanceof (derivedfrom drscheme:unit:interactions-canvas))
```

This canvas is the canvas containing the `get-interactions-text`. It is initially the bottom half of the drscheme window.

This canvas defaults to a `drscheme:unit:interactions-canvas%` object, but if you use the `drscheme:get/extend:extend-interactions-canvas` procedure, it will use the extended class to create the canvas.

`get-interactions-text`

- (`send a-drscheme:unit:frame get-interactions-text`) ⇒ (`instanceof` (`derivedfrom` `drscheme:rep:text%`))
 Calls result of `get-current-tab`'s `get-defs` method.

`get-special-menu`

Returns the "Special" menu.

- (`send a-drscheme:unit:frame get-special-menu`) ⇒ (`is-a?/c menu%`)

`on-tab-change` (*augmentable only*)

Called after a new tab becomes the selected tab in the frame.

- (`send a-drscheme:unit:frame on-tab-change from-tab to-tab`) ⇒ `void`
`from-tab`: (`is-a?/c drscheme:unit:tab<%>`)
`to-tab`: (`is-a?/c drscheme:unit:tab<%>`)

The `from-tab` argument is the previously selected tab, and the `to-tab` argument is the newly selected tab.

3.29 `drscheme:unit:frame%` = (`drscheme:frame:basics-mixin` (`drscheme:frame:mixin` `frame:searchable%`))

`drscheme:unit:frame%` = (`drscheme:frame:basics-mixin` (`drscheme:frame:mixin` `frame:searchable%`))

Extends: `drscheme:unit:frame<%>`

This frame inserts the Scheme and Language menus into the menu bar as it is initialized.

- (`new drscheme:unit:frame%` [(`parent` `_`)] [(`width` `_`)] [(`height` `_`)] [(`x` `_`)] [(`y` `_`)]
 [(`style` `_`)] [(`enabled` `_`)] [(`border` `_`)] [(`spacing` `_`)] [(`alignment` `_`)] [(`min-width`
`_`)] [(`min-height` `_`)] [(`stretchable-width` `_`)] [(`stretchable-height` `_`)] (`filename`
`_`) ⇒ `drscheme:unit:frame%` object
`parent` = #f: `frame%` object or #f
`width` = #f: exact integer in [0, 10000] or #f
`height` = #f: exact integer in [0, 10000] or #f
`x` = #f: exact integer in [-10000, 10000] or #f
`y` = #f: exact integer in [-10000, 10000] or #f
`style` = `null`: list of symbols in '(no-resize-border no-caption no-system-menu
 mdi-parent mdi-child toolbar-button hide-menu-bar float
 metal)
`enabled` = #t: boolean
`border` = 0: exact integer in [0, 1000]
`spacing` = 0: exact integer in [0, 1000]
`alignment` = '(center top): two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
`min-width` = 0: exact integer in [0, 10000]
`min-height` = 0: exact integer in [0, 10000]
`stretchable-width` = #t: boolean

```
stretchable-height = #t : boolean
filename : string?
```

Passes all arguments to super-init.

```
- (new drscheme:unit:frame% [(parent _)] [(width _)] [(height _)] [(x _)] [(y _)]
  [(style _)] [(enabled _)] [(border _)] [(spacing _)] [(alignment _)] [(min-width
  _)] [(min-height _)] [(stretchable-width _)] [(stretchable-height _)] (filename
  _)) => drscheme:unit:frame% object
  parent = #f : frame% object or #f
  width = #f : exact integer in [0, 10000] or #f
  height = #f : exact integer in [0, 10000] or #f
  x = #f : exact integer in [-10000, 10000] or #f
  y = #f : exact integer in [-10000, 10000] or #f
  style = null : list of symbols in '(no-resize-border no-caption no-system-menu
    mdi-parent mdi-child toolbar-button hide-menu-bar float
    metal)
  enabled = #t : boolean
  border = 0 : exact integer in [0, 1000]
  spacing = 0 : exact integer in [0, 1000]
  alignment = '(center top) : two-element list: 'left, 'center, or 'right and 'top, 'center, or 'bottom
  min-width = 0 : exact integer in [0, 10000]
  min-height = 0 : exact integer in [0, 10000]
  stretchable-width = #t : boolean
  stretchable-height = #t : boolean
  filename : string?
```

Passes all arguments to super-init.

add-show-menu-items

This method is called during the construction of the view menu. This method is intended to be overridden. It is expected to add other Show/Hide menu items to the show menu.

See also [get-show-menu](#).

```
- (send a-drscheme:unit:frame add-show-menu-items show-menu) => void
  show-menu : (is-a?/c menu%)
```

Adds the “Show Definitions”, “Show Interactions” and “Show Contour” menu items.

break-callback

This method is called when the user clicks on the break button or chooses the break menu item.

```
- (send a-drscheme:unit:frame break-callback) => void
```

Breaks the user’s evaluation started by the Run button (or possibly a queued callback in the user’s eventspace).

change-to-file

```
- (send a-drscheme:unit:frame change-to-file file) => void
  file : string
```

Loads this file into this already created frame. In normal DrScheme use, this method is only called if this is the first frame opened and no editing has occurred. It should be safe to call this at anytime, however.

`edit-menu:between-select-all-and-find`

This method is called between the addition of the select-all menu-item and before the addition of the find menu-item to the edit-menu menu. Override it to add additional menus at that point.

- (`send a-drscheme:unit:frame edit-menu:between-select-all-and-find`) ⇒ void
Adds the "Split" and "Collapse" menu items.

`ensure-defs-shown`

Ensures that the definitions window is visible.

- (`send a-drscheme:unit:frame ensure-defs-shown`) ⇒ void

`ensure-rep-shown`

- (`send a-drscheme:unit:frame ensure-rep-shown`) ⇒ void
Shows the interactions window

`execute-callback`

This method is called when the user clicks on the Run button or chooses the Run menu item.

- (`send a-drscheme:unit:frame execute-callback`) ⇒ void
It calls `ensure-rep-shown` and then it calls `do-many-text-evals` passing in the result of `get-interactions-text` and its entire range, unless the first two characters are "#!" in which case, it skips the first line.

`file-menu:between-open-and-revert`

This method is called between the addition of the open menu-item and before the addition of the revert menu-item to the file-menu menu. Override it to add additional menus at that point.

- (`send a-drscheme:unit:frame file-menu:between-open-and-revert`) ⇒ void
Calls the super method and adds a `separator-menu-item%` to the menu.

`file-menu:between-print-and-close`

This method is called between the addition of the print menu-item and before the addition of the close menu-item to the file-menu menu. Override it to add additional menus at that point.

- (`send a-drscheme:unit:frame file-menu:between-print-and-close`) ⇒ void
Adds a menu item for printing the interactions.

`file-menu:between-save-as-and-print`

This method is called between the addition of the save-as menu-item and before the addition of the print menu-item to the file-menu menu. Override it to add additional menus at that point.

```
- (send a-drscheme:unit:frame file-menu:between-save-as-and-print) ⇒ void
```

Adds a submenu that contains various save options:

- save definitions as text
- save interactions
- save interactions as
- save interactions as text

and adds a separator item.

`file-menu:print-string`

The result of this method is the name of this menu.

```
- (send a-drscheme:unit:frame file-menu:print-string) ⇒ void  
  returns "Definitions"
```

`file-menu:save-as-string`

The result of this method is the name of this menu.

```
- (send a-drscheme:unit:frame file-menu:save-as-string) ⇒ void  
  Returns "Definitions".
```

`file-menu:save-string`

The result of this method is the name of this menu.

```
- (send a-drscheme:unit:frame file-menu:save-string) ⇒ void  
  Returns "Definitions".
```

`get-break-button`

Returns the break button. Mostly used for test suites.

```
- (send a-drscheme:unit:frame get-break-button) ⇒ (instanceof button%)
```

`get-button-panel`

This panel goes along the top of the drscheme window and has buttons for important actions the user frequently executes.

A tool can add a button to this panel to make some new functionality easily accessible to the user.

See also `mrlib's bitmap-label-maker`.

- (`send a-drscheme:unit:frame get-button-panel`) ⇒ (`instanceof horizontal-panel%`)

`get-canvas`

Returns the canvas used to display the `editor<%>` in this frame.

- (`send a-drscheme:unit:frame get-canvas`) ⇒ (`instanceof editor-canvas%`)
Returns the result of `get-definitions-canvas`.

`get-canvas%`

The result of this method is used to create the canvas for the `editor<%>` in this frame.

- (`send a-drscheme:unit:frame get-canvas%`) ⇒ (`instanceof (derived-from canvas%)`)
Returns the result of `drscheme:get/extend:get-definitions-canvas`.

`get-definitions/interactions-panel-parent`

This method is provided so that tools can add `area-container<%>`s to the `drscheme` frame. Override this method so that it returns a child of the super-classes's result and insert new children inbetween.

- (`send a-drscheme:unit:frame get-definitions/interactions-panel-parent`) ⇒ (`instanceof vertical-panel%`)
Returns the result of `get-area-container`
- (`send a-drscheme:unit:frame get-definitions/interactions-panel-parent`) ⇒ `void`

`get-editor`

Returns the editor in this frame.

- (`send a-drscheme:unit:frame get-editor`) ⇒ (`instanceof editor<%>`)
Returns the result of `get-definitions-text`.

`get-editor%`

The result of this class is used to create the `editor<%>` in this frame.

Override this method to specify a different editor class.

- (`send a-drscheme:unit:frame get-editor%`) ⇒ (`instanceof (derived-from editor<%>)`)
Returns the result of `drscheme:get/extend:get-definitions-text`.

`get-execute-button`

Returns the Run button. Mostly used for test suites.

- (`send a-drscheme:unit:frame get-execute-button`) ⇒ (`instanceof button%`)

`get-text-to-search`

Override this method to specify which text to search.

- (`send a-drscheme:unit:frame get-text-to-search`) ⇒ a `text:searching%` object
returns the text that is active in the last canvas passed to `make-searchable`

`make-searchable`

- (`send a-drscheme:unit:frame make-searchable canvas`) ⇒ void
`canvas`: a `drscheme:unit:interactions-canvas%` object
stores the canvas, until `get-text-to-search` is called.

`on-close`

Called just before the window is closed (e.g., by the window manager). This method is *not* called by `show`.

See also `can-close?`.

- (`send a-drscheme:unit:frame on-close`) ⇒ void
Sends the result of `get-interactions-text` the `shutdown` and `on-close` methods.
Calls the super method.

`on-size`

Called when the window is resized. The window's new size (in pixels) is provided to the method. The size values are for the entire window, not just the client area.

- (`send a-drscheme:unit:frame on-size width height`) ⇒ void
`width`: exact integer in [0, 10000]
`height`: exact integer in [0, 10000]
Updates the preferences for the window width and height so next time a `drscheme` window is opened, it will be this width and height.

`still-untouched?`

determines if the definitions window has not been modified. Used in conjunction with `change-to-file`.

- (`send a-drscheme:unit:frame still-untouched?`) ⇒ boolean
Returns `#t` if the buffer is empty, it has not been saved and it is unmodified.

`update-save-button`

- (`send a-drscheme:unit:frame update-save-button modified?`) ⇒ void
`modified?`: boolean
This method hides or shows the save button, based on the `modified?` argument.

If the save button has not been created yet, it remembers the *modified?* argument as an initial visibility for the save button.

This method is called by the `set-modified` method.

update-save-message

```
- (send a-drscheme:unit:frame update-save-message name) ⇒ void
  name : string
```

Updates the save message on the drscheme frame. This method is called by the `set-filename` method.

update-shown

This method is intended to be overridden. It's job is to update the "View" menu to match the state of the visible windows. In the case of the standard DrScheme window, it change the menu items to reflect the visibility of the definitions and interaction `editor-canvas%`.

Call this method whenever the state of the show menu might need to change.

See also `get-show-menu`.

```
- (send a-drscheme:unit:frame update-shown) ⇒ void
```

Updates the interactions, definitions, and contour menu items based on the contents of the windows.

3.30 drscheme:unit:interactions-canvas%

```
- (new drscheme:unit:interactions-canvas% (parent _) [(editor _)] [(style _)] [(scrolls-per-
_) [(label _)] [(wheel-step _)] [(line-count _)] [(horizontal-inset _)] [(vertical-inset
_) [(enabled _)] [(vert-margin _)] [(horiz-margin _)] [(min-width _)] [(min-height
_) [(stretchable-width _)] [(stretchable-height _)]) ⇒ drscheme:unit:interactions-canvas-
```

object

parent = `frame%`, `dialog%`, `panel%`, or `pane%` object

editor = #f : `text%` or `pasteboard%` object or #f

style = null : list of symbols in '(no-border control-border combo no-hscroll
no-vscroll hide-hscroll hide-vscroll auto-vscroll
auto-hscroll resize-corner deleted transparent)

scrolls-per-page = 100 : exact integer in [1, 10000]

label = #f : string (up to 200 characters) or #f

wheel-step = 3 : exact integer in [1, 10000] or #f

line-count = #f : exact integer in [1, 1000] or #f

horizontal-inset = 5 : exact integer in [0, 1000]

vertical-inset = 5 : exact integer in [0, 1000]

enabled = #t : boolean

vert-margin = 0 : exact integer in [0, 1000]

horiz-margin = 0 : exact integer in [0, 1000]

min-width = 0 : exact integer in [0, 10000]

min-height = 0 : exact integer in [0, 10000]

stretchable-width = #t : boolean

stretchable-height = #t : boolean

Passes all arguments to `super-init`.

3.31 drscheme:unit:program-editor-mixin

Domain: `editor:basic<%>`

Domain: (class->interface `text%`)

Implements: `editor:basic<%>`

This mixes in the ability to reset the highlighting for error message when the user modifies the buffer. Use it for editors that have program text where errors can occur.

```
- init args: [(line-spacing _)] [(tab-stops _)] [(auto-wrap _)]
  line-spacing = 1.0 : non-negative real number
  tab-stops = null : list of real numbers
  auto-wrap = #f : boolean
```

The `line-spacing` argument sets the additional amount of space (in DC units) inserted between each line in the editor when the editor is displayed. This spacing is included in the reported height of each line.

See `set-tabs` for information about `tabstops`.

If `auto-wrap` is true, then auto-wrapping is enabled via `auto-wrap`.

A new `keymap%` object is created for the new editor. See also `get-keymap` and `set-keymap`.

A new `style-list%` object is created for the new editor. See also `get-style-list` and `set-style-list`.

`after-delete`

Called after a given range is deleted from the editor (and after the `display` is refreshed; use `on-delete` and `begin-edit-sequence` to avoid extra refreshes when `after-delete` modifies the editor).

See also `can-delete?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-drscheme:unit:program-editor-mixin after-delete start len void
  start : number
  len : number
```

Calls the super method.

Resets an error highlighting.

`after-insert`

Called after `items` are inserted into the editor (and after the `display` is refreshed; use `on-insert` and `begin-edit-sequence` to avoid extra refreshes when `after-insert` modifies the editor).

See also `can-insert?` and `on-edit-sequence`.

No internals locks are set when this method is called.

```
- (send a-drscheme:unit:program-editor-mixin after-insert start len void
    start: number
    len: number
```

Calls the super method.

Resets an error highlighting.

3.32 `drscheme:unit:tab<%>`

Extends: `drscheme:rep:context<%>`

`break-callback`

This method is called when the break button is clicked and this tab is the active tab.

```
- (send a-drscheme:unit:tab break-callback) ⇒ void
```

By default, breaks any evaluation that may be happening at this point.

`can-close?` (*augmentable only*)

This method is called to determine if it is okay to close this tab.

```
- (send a-drscheme:unit:tab can-close?) ⇒ boolean
```

Calls the definitions text's and interactions text's `can-close?` method.

`disable-evaluation`

Call this method to disable evaluation GUI evaluation while some evaluation (or expansion) is taking place on another thread.

Override this method if you add a GUI-based mechanism for initiating evaluation in the frame.

This method is also called when the user switches tabs.

See also `enable-evaluation`.

```
- (send a-drscheme:unit:tab disable-evaluation) ⇒ void
```

Disables the Run button, and the Run menu item and `locks` the interactions window, and the definitions window.

`enable-evaluation`

This method must disable the GUI controls that start user-sponsored evaluation. It is called once the user starts some evaluation to ensure that only one evaluation proceeds at a time.

It is also called when the user switches tabs.

See also `disable-evaluation`.

```
- (send a-drscheme:unit:tab enable-evaluation) ⇒ void
```

Enables the Run button, and the Run menu item and **locks** the interactions window and the definitions window.

get-breakables

Returns the last values passed to **set-breakables**.

```
- (send a-drscheme:unit:tab get-breakables) ⇒ (values (union thread #f) (union custodian #f))
```

get-defs

This text is initially the bottom half of the drscheme window and contains the users interactions with the REPL.

This text defaults to a **drscheme:rep:text%** object, but if you use the **drscheme:get/extend:extend-interactions-text** procedure, it will use the extended class to create the text.

```
- (send a-drscheme:unit:tab get-defs) ⇒ (is-a?/c drscheme:unit:definitions-text<%>)
```

get-directory

The result of this method is used as the initial directory for the user's program to be evaluated in.

```
- (send a-drscheme:unit:tab get-directory) ⇒ (union string #f)
```

This is the directory that the file is saved in, or the directory DrScheme started up in, if the file has not been saved.

get-enabled

Indicates if evaluation is currently enabled in this tab. Evaluation is typically disabled when some evaluation is already running (in another thread).

```
- (send a-drscheme:unit:tab get-enabled) ⇒ boolean
```

get-frame

Returns the frame that this tab is inside.

```
- (send a-drscheme:unit:tab get-frame) ⇒ (is-a?/c drscheme:unit:frame%)
```

get-ints

This text is initially the top half of the drscheme window and contains the users program.

This text defaults to a **text%** object, but if you change **drscheme:get/extend:extend-interactions-text** procedure, it will use the extended class to create the text.

```
- (send a-drscheme:unit:tab get-ints) ⇒ (is-a?/c drscheme:rep:text%)
```

`is-current-tab?`

Indicates if this tab is the currently active tab.

```
- (send a-drscheme:unit:tab is-current-tab?) ⇒ boolean
```

`on-close` (*augmentable only*)

This method is called when the tab is closed.

```
- (send a-drscheme:unit:tab on-close) ⇒ void
```

Calls the definitions text's `on-close` and interactions text's `on-close` methods.

`reset-offer-kill`

The break button typically offers to kill if it has been pushed twice in a row. If this method is called, however, it ignores any prior clicks.

```
- (send a-drscheme:unit:tab reset-offer-kill) ⇒ void
```

`set-breakables`

Calling this method with a thread and a custodian means that the next time the break button is clicked, it will either break the thread or shutdown the custodian.

See also `get-breakables`.

```
- (send a-drscheme:unit:tab set-breakables thread custodian) ⇒ void
  thread: (union thread #f)
  custodian: (union custodian #f)
```

3.33 `drscheme:unit:tab%`

Implements: `drscheme:unit:tab<%>`

The base class that implements the tab's functionality.

```
- (make-object drscheme:unit:tab%) ⇒ drscheme:unit:tab% object
```

`clear-annotations`

Call this method to clear any annotations in the text before executing or analyzing or other such activities that should process the program.

Tools that annotate the program text should override this method to clear annotations.

DrScheme calls this method before a program is run (via the Run button).

- (send a-drscheme:unit:tab clear-annotations) ⇒ void

Clears any error highlighting.

3.34 DrScheme Tools Functions

drscheme:debug:add-prefs-panel

- (drscheme:debug:add-prefs-panel) ⇒ void?

Adds the profiling preferences panel.

drscheme:debug:get-cm-key

- (drscheme:debug:get-cm-key) ⇒ any

Returns a key used with *continuation-mark-set->list*. The continuation mark set attached to an exception record for the user's program may use this mark. If it does, each mark on the continuation is the same type as the input to `drscheme:debug:open-and-highlight-in-file`.

drscheme:debug:hide-backtrace-window

- (drscheme:debug:hide-backtrace-window) ⇒ void?

Hides the backtrace window.

drscheme:debug:make-debug-error-display-handler

- (drscheme:debug:make-debug-error-display-handler oedh) ⇒ (string? (union any/c exn?)
.-> . any)
oedh : (string? (union any/c exn?) .-> . any)

This function implements an error-display-handler in terms of another error-display-handler.

This function is designed to work in conjunction with `drscheme:debug:make-debug-eval-handler`.

See also MzScheme's MzLinkmz:p:error-display-handlererror-display-handler parameter.

If the current-error-port is the definitions window in drscheme, this error handler inserts some debugging annotations, calls *oedh*, and then highlights the source location of the runtime error.

drscheme:debug:make-debug-eval-handler

- (drscheme:debug:make-debug-eval-handler odeh) ⇒ (any/c .-> . any/c)
odeh : (any/c .-> . any/c)

This function implements an eval-handler in terms of another eval-handler.

This function is designed to work in conjunction with `drscheme:debug:make-debug-error-display-handler`.

See also MzScheme's MzLinkmz:p:eval-handlereval-handler parameter.

The resulting eval-handler expands and annotates the input expression and then passes it to the input eval-handler, unless the input expression is already compiled, in which case it just hands it directly to the input eval-handler.

`drscheme:debug:open-and-highlight-in-file`

- (`drscheme:debug:open-and-highlight-in-file` *debug-info*) \Rightarrow void?
debug-info : srcloc?

This function opens a DrScheme to display *debug-info*. The first element in the cons indicates where the file is and the two number indicate a range of text to show.

See also `drscheme:debug:get-cm-key`.

`drscheme:debug:profiling-enabled`

- (`drscheme:debug:profiling-enabled` *enabled?*) \Rightarrow void?
enabled? : boolean?
- (`drscheme:debug:profiling-enabled`) \Rightarrow boolean?

A parameter that controls if profiling information is recorded.

Defaults to `\#f`.

Only applies if `drscheme:debug:make-debug-eval-handler` has been added to the eval handler.

`drscheme:debug:show-backtrace-window`

- (`drscheme:debug:show-backtrace-window` *error-message* *dis*) \Rightarrow void?
error-message : string?
dis : (listof any/c)

Shows the backtrace window you get when clicking on the bug in DrScheme's REPL.

The *error-message* argument is the text of the error, *dis* is the debug information, extracted from the continuation mark in the exception record, using `drscheme:debug:get-cm-key`.

`drscheme:debug:show-error-and-highlight`

- (`drscheme:debug:show-error-and-highlight` *msg* *exn* *highlight-errors*) \Rightarrow any
msg : string?
exn : (union any/c exn?)
highlight-errors : ((listof srcloc?) (union false/c (listof (list/c (is-a?/c text%) number? number?))) . -> . any)

The first two arguments are the same as the arguments to the error-display-handler. This function prints the error message to the current-error-port, like the default error-display-handler and also calls *highlight-errors* to do error highlighting. It is be passed the stack trace for the error message.

This function should be called on the same thread/eventspace where the error happened.

`drscheme:eval:build-user-eventspace/custodian`

- (`drscheme:eval:build-user-eventspace/custodian` *language-settings* *init* *kill-termination*)
 \Rightarrow (values eventspace? custodian?)
language-settings : drscheme:language-configuration:language-settings?
init : (-> void?)
kill-termination : (-> void?)

This function creates a custodian and an eventspace (on the new custodian) to expand the user's program. It does not kill this custodian, but it can safely be shutdown (with `custodian-shutdown-all`) after the expansion is finished.

It initializes the user's eventspace's main thread with several parameters:

- `current-custodian` is set to a new custodian.
- In addition, it calls `drscheme:eval:set-basic-parameters`.

The `language-settings` argument is the current language and its settings. See `drscheme:language-configuration` for details on that structure.

If the program is associated with a DrScheme frame, get the frame's language settings from the `get-next-settings` method of `drscheme:unit:definitions-text<%>`. Also, the most recently chosen language in the language dialog is saved via the framework's preferences. Apply `preferences:get` to `drscheme:language-configuration:get-settings-preferences-symbol` for that `language-setting`.

The `init` argument is called after the user's parameters are all set, but before the program is run. It is called on the user's thread. The `current-directory` and `current-load-relative-directory` parameters are not set, so if there are appropriate directories, the `init` argument is a good place to set them.

The `kill-termination` argument is called when the main thread of the eventspace terminates, no matter if the custodian was shutdown, or the thread was killed. This procedure is also called when the thread terminates normally. This procedure is called from a new, dedicated thread (*i. e.*, not the thread created to do the expansion, nor the thread that `drscheme:eval:build-user-eventspace/custodian` was called from.)

`drscheme:eval:expand-program`

- ```
- (drscheme:eval:expand-program input language-settings eval-compile-time-part?
 init kill-termination iter) => void?
 input : (union port? drscheme:language:text/pos?)
 language-settings : drscheme:language-configuration:language-settings?
 eval-compile-time-part? : boolean?
 init : (-> void?)
 kill-termination : (-> void?)
 iter : ((union eof-object? syntax? (cons/c string? any/c)) (-> any) . -> . any)
```

Use this function to expand the contents of the definitions window for use with external program processing tools.

This function uses `drscheme:eval:build-user-eventspace/custodian` to build the user's environment. The arguments `language-settings`, `init`, and `kill-termination` are passed to `drscheme:eval:build-user-eventspace/custodian`.

The `input` argument specifies the source of the program.

The `eval-compile-time-part?` argument indicates if `awscmexpand`, §12.6.1 in *PLT MzScheme: Language Manual* is called or if `expand-top-level-with-compile-time-evals` is called when the program is expanded. Roughly speaking, if your tool will evaluate each expression itself by calling `eval`, §14.1 in *PLT MzScheme: Language Manual* then pass `#f`. Otherwise, if your tool just processes the expanded program, be sure to pass `#t`.

This function calls `front-end/complete-program` to expand the program.

The first argument to `iter` is the expanded program (represented as syntax) or eof. The `iter` argument is called for each expression in the expanded program and once more with eof, unless an error is raised during expansion. It is called from the user's thread. If an exception is raised during expansion of the user's program, `iter` is not called. Consider setting the exception-handler during `init` to handle this situation.

The second argument to `iter` is a thunk that continues expanding the rest of the contents of the definitions window. If the first argument to `iter` was eof, this argument is just the primitive void.

See also `drscheme:eval:expand-program/multiple`.

`drscheme:eval:expand-program/multiple`

- ```
- (drscheme:eval:expand-program/multiple language-settings eval-compile-time-part?
```

```

init kill-termination) ⇒ ((union port? drscheme:language:text/pos?) ((union eof-object? syntax?
(cons/c string? any/c)) (-> any) . -> . any) boolean? . -> . void?)
  language-settings: drscheme:language-configuration:language-settings?
  eval-compile-time-part?: boolean?
  init: (-> void?)
  kill-termination: (-> void?)

```

This function is just like `drscheme:eval:expand-program` except that it is curried and the second application can be used multiple times. Use this function if you want to initialize the user's thread (and namespace, etc) once but have program text that comes from multiple sources.

The extra boolean argument to the result function determines if `front-end/complete-program` or `front-end/interaction` is called.

`drscheme:eval:get-snip-classes`

```
- (drscheme:eval:get-snip-classes) ⇒ (listof (is-a?/c snip-class%))
```

Returns a list of all of the snipclasses in the current eventspace

`drscheme:eval:set-basic-parameters`

```
- (drscheme:eval:set-basic-parameters snipclasses) ⇒ void?
  snipclasses: (listof (is-a?/c snip-class%))
```

sets the parameters that are shared between the repl's initialization and `drscheme:eval:build-user-eventspace/custodian`

Specifically, it sets these parameters:

- `current-namespace` has been set to a newly created empty namespace. This namespace has the following modules copied (with `namespace-attach-module`) from DrScheme's original namespace:
 - * `'mzscheme`
 - * `'(lib "mred.ss" "mred")`
- `read-curly-brace-as-paren` is #t,
- `read-square-bracket-as-paren` is #t,
- `error-print-width` is set to 250.
- `current-ps-setup` is set to a newly created `ps-setup%` object.
- The `exit-handler` is set to a parameter that kills the user's custodian.
- The `snip-class-list`, returned by `get-the-snip-class-list` is initialized with all of the snipclasses in DrScheme's eventspace's `snip-class-list`.

`drscheme:eval:traverse-program/multiple`

```

- (drscheme:eval:traverse-program/multiple language-settings init kill-termination)
⇒ ((union port? drscheme:language:text/pos?) ((union eof-object? syntax? (cons/c string? any/c)) (-> any) .
-> . any) boolean? . -> . void?)
  language-settings: drscheme:language-configuration:language-settings?
  init: (-> void?)
  kill-termination: (-> void?)

```

This function is similar to `drscheme:eval:expand-program/multiple`. The only difference is that it does not expand the program in the editor; instead the processing function can decide how to expand the program.

`drscheme:get/extend:extend-definitions-canvas`

```
- (drscheme:get/extend:extend-definitions-canvas mixin) ⇒ void?
```

```
mixin : (make-mixin-contract drscheme:unit:definitions-canvas%)
```

- (drscheme:get/extend:extend-definitions-canvas *mixin* *before?*) ⇒ void?
mixin : (make-mixin-contract drscheme:unit:definitions-canvas%)
before? : boolean?

This canvas is used in the top window of drscheme frames. The argument, *before*, controls if the mixin is applied before or after already installed mixins. If unsupplied, this is the same as supplying #f.

```
drscheme:get/extend:extend-definitions-text
```

- (drscheme:get/extend:extend-definitions-text *mixin*) ⇒ void?
mixin : (make-mixin-contract drscheme:unit:definitions-text<%>)
- (drscheme:get/extend:extend-definitions-text *mixin* *before?*) ⇒ void?
mixin : (make-mixin-contract drscheme:unit:definitions-text<%>)
before? : boolean?

This text is used in the top window of drscheme frames.

The argument, *before*, controls if the mixin is applied before or after already installed mixins. If unsupplied, this is the same as supplying #f.

```
drscheme:get/extend:extend-interactions-canvas
```

- (drscheme:get/extend:extend-interactions-canvas *mixin*) ⇒ void?
mixin : (make-mixin-contract drscheme:unit:interactions-canvas%)
- (drscheme:get/extend:extend-interactions-canvas *mixin* *before?*) ⇒ void?
mixin : (make-mixin-contract drscheme:unit:interactions-canvas%)
before? : boolean?

This canvas is used in the bottom window of drscheme frames.

The argument, *before*, controls if the mixin is applied before or after already installed mixins. If unsupplied, this is the same as supplying #f.

```
drscheme:get/extend:extend-interactions-text
```

- (drscheme:get/extend:extend-interactions-text *mixin*) ⇒ void?
mixin : (make-mixin-contract drscheme:rep:text<%>)
- (drscheme:get/extend:extend-interactions-text *mixin* *before?*) ⇒ void?
mixin : (make-mixin-contract drscheme:rep:text<%>)
before? : boolean?

This text is used in the bottom window of drscheme frames.

The argument, *before*, controls if the mixin is applied before or after already installed mixins. If unsupplied, this is the same as supplying #t.

```
drscheme:get/extend:extend-tab
```

- (drscheme:get/extend:extend-tab *mixin*) ⇒ void?
mixin : (make-mixin-contract drscheme:unit:tab%)

- (`drscheme:get/extend:extend-tab` *mixin before?*) \Rightarrow void?
`mixin` : (make-mixin-contract drscheme:unit:tab%)
`before?` : boolean?

This class implements the tabs in drscheme. One is created for each tab in a frame (each frame always has at least one tab, even if the tab bar is not shown)

The argument, *before*, controls if the mixin is applied before or after already installed mixins. If unsupplied, this is the same as supplying #t.

`drscheme:get/extend:extend-unit-frame`

- (`drscheme:get/extend:extend-unit-frame` *mixin*) \Rightarrow void?
`mixin` : (make-mixin-contract drscheme:unit:frame%)
- (`drscheme:get/extend:extend-unit-frame` *mixin before?*) \Rightarrow void?
`mixin` : (make-mixin-contract drscheme:unit:frame%)
`before?` : boolean?

This is the frame that implements the main drscheme window.

The argument, *before*, controls if the mixin is applied before or after already installed mixins. If unsupplied, this is the same as supplying #f.

`drscheme:get/extend:get-definitions-canvas`

- (`drscheme:get/extend:get-definitions-canvas`) \Rightarrow (subclass?/c `drscheme:unit:definitions-canvas`)

Once this function is called, `drscheme:get/extend:extend-definitions-canvas` raises an error, disallowing any more extensions.

`drscheme:get/extend:get-definitions-text`

- (`drscheme:get/extend:get-definitions-text`) \Rightarrow (implementation?/c drscheme:unit:definitions-text<%>)

Once this function is called, `drscheme:get/extend:extend-definitions-text` raises an error, disallowing any more extensions.

`drscheme:get/extend:get-interactions-canvas`

- (`drscheme:get/extend:get-interactions-canvas`) \Rightarrow (subclass?/c `drscheme:unit:interactions-canvas`)

Once this function is called, `drscheme:get/extend:extend-interactions-canvas` raises an error, disallowing any more extensions.

`drscheme:get/extend:get-interactions-text`

- (`drscheme:get/extend:get-interactions-text`) \Rightarrow (implementation?/c drscheme:rep:text<%>)

Once this function is called, `drscheme:get/extend:extend-interactions-text` raises an error, disallowing any more extensions.

`drscheme:get/extend:get-unit-frame`

- (`drscheme:get/extend:get-unit-frame`) \Rightarrow (subclass?/c `drscheme:unit:frame%`)

Once this function is called, `drscheme:get/extend:extend-unit-frame` raises an error, disallowing any more extensions.

`drscheme:help-desk:help-desk`

- (`drscheme:help-desk:help-desk`) \Rightarrow void?
- (`drscheme:help-desk:help-desk key lucky? type mode`) \Rightarrow void?
`key` : string?
`lucky?` : boolean?
`type` : (symbols 'keyword 'keyword+index 'all)
`mode` : (symbols 'exact 'contains 'regexp)
- (`drscheme:help-desk:help-desk key lucky? type`) \Rightarrow void?
`key` : string?
`lucky?` : boolean?
`type` : (symbols 'keyword 'keyword+index 'all)
- (`drscheme:help-desk:help-desk key lucky?`) \Rightarrow void?
`key` : string?
`lucky?` : boolean?

This function opens a help desk window, or brings an already open help desk window to the front. If an argument is specified, that key is searched for.

If no arguments are supplied, this function opens a help-desk window to the starting page, or just brings a help-desk window to the front (without changing what page it is viewing).

If any arguments are supplied, this function opens a help-desk window and searches for `key`, according to `lucky?`, `type`, and `mode`. If the second, third, fourth, and/or fifth arguments are omitted, they default to `awscm#t 'keyword+index` and `'exact`, and `'all` respectively.

`drscheme:help-desk:open-url`

- (`drscheme:help-desk:open-url url`) \Rightarrow void?
`url` : string?

Opens `url` in a new help desk window.

`drscheme:language-configuration:add-language`

- (`drscheme:language-configuration:add-language language`) \Rightarrow void?
`language` : (and/c (is-a?/c `drscheme:language:language<%>`) language-object)

This function can only be called in phase 2 (see section 2 for details).

Adds `language` to the languages offered by DrScheme.

`drscheme:language-configuration:fill-language-dialog`

- (`drscheme:language-configuration:fill-language-dialog panel button-panel language-setting re-center manuals?`) \Rightarrow `drscheme:language-configuration:language-settings?`
`panel` : (is-a?/c `vertical-panel%`)
`button-panel` : (is-a?/c `area-container<%>`)
`language-setting` : `drscheme:language-configuration:language-settings?`
`re-center` = #f : (union false/c (is-a?/c `top-level-window<%>`))
`manuals?` = #f : boolean?

This procedure accepts two parent panels and fills them with the contents of the language dialog. It is used to include language configuration controls in some larger context in another dialog.

The *panel* argument is the main panel where the language controls will be placed. The function adds buttons to the *button-panel* to revert a language to its default settings and to show the details of a language.

The *language-setting* is the default language to show in the dialog. The *re-center* argument is used when the Show Details button is clicked. If that argument is a `top-level-window<%>`, the Show Details callback will recenter the window each time it is clicked. Otherwise, the argument is not used.

If *manuals?* is `#f` the usual language dialog (as seen in the start up drscheme window and from the Choose Language dialog created when drscheme is started up) is shown. If it isn't, the dialog does not have the details and on the right-hand side shows the manual ordering for the chosen language. This is used in Help Desk.

`drscheme:language-configuration:get-languages`

- (`drscheme:language-configuration:get-languages`) \Rightarrow (`listof (is-a?/c drscheme:language:language<%>)`)

This can only be called after all of the tools initialization phases have completed.

Returns the list of all of the languages installed in DrScheme.

`drscheme:language-configuration:get-settings-preferences-symbol`

- (`drscheme:language-configuration:get-settings-preferences-symbol`) \Rightarrow `symbol?`

Returns the symbol that is used to store the user's language settings. Use as an argument to either `preferences:get` or `preferences:set`.

`drscheme:language-configuration:language-dialog`

- (`drscheme:language-configuration:language-dialog` *show-welcome?* *language-settings-to-show* *parent* *manuals?*) \Rightarrow (`union false/c drscheme:language-configuration:language-settings?`)
show-welcome? : `boolean?`
language-settings-to-show : `drscheme:language-configuration:language-settings?`
parent = `#t` : (`union false/c (is-a?/c top-level-window<%>)`)
manuals? = `#f` : `boolean?`

Opens the language configuration dialog. See also `drscheme:language-configuration:fill-language-dialog`.

The *show-welcome?* argument determines if a "Welcome to DrScheme" message and some natural language buttons are shown.

The *language-settings-to-show* argument must be some default language settings that the dialog is initialized to. If unsure of a default, the currently set language in the user's preferences can be obtained via:

(`preferences:get (drscheme:language-configuration:get-settings-preferences-symbol)`)

The *parent* argument is used as the parent to the dialog.

The *manuals?* argument is passed to `drscheme:language-configuration:fill-language-dialog`.

The result is `#f` when the user cancels the dialog, and the selected language if they hit ok.

`drscheme:language-configuration:language-settings-language`

- (`drscheme:language-configuration:language-settings-language` *ls*) \Rightarrow (`union (is-a?/c drscheme:language:language<%>) language-object`)

ls : `drscheme:language-configuration:language-settings?`

Extracts the language field of a language-settings.

`drscheme:language-configuration:language-settings-settings`

- (`drscheme:language-configuration:language-settings-settings ls`) \Rightarrow `any/c`
`ls : drscheme:language-configuration:language-settings?`

Extracts the settings field of a language-settings.

`drscheme:language-configuration:language-settings?`

- (`drscheme:language-configuration:language-settings? val`) \Rightarrow `boolean?`
`val : any/c`

Determines if the argument is a language-settings or not.

`drscheme:language-configuration:make-language-settings`

- (`drscheme:language-configuration:make-language-settings language settings`) \Rightarrow
`drscheme:language-configuration:language-settings?`
`language : (union (is-a?/c drscheme:language:language<%>) language-object)`
`settings : any/c`

This is the constructor for a record consisting of two elements, a language and its settings.

The settings is a language-specific record that holds a value describing a parameterization of the language.

It has two selectors, `drscheme:language-configuration:language-settings-language` and `drscheme:language-configuration:language-settings-settings`, and a predicate, `drscheme:language-configuration:language-settings?`

`drscheme:language:add-snip-value`

- (`drscheme:language:add-snip-value test-value convert-value`) \Rightarrow `void?`
`test-value : (any/c .-> . boolean?)`
`convert-value : (any/c .-> . (is-a?/c snip%))`

Registers a handler to convert values into snips as they are printed in the REPL.

The `test-snip` argument is called to determine if this handler can convert the value and the `convert-value` argument is called to build a snip. Both functions are called on the user's thread and with the user's settings.

`drscheme:language:create-executable-gui`

- (`drscheme:language:create-executable-gui parent program-name show-type? show-base?`)
 \Rightarrow (`union false/c (list/c (symbols (quote no-show) (quote launcher) (quote stand-alone)) (symbols (quote no-show) (quote mred) (quote mzscheme)) string?)`)
`parent : (union false/c (is-a?/c top-level-window<%>))`
`program-name : (union false/c string?)`
`show-type? : (union ((x) (eq? x #t)) (symbols 'launcher 'standalone))`
`show-base? : (union ((x) (eq? x #t)) (symbols 'mzscheme 'mred))`

Opens a dialog to prompt the user about their choice of executable. If `show-type?` is `\#t`, the user is prompted about a choice of executable: stand-alone, or launcher. If `show-base?` is `\#t`, the user is prompted about a choice of base binary: mzscheme or mred.

The `program-name` argument is used to construct the default executable name in a platform-specific manner.

The `parent` argument is used for the parent of the dialog.

The result of this function is `\#f` if the user cancel's the dialog and a list of three items indicating what options they chose. If either `show-type?` or `show-base?` was `\#f`, the corresponding result will be 'no-show, otherwise it will indicate the user's choice.

`drscheme:language:create-module-based-launcher`

- ```
- (drscheme:language:create-module-based-launcher program-filename executable-filename
 module-language-spec transformer-module-language-spec init-code gui? use-copy?)
 => void?
 program-filename : (union path? string?)
 executable-filename : (union path? string?)
 module-language-spec : any/c
 transformer-module-language-spec : any/c
 init-code : any/c
 gui? : boolean?
 use-copy? : boolean?
```

This procedure is identical to `drscheme:language:create-module-based-stand-alone-executable`, except that it creates a launcher instead of a stand-alone executable.

`drscheme:language:create-module-based-stand-alone-executable`

- ```
- (drscheme:language:create-module-based-stand-alone-executable program-filename
  executable-filename module-language-spec transformer-module-language-spec init-code
  gui? use-copy?) => void?
  program-filename : (union path? string?)
  executable-filename : (union path? string?)
  module-language-spec : any/c
  transformer-module-language-spec : any/c
  init-code : any/c
  gui? : boolean?
  use-copy? : boolean?
```

This procedure creates a stand-alone executable in the file `executable-filename` that runs the program `program-filename`.

The arguments `module-language-spec` and `transformer-module-language-spec` specify the settings of the initial namespace, both the transformer portion and the regular portion.

The `init-code` argument is an s-expression representing the code for a module. This module is expected to provide the identifier `init-code`, bound to a procedure of no arguments. That module is required and the `init-code` procedure is executed to initialize language-specific settings before the code in `program-filename` runs.

The `gui?` argument indicates if a MrEd or MzScheme stand-alone executable is created.

The `use-copy?` argument indicates if the initial namespace should be populated with `namespace-require/copy` or `namespace-require`.

`drscheme:language:extend-language-interface`

- ```
- (drscheme:language:extend-language-interface interface default-implementation)
 => void?
 interface : interface?
 default-implementation : ((implementation?/c drscheme:language:language<%>) . ->d . ((%) (subclass?/c %)))
```

This function can only be called in phase 1 (see section 2 for details).

Each language added passed to `drscheme:language-configuration:add-language` must implement *interface*.

The *default-implementation* is a mixin that provides a default implementation of *interface*. Languages that are unaware of the specifics of *extension* use *default-implementation* via `drscheme:language:get-default-mixin`.

`drscheme:language:get-default-mixin`

```
- (drscheme:language:get-default-mixin) => ((implementation?/c drscheme:language:language<%>)
.->d. ((%) (subclass?/c %)))
```

This function can only be called in phase 2 (see section 2 for details).

The result of this function is the composite of all of the *default-implementation* arguments passed to `drscheme:language:extend-language-interface`.

`drscheme:language:get-language-extensions`

```
- (drscheme:language:get-language-extensions) => (listof interface?)
```

This function can only be called in phase 2 (see section 2 for details).

Returns a list of the interfaces passed to `drscheme:language:extend-language-interface`.

`drscheme:language:make-simple-settings`

```
- (drscheme:language:make-simple-settings case-sensitive printing-style fraction-style
show-sharing insert-newlines debugging) => drscheme:language:simple-settings?
 case-sensitive: boolean?
 printing-style: (symbols 'constructor 'quasiquote 'write 'current-print)
 fraction-style: (symbols 'mixed-fraction 'mixed-fraction-e 'repeating-decimal 'repeating-decimal-e)
 show-sharing: boolean?
 insert-newlines: boolean?
 debugging: (symbols 'none 'debug 'debug/profile 'test-coverage)
```

Constructs a simple settings.

`drscheme:language:make-text/pos`

```
- (drscheme:language:make-text/pos text start end) => drscheme:language:text/pos?
 text: (is-a?/c text%)
 start: number?
 end: number?
```

Constructs a text/pos.

`drscheme:language:put-executable`

```
- (drscheme:language:put-executable parent program-filename mred? launcher? title)
=> (union false/c path?)
 parent: (is-a?/c top-level-window<%>)
 program-filename: path?
 mred?: boolean?
 launcher?: boolean?
 title: string?
```

Calls the MrEd primitive `put-file` with arguments appropriate for creating an executable from the file *program-filename*.

The arguments *mred?* and *launcher?* indicate what type of executable this should be (and the dialog may be slightly different on some platforms, depending on these arguments).

The *title* argument is used as the title to the primitive `put-file` or `get-directory` primitive.

`drscheme:language:simple-settings->vector`

- (`drscheme:language:simple-settings->vector` *simple-settings*) ⇒ `vector?`  
*simple-settings*: `drscheme:language:simple-settings?`

Constructs a vector whose first index is the symbol `'struct:simple-settings` and the other elements are the fields of *simple-settings*.

`drscheme:language:simple-settings-annotations`

- (`drscheme:language:simple-settings-annotations` *simple-settings*) ⇒ (symbols `'none` `'debug` `'debug/profile` `'test-coverage`)  
*simple-settings*: `drscheme:language:simple-settings?`

Extracts the debugging setting from a simple-settings.

`drscheme:language:simple-settings-case-sensitive`

- (`drscheme:language:simple-settings-case-sensitive` *simple-settings*) ⇒ `boolean?`  
*simple-settings*: `drscheme:language:simple-settings?`

Extracts the case-sensitive setting from a simple-settings.

`drscheme:language:simple-settings-fraction-style`

- (`drscheme:language:simple-settings-fraction-style` *simple-settings*) ⇒ (symbols `'mixed-fraction` `'mixed-fraction-e` `'repeating-decimal` `'repeating-decimal-e`)  
*simple-settings*: `drscheme:language:simple-settings?`

Extracts the fraction-style setting from a simple-settings.

`drscheme:language:simple-settings-insert-newlines`

- (`drscheme:language:simple-settings-insert-newlines` *simple-settings*) ⇒ `boolean?`  
*simple-settings*: `drscheme:language:simple-settings?`

Extracts the insert-newline setting from a simple-settings.

`drscheme:language:simple-settings-printing-style`

- (`drscheme:language:simple-settings-printing-style` *simple-settings*) ⇒ (symbols `'constructor` `'quasiquote` `'write` `'current-print`)  
*simple-settings*: `drscheme:language:simple-settings?`

Extracts the printing-style setting from a simple-settings.

`drscheme:language:simple-settings-show-sharing`

- (`drscheme:language:simple-settings-show-sharing simple-settings`)  $\Rightarrow$  `boolean?`  
`simple-settings: drscheme:language:simple-settings?`

Extracts the show-sharing setting from a simple-settings.

`drscheme:language:simple-settings?`

- (`drscheme:language:simple-settings? val`)  $\Rightarrow$  `boolean?`  
`val: any/c`

Determines if `val` is a simple-settings.

`drscheme:language:text/pos-end`

- (`drscheme:language:text/pos-end text/pos`)  $\Rightarrow$  `number?`  
`text/pos: drscheme:language:text/pos?`

Selects the ending position from a text/pos.

`drscheme:language:text/pos-start`

- (`drscheme:language:text/pos-start text/pos`)  $\Rightarrow$  `number?`  
`text/pos: drscheme:language:text/pos?`

Selects the starting position from a text/pos.

`drscheme:language:text/pos-text`

- (`drscheme:language:text/pos-text text/pos`)  $\Rightarrow$  `(is-a?/c text%)`  
`text/pos: drscheme:language:text/pos?`

Selects the `text%` from a text/pos.

`drscheme:language:text/pos?`

- (`drscheme:language:text/pos? val`)  $\Rightarrow$  `boolean?`  
`val: any/c`

Returns `#t` if `val` is a text/pos, and `#f` otherwise.

`drscheme:modes:add-mode`

- (`drscheme:modes:add-mode name surrogate repl-submit matches-language`)  $\Rightarrow$  `drscheme:modes:mode`  
`name: string?`  
`surrogate: (union false/c (is-a?/c mode:surrogate-text<%>))`  
`repl-submit: ((is-a?/c drscheme:rep:text%) number? .-> . boolean?)`  
`matches-language: ((union false/c (listof string?)) .-> . boolean?)`

Adds a mode to DrScheme. Returns a mode value that identifies the mode.

The first argument, `name`, is the name of the mode, used in DrScheme's GUI to allow the user to select this mode.

The `surrogate` argument is set to the definitions text and the interactions text (via the `set-surrogate` method) whenever this mode is enabled.

The *repl-submit* procedure is called whenever the user types a return in the interactions window. It is passed the interactions editor and the position where the last prompt occurs. If it returns #t, the text after the last prompt is treated as a program fragment and evaluated, according to the language settings. If it returns #f, the text is assumed to be an incomplete program fragment, and the keystroke is not treated specially.

The *matches-language* predicate is called whenever the language changes. If it returns #t this mode is installed. It is passed the list of strings that correspond to the names of the language in the language dialog.

Modes are tested in the opposite order that they are added. That is, the last mode to be added gets tested first when the filename changes or when the language changes.

See also `drscheme:modes:get-modes`.

`drscheme:modes:get-modes`

- (`drscheme:modes:get-modes`)  $\Rightarrow$  (listof `drscheme:modes:mode?`)

Returns all of the modes currently added to DrScheme.

See also `drscheme:modes:add-mode`.

`drscheme:modes:mode-matches-language`

- (`drscheme:modes:mode-matches-language mode`)  $\Rightarrow$  ((union false/c (listof string?)) . -> . boolean?)  
`mode` : `drscheme:modes:mode?`

Extracts the language matching predicate of the mode.

See also `drscheme:modes:add-mode`.

`drscheme:modes:mode-name`

- (`drscheme:modes:mode-name mode`)  $\Rightarrow$  string?  
`mode` : `drscheme:modes:mode?`

Extracts the name of the mode.

See also `drscheme:modes:add-mode`.

`drscheme:modes:mode-repl-submit`

- (`drscheme:modes:mode-repl-submit mode`)  $\Rightarrow$  any  
`mode` : `drscheme:modes:mode?`

Extracts the repl submission predicate of the mode.

See also `drscheme:modes:add-mode`.

`drscheme:modes:mode-surrogate`

- (`drscheme:modes:mode-surrogate mode`)  $\Rightarrow$  (union false/c (is-a?/c `mode:surrogate-text<%>`))  
`mode` : `drscheme:modes:mode?`

Extracts the surrogate of the mode.

See also `drscheme:modes:add-mode`.

`drscheme:modes:mode?`

- (`drscheme:modes:mode? val`)  $\Rightarrow$  `boolean?`  
`val` : `any/c`

Determines if `val` is a mode.

`drscheme:rep:current-rep`

- (`drscheme:rep:current-rep`)  $\Rightarrow$  (`union false/c (is-a?/c drscheme:rep:text%)`)

This is a parameter whose value should not be set by tools. It is initialized to the repl that controls this evaluation in the user's thread.

It only returns `#f` if the program not running in the context of a repl (eg, the test suite window).

`drscheme:rep:current-value-port`

- (`drscheme:rep:current-value-port`)  $\Rightarrow$  (`union false/c port?`)

This is a parameter whose value is a port that prints in the REPL in blue. It is used to print the values of toplevel expressions in the REPL.

It is only initialized on the user's thread

`drscheme:rep:get-drs-bindings-keymap`

- (`drscheme:rep:get-drs-bindings-keymap`)  $\Rightarrow$  (`is-a?/c keymap%`)

Returns a keymap that bindings various DrScheme-specific keybindings. This keymap is used in the definitions and interactions window.

Defaultly binds `C-x;0` to a function that switches the focus between the definitions and interactions windows. Also binds `f5` to Execute and `f1` to Help Desk.

`drscheme:teachpack:install-teachpacks`

- (`drscheme:teachpack:install-teachpacks teachpack-cache`)  $\Rightarrow$  `void?`  
`teachpack-cache` : `drscheme:teachpack:teachpack-cache?`

Installs the teachpack cache in the current namespace. Passing `'drscheme:teachpacks` to `preferences:get` returns the user's currently selected TeachPacks.

`drscheme:teachpack:teachpack-cache-filenames`

- (`drscheme:teachpack:teachpack-cache-filenames teachpack-cache`)  $\Rightarrow$  (`listof path?`)  
`teachpack-cache` : `drscheme:teachpack:teachpack-cache?`

Returns the list of filenames for the teachpacks in `teachpack-cache`.

See also `drscheme:teachpack:install-teachpacks`.

`drscheme:teachpack:teachpack-cache?`

- (`drscheme:teachpack:teachpack-cache? val`)  $\Rightarrow$  `boolean?`  
`val` : `any/c`

Determines if `val` is a teachpack cache or not.

`drscheme:unit:add-to-program-editor-mixin`

- (`drscheme:unit:add-to-program-editor-mixin` *mixin*) ⇒ `void?`  
`mixin: ((subclass?/c text%) .-> . (subclass?/c text%))`

This function can only be called in phase 1 (see section 2 for details).

Adds *mixin* to the result of `drscheme:unit:get-program-editor-mixin`.

`drscheme:unit:get-program-editor-mixin`

- (`drscheme:unit:get-program-editor-mixin`) ⇒ `((subclass?/c text%) .-> . (subclass?/c text%))`

Returns a mixin that must be mixed in to any *text%* object that might contain program text (and thus can be in the source field of some syntax object).

See also `drscheme:unit:add-to-program-editor-mixin`.

`drscheme:unit:open-drscheme-window`

- (`drscheme:unit:open-drscheme-window`) ⇒ `(is-a?/c drscheme:unit:frame%)`
- (`drscheme:unit:open-drscheme-window` *filename*) ⇒ `(is-a?/c drscheme:unit:frame%)`  
`filename: (union string? false/c)`

Opens a drscheme frame that displays *filename*, or nothing if *filename* is #f or not supplied.

### 3.35 Contract Helpers

`language-object`

```
(object-contract
 (config-panel
 (-> (is-a?/c area-container<%>) (case-> (-> any/c void?) (-> any/c))))
 (create-executable
 (->
 any/c
 (union (is-a?/c dialog%) (is-a?/c frame%))
 path?
 drscheme:teachpack:teachpack-cache?
 void?))
 (default-settings (-> any/c))
 (default-settings? (-> any/c boolean?))
 (order-manuals (-> (listof bytes?) (values (listof bytes?) boolean?)))
 (front-end/complete-program
 (-> input-port? any/c drscheme:teachpack:teachpack-cache? (-> any/c)))
 (front-end/interaction
 (-> input-port? any/c drscheme:teachpack:teachpack-cache? (-> any/c)))
 (get-language-name (-> string?))
 (get-language-numbers (-> (cons/c number? (listof number?))))
 (get-language-position (-> (cons/c string? (listof string?))))
 (get-language-url (-> (union false/c string?)))
 (get-one-line-summary (-> string?))
 (get-comment-character (-> (values string? char?)))
 (get-style-delta
```

```
(->
 (union
 false/c
 (is-a?/c style-delta%)
 (listof (list/c (is-a?/c style-delta%) number? number?))))
(marshall-settings (-> any/c printable/c))
(on-execute (-> any/c (-> (-> any) any) any))
(render-value (-> any/c any/c output-port? void?))
(render-value/format (-> any/c any/c output-port? number? any))
(unmarshall-settings (-> printable/c any)))
```

- *a-language-object* ⇒ contract

# Index

- [:, 22](#)
- [add-show-menu-items, 8, 43](#)
- [adding languages to DrScheme, 4](#)
- [after-delete, 34, 49](#)
- [after-insert, 34, 49](#)
- [after-set-next-settings, 40](#)
- [alignment, 16, 42, 43](#)
- ['auto-hscroll, 13–15, 39, 48](#)
- ['auto-vscroll, 13–15, 39, 48](#)
- [auto-wrap, 7, 14, 40, 49](#)
  
- [border, 16, 42, 43](#)
- [break button, 6](#)
- [break-callback, 43, 50](#)
- [breaking, 6](#)
  
- [can-close?, 50](#)
- [canvas
  - \[scroll bars, 39\]\(#\)](#)
- [canvas%, 12](#)
- [change-to-file, 43](#)
- [clear-annotations, 32, 52](#)
- ['combo, 13–15, 39, 48](#)
- [config-panel, 16, 22, 29](#)
- [context, 15](#)
- ['control-border, 13–15, 39, 48](#)
- [create-executable, 16](#)
  
- [default-settings, 17, 22, 29](#)
- [default-settings?, 17, 22, 29](#)
- ['deleted, 13–15, 39, 48](#)
- [disable-evaluation, 32, 50](#)
- [display-results, 35](#)
- [do-many-evals, 35](#)
- [do-many-text-evals, 35](#)
- [drscheme-language-modules, 4](#)
- [drscheme-language-numbers, 4](#)
- [drscheme-language-one-line-summaries, 4](#)
- [drscheme-language-positions, 4](#)
- [drscheme-language-readers, 4](#)
- [drscheme-language-urls, 4](#)
- [drscheme:debug:add-prefs-panel, 53](#)
- [drscheme:debug:get-cm-key, 53](#)
- [drscheme:debug:hide-backtrace-window, 53](#)
- [drscheme:debug:make-debug-error-display-handler, 57](#)
- [drscheme:debug:make-debug-eval-handler, 53](#)
  
- [drscheme:debug:open-and-highlight-in-file, 54](#)
- [drscheme:debug:profile-definitions-text-mixin, 7](#)
- [drscheme:debug:profile-interactions-text-mixin, 7](#)
- [drscheme:debug:profile-unit-frame-mixin, 7](#)
- [drscheme:debug:profiling-enabled, 54](#)
- [drscheme:debug:show-backtrace-window, 54](#)
- [drscheme:debug:show-error-and-highlight, 54](#)
- [drscheme:eval:build-user-eventspace/custodian, 54](#)
- [drscheme:eval:expand-program, 55](#)
- [drscheme:eval:expand-program/multiple, 55](#)
- [drscheme:eval:get-snip-classes, 56](#)
- [drscheme:eval:set-basic-parameters, 56](#)
- [drscheme:eval:traverse-program/multiple, 56](#)
- [drscheme:frame:<%>, 8](#)
- [drscheme:frame:basics-mixin, 9](#)
- [drscheme:frame:basics<%>, 9](#)
- [drscheme:frame:mixin, 11](#)
- [drscheme:frame:name-message%, 12](#)
- [drscheme:get/extend:base-definitions-canvas%, 12](#)
- [drscheme:get/extend:base-definitions-text%, 14](#)
- [drscheme:get/extend:base-interactions-canvas%, 14](#)
- [drscheme:get/extend:base-interactions-text%, 15](#)
- [drscheme:get/extend:base-tab%, 15](#)
- [drscheme:get/extend:base-unit-frame%, 16](#)
- [drscheme:get/extend:extend-definitions-canvas, 56](#)
- [drscheme:get/extend:extend-definitions-text, 57](#)
- [drscheme:get/extend:extend-interactions-canvas, 57](#)
- [drscheme:get/extend:extend-interactions-text, 57](#)
- [drscheme:get/extend:extend-tab, 57](#)
- [drscheme:get/extend:extend-unit-frame, 58](#)



- drscheme:unit:get-program-editor-mixin, 68
- drscheme:unit:interactions-canvas%, 48
- drscheme:unit:open-drscheme-window, 68
- drscheme:unit:program-editor-mixin, 49
- drscheme:unit:tab<%>, 50
- drscheme:unit:tab%, 52
  
- edit-menu:between-find-and-preferences, 9
- edit-menu:between-select-all-and-find, 44
- editor, 12, 14, 38, 48
- editor-canvas%, 38
- editors
  - hooks, 34, 49
  - modified, 41
- enable-evaluation, 32, 50
- enabled, 12, 14, 16, 39, 42, 43, 48
- ensure-defs-shown, 44
- ensure-rep-shown, 32, 44
- execute-callback, 44
- expanding user programs, 6
  
- file-menu:between-open-and-revert, 9, 44
- file-menu:between-print-and-close, 10, 44
- file-menu:between-save-as-and-print, 45
- file-menu:new-callback, 10
- file-menu:new-string, 10
- file-menu:open-callback, 10
- file-menu:open-string, 10
- file-menu:print-string, 45
- file-menu:save-as-string, 45
- file-menu:save-string, 45
- filename, 16, 42, 43
- files
  - names, 40
- first-opened, 17
- 'float, 16, 42, 43
- front-end/complete-program, 17, 25
- front-end/interaction, 18, 26
  
- get-additional-important-urls, 11
- get-break-button, 45
- get-breakables, 32, 51
- get-button-panel, 45
- get-canvas, 46
- get-canvas%, 46
- get-comment-character, 18
- get-current-tab, 41
- get-definitions-canvas, 41
- get-definitions-text, 41
- get-definitions/interactions-panel-parent, 46
- get-defs, 51
- get-directory, 33, 51
- get-editor, 46
- get-editor%, 46
- get-enabled, 51
- get-error-range, 35
- get-execute-button, 46
- get-frame, 51
- get-init-code, 22, 29
- get-interactions-canvas, 41
- get-interactions-text, 42
- get-ints, 51
- get-keymaps, 34
- get-language-name, 18, 26
- get-language-numbers, 18, 23, 26, 27
- get-language-position, 19, 23, 27, 28
- get-language-url, 19
- get-module, 23, 27, 28
- get-next-settings, 40
- get-one-line-summary, 19, 23, 27, 28
- get-show-menu, 8
- get-special-menu, 42
- get-style-delta, 19
- get-tab, 40
- get-text-to-search, 47
- get-transformer-module, 23, 30
- get-user-custodian, 36
- get-user-eventspace, 36
- get-user-language-settings, 36
- get-user-namespace, 36
- get-user-thread, 36
  
- height, 16, 42, 43
- help-menu:about-callback, 11
- help-menu:about-string, 11
- help-menu:before-about, 11
- help-menu:create-about?, 11
- 'hide-hscroll, 13-15, 39, 48
- 'hide-menu-bar, 16, 42, 43
- 'hide-vscroll, 13-15, 39, 48
- highlight-errors, 36
- highlight-errors/exn, 37
- horiz-margin, 12, 14, 39, 48
- horizontal-inset, 12, 14, 38, 48
  
- initialize-console, 37
- insert-prompt, 37
- is-current-tab?, 52
  
- keyboard focus
  - notification, 13, 15
- keymaps

in an editor, 7, 49  
 kill-evaluation, 37  
 label, 12, 14, 38, 48  
 language-object, 68  
 line-count, 12, 14, 38, 48  
 line-spacing, 7, 14, 40, 49  
 make-searchable, 47  
 marshall-settings, 20, 23, 30  
 'mdi-child, 16, 42, 43  
 'mdi-parent, 16, 42, 43  
 'metal, 16, 42, 43  
 min-height, 12, 14, 16, 39, 42, 43, 48  
 min-width, 12, 14, 16, 39, 42, 43, 48  
 modes, 6  
 '|MrEd:wheelStep|, 39  
 needs-execution, 33  
 'no-border, 13–15, 39, 48  
 'no-caption, 16, 42, 43  
 'no-hscroll, 13–15, 39, 48  
 'no-resize-border, 16, 42, 43  
 'no-system-menu, 16, 42, 43  
 'no-vscroll, 13–15, 39, 48  
 not-running, 8  
 on-close, 37, 47, 52  
 on-execute, 20, 24, 26, 30  
 on-focus, 13, 15  
 on-size, 47  
 on-tab-change, 42  
 order-manuals, 21  
 parent, 12, 14, 16, 38, 42, 43, 48  
 phase1, 2  
 phase2, 2  
 queue-output, 37  
 render-value, 21, 24, 30  
 render-value/format, 21, 24, 31  
 reset-console, 37  
 reset-highlighting, 38  
 reset-offer-kill, 33, 52  
 'resize-corner, 13–15, 39, 48  
 run-in-evaluation-thread, 38  
 running, 8  
 scheme mode, 6  
 scrolls-per-page, 12, 14, 38, 48  
 set-breakables, 33, 52  
 set-filename, 40  
 set-message, 12  
 set-modified, 41  
 shutdown, 38  
 spacing, 16, 42, 43  
 still-untouched?, 47  
 stretchable-height, 12, 14, 16, 39, 42, 43, 48  
 stretchable-width, 12, 14, 16, 39, 42, 43, 48  
 style, 12, 14, 16, 38, 42, 43, 48  
 style lists  
     in an editor, 7, 49  
 tab-stops, 7, 14, 40, 49  
 tool-icons, 2  
 tool-names, 2  
 tool-urls, 2  
**tool.ss**, 2  
 'toolbar-button, 16, 42, 43  
 'transparent, 13–15, 39, 48  
 unmarshall-settings, 21, 24, 31  
 update-running, 33  
 update-save-button, 47  
 update-save-message, 48  
 update-shown, 9, 48  
 use-mred-launcher, 24, 31  
 use-namespace-require/copy?, 24  
 vert-margin, 12, 14, 39, 48  
 vertical-inset, 12, 14, 38, 48  
 View menu, 8  
 wait-for-io-to-complete, 38  
 wait-for-io-to-complete/user, 38  
 wheel on mouse, 39  
 wheel-step, 12, 14, 38, 48  
 width, 16, 42, 43  
 x, 16, 42, 43  
 y, 16, 42, 43